COMPARATIVE ANALYSIS OF VARIOUS ALGORITHMS WITH DEEP NEURAL NETWORK FOR INTRUSION DETECTION SYSTEM

PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF A DEGREE OF

MASTER OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING

Submitted By
PRATEEK SHRIVASTVA
2K21/CSE/26

under the supervision of

Dr. R.K. YADAV
(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of
Engineering)
Bawana Road, Delhi-110042
May 2022

MTECH (COMPUTER SCIENCE & ENGINEERING)

PRATEEK SHRIVASTAVA

2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, **Prateek Shrivastava**, Roll No. 2K21/CSE/26 student of M.Tech (Computer Science and Engineering), hereby declare that the Project Dissertation titled **"Comparative Analysis of Various Algorithms with Deep Neural Network for Intrusion Detection System"** which is being submitted by me to Delhi Technological University, Delhi, in partial fulfillment of requirements for the degree of Master of Technology in Computer Science and Engineering legitimate record of my work and is not copied from any source. The work contained in this report has not been submitted to any other University/Institution for the award of any degree.

Place: Delhi
Date: 29 May2022

Prateek Shrivastava
(2K21/CSE/26)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CERTIFICATE

I, hereby certify that the Project titled **"Comparative Analysis of Various Algorithms with Deep Neural Network for Intrusion Detection System",** submitted by Prateek Shrivastava, Roll No. 2K21/CSE/26, Department of Computer Science & Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of M.Tech in Computer Science and Engineering is a genuine record of the project work carried out by the student under my supervision. To the best of my knowledge, this work has not been submitted in part or full for any Degree to this University or elsewhere.

Place: Delhi
May 2023

Dr. R. K. Yadav
Assistant Professor
(Supervisor)

ABSTRACT

An intrusion detection system (IDS) uses a lot of machine learning techniques to find and classify cyberattacks at the organization and host levels in a timely and independent way. However, a scalable solution is required because aggressive assaults are constantly evolving and occur in such large numbers. The network protection local area can secretly get to different malware datasets for additional investigation. Be that as it may, no ongoing review has analyzed the presentation of different AI calculations utilizing an assortment of secretly open datasets from top to bottom. Due to the dynamic nature of malware and its constantly shifting attack strategies, the privately provided malware datasets must be properly optimized and benchmarked. This investigation focuses on a deep neural network (DNN), also known as a deep knowledge model, to develop an adaptable and effective intrusion detection system (IDS) for describing and classifying shifting and unexpected cyberattacks. It is anticipated to evaluate various datasets created over time using static and dynamic procedures due to the constant change in network structure and the rapid definition of attacks. This kind of research can connect to the swish algorithm, which can accurately predict threats in the future. On several privately accessible standard malware datasets, a comprehensive evaluation of trials of DNNs and other conventional machine learning classifiers is presented. The following hyperparameter selection methods are used to select the ideal network parameters and network topologies for DNNs using the KDDCup 99 dataset. Over the course of one thousand DNN experiments, the knowledge rate fluctuates between 0.01 and 0.5. To comply with the standard, the DNN model that performed well on KDDCup 99 is applied to a variety of datasets, including NSL-KDD, UNSW-NB15, Kyoto, WSN-DS, and CICIDS 2017. Through several protected layers, our DNN model acquires the abstract and high-dimensional point representation of the IDS data. DNNs perform better in standing out from ordinary ML classifiers, according to exhaustive exploratory testing. Finally, we provide scale-crossbred-IDS-Alert Net, a cold-thoroughbred DNNs framework that is significantly scalable, can be used in real time to cover network business and host-location events effectively, and is able to proactively notify potential cyberattacks.

ACKNOWLEDGEMENT

The success of a project is not solely dependent on the efforts of the person to whom the project is assigned, but also on the support and guidance of those who contributed to its completion. I would like to express my gratitude to all those who have helped me in this research and inspired me during my study.

I am particularly grateful to my Research Supervisor, Dr. R. K. Yadav, for his encouragement, support, patience, and guidance in this thesis work. His valuable feedback and guidance enabled me to complete this study in an improved manner.

I also extend my heartfelt thanks to my wife & daughter as they were the key source of my energy to complete my research work in a meticulous manner. I would also extend my warm thanks to my family and friends for their unwavering moral support, which has been instrumental in keeping me motivated to continue with this study and to achieve the expected results throughout my research work.

Prateek Shrivastava

(2K21/CSE/26)

CONTENT

LIST OF FIGURES

LIST OF ABBREVIATIONS

ICT	Information and Communication Technology
IDS	Intrusion Detection System
NIDS	Network-Based Intrusion Detection System
HIDS	Host-Based Intrusion Detection System
DNN	Deep Neural Network
NLP	Natural Language Processing
LD	Linux Datasets
WD	Windows Datasets
SVM	Support Vector Machine
DDoS	Distributed Denial of Service
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
NB	Nave Bayes
NSL	Nuziveedu Seeds Limited
KDD	Knowledge Discovery in Databases

CHAPTER 1: INTRODUCTION

1.1 Overview

Information and communication technology (ICT) systems and networks handle sensitive colorful stoner data, making them susceptible to color attacks from both internal and external interference (1). These attacks can be made by hand or by machine, and their obfuscations vary, allowing data breaches to go unnoticed. For instance, the Yahoo information break brought about a deficiency of $ 350 million, while the Bitcoin hack brought about a deficiency of around $ 70 million (2). Comparable hacks are consistently creating with additional mind boggling calculations as innovation, programming, and arrange geographies develop, particularly current advancements in the Internet of Things (IoT)(4). Pernicious cyberattacks present significant security challenges, requiring the improvement of a new, versatile, and more dependable intrusion detection system (IDS). An IDS is a groundbreaking interruption identification innovation that recognizes and groups interruptions, attacks, or breaks of safety programs progressively at the organization and host levels. The two types of intrusion detection are based on protruding behaviors: network-grounded intrusion detection system (NIDS) and have grounded intrusion detection system(HIDS)(5).

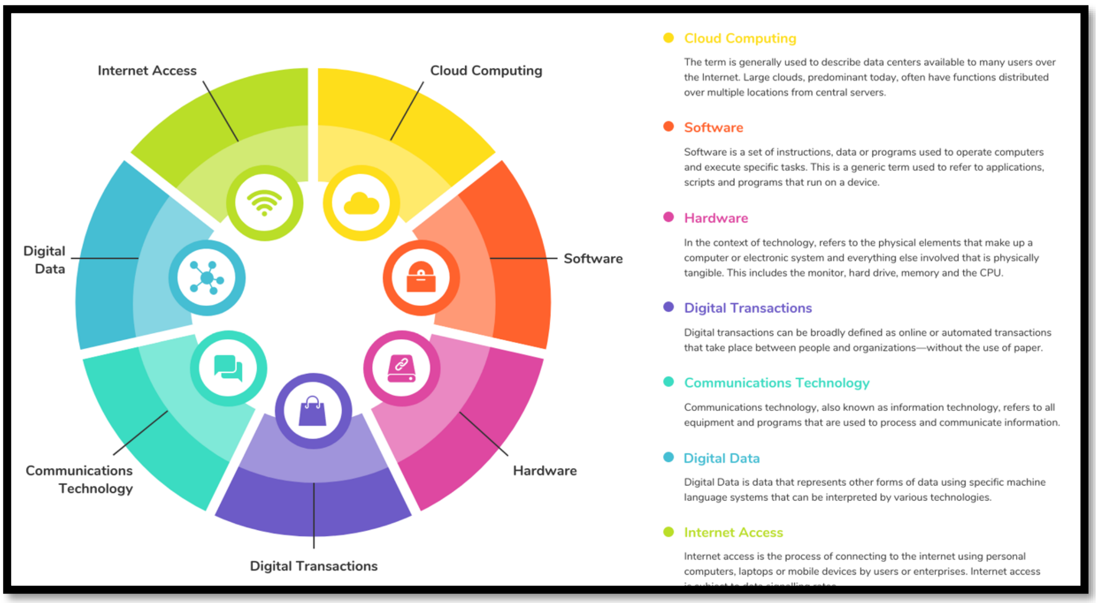


Figure 1: Component of ICT

1.2 Problem Statement

Cybercrime is a prime example of misconduct that occurs across borders. Scallywags might cause any type of mischief anyplace on the earth without leaving their homework area since PC networks interface all nations of the world. The potential harm is surprising distributed, ranging from individuals being unable to access their personal computers for a few hours or accidentally finding material that is bigoted or indecent on the Internet to the theft of proprietary innovations, the disruption of public government websites, and the appearance of state secrets online. Monetary losses range from compelled losses of two to three hundred dollars to enormous losses brought on by digital harm or misrepresentation. Cybercrime poses a threat of psychological oppressor attacks that could wipe out a significant portion of the internet and cause a global financial and social catastrophe as the internet becomes more integrated with everyday life. [6] In addition, we will investigate the Extra Territoriality Act of 2000 and provide a fundamental evaluation of the Act, which aims to include weather conditions.The act is just a paper record unless it is competent enough to run more regional wards. [7] The use of deep learning algorithms and machine learning techniques like CNN to identify various kinds of cyberattacks. When compared to other data mining techniques, machine learning algorithms' cyber-attack detection accuracy is quite high. Methods based on machine learning determine the specific flood attack detection. The Python-coded Jupiter Anaconda Navigator Simulator and the CNN and DEEP learning algorithms provide precise results for identifying various cyberattacks. [8]

1.3 Project Objectives

- Displaying a deep neural network (DNN) to join NIDS and HIDS for proactive cyberattack recognition is proposed as a productive profound proficiency technique. This review ascertains the adequacy of vivid old-style ML methods and DNNs on bright NIDS and HIDS datasets to decide if an assault with matching assault orders made network business substance be typical or strange.

- Utilizing host-position occasions, otherwise called framework calls, the high-level course book portrayal styles of natural language processing (NLP) are examined with a definitive objective of saving framework call succession data and catching logical and semantic similitudes. The overall presentation of these styles is looked at utilizing the ADFA-LD and ADFA-WD datasets.[3]

- This study utilizes dynamic standard datasets for a relative preliminary. This is for the most part because each dataset encounters an extent of hardships like data debasement, business variety, differences, old and present day assaults.

- SHIA, a versatile mutt interruption identification system, is introduced to consequently distinguish horrible elements and issue relevant admonitions to organize heads by reusing a huge amount of host-position and organization position occasions. The proposed system is exceptionally versatile on normal equipment garçon, and execution can be additionally upgraded to deal with gigantic measures of information continuously applications by integrating extra computational assets.

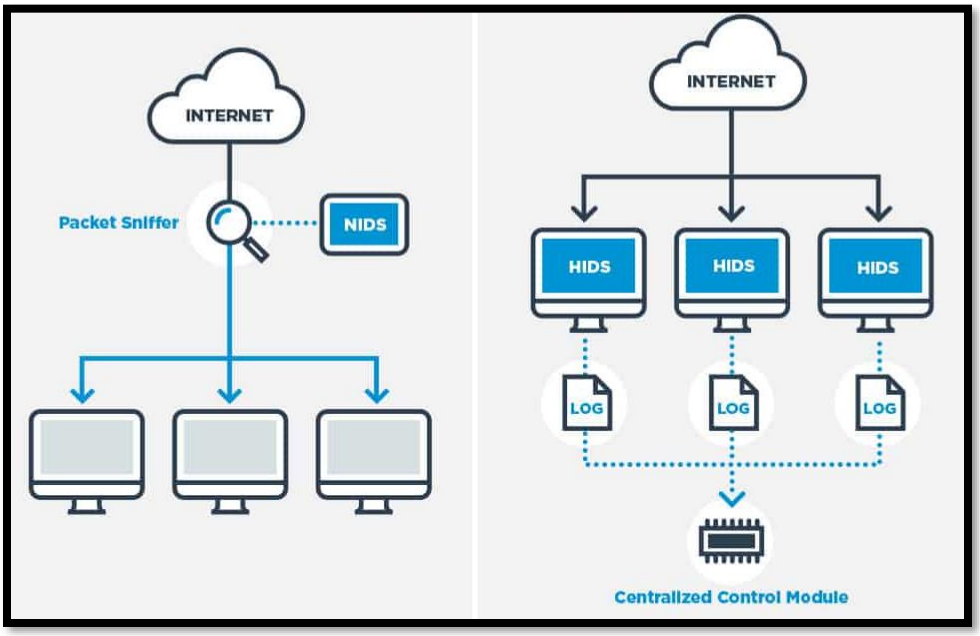


Figure 2: NIDS & HIDS [5]

1.4 SVM Algorithm

In ML, support vector machines (SVMs, otherwise called help heading networks(1)) are managed data models that make sense of information for type and inversion study. Vladimir

Vapnik and associates grown SVMs at AT&T Bell Laboratories (Boseretal., Guyonetal, 1992, 1993, (1) Vapniketal., Cortes and Vapnik, 1995 Based on mathematical information materials or the VC proposition projected by Vapnik (1982, 1995) and Chervonenkis (1974), (award necessary) A SVM fitting prediction fabricates a model that allocates new models to one or the other orders likely a bunch of fitting tests, making it a non-probabilistic binary direct classifier (still the event that styles like Platt weighing live to include SVM in a probabilistic kind background). To improve the sphere of the contrast 'tween two together orders, SVM maps fitting tests to focus presage. Depending on that side of the breach they attack, new samples are still received into the alike room and envisioned expected in the order. By inevitably plan their inputs into extreme-spatial point rooms utilizing the essence arrangement, SVMs can efficiently kill non-direct type apart from direct type. Hava Siegelmann and Vladimir Vapnik formulated the help heading assembling (2) process, that takes advantage of the calculations assisting headings collected in the help heading machines prediction to order unlabeled dossier.(Citation necessary) These educational indexes include independent news patterns, that endeavor to disclose usual assemblage of facts into gatherings and, also, to produce new facts taking everything in mind these groups.

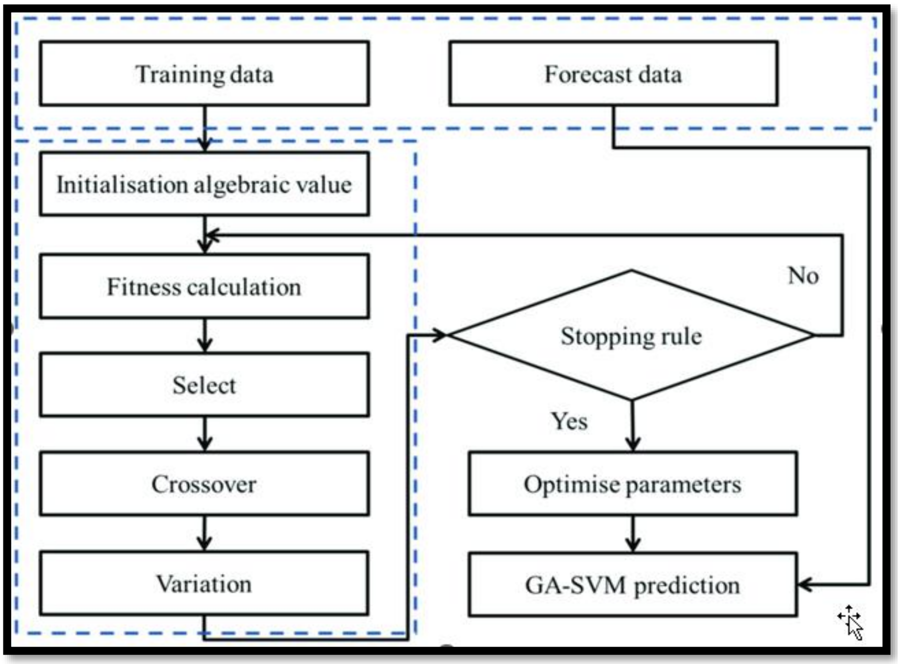


Figure 3: SVM Algorithm Flowchart

1.5 Random Forest Algorithm

An ensemble proficiency arrangement for classification, regression, and additional tasks is Random forests, also known as arbitrary decision forests. It everything by preparation a lot of conclusion timbers. The class namely named for one extreme forests for support questions is the subject of the dictatorial thickets. The mean or rational prophecy of the various shrubs is restored for regression tasks.(1)(2) Random decision forests right choice seedlings' slant to overfit their development set.(3) Random forests beat conclusion wood usually, but their deliciousness is inferior to of grade-improved timbers (Citation necessary). However, dossier facial characteristics can influence by virtue of what well they act. In 1995, Tin Kam Ho grown the first arrangement for dictatorial decision forests, engaging the dictatorial subspace structure. Ho interpreted the form as a habit to ask Eugene Kleinberg's "theory of probability boundary" approach to connect. In 2006, Leo Bre Because they support superior prognoses over a off-course range of data accompanying littlest arrangement, chance forest models are commonly secondhand as flight data recorder models in trades.

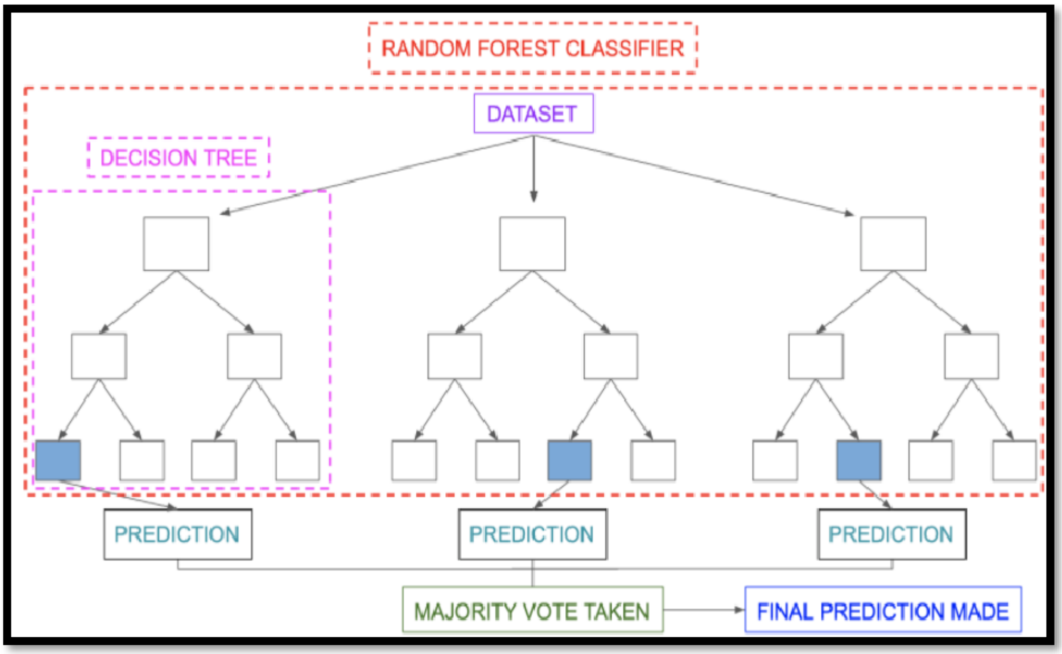


Figure 4: Random Forest Algorithm

1.6 DNN Algorithm

Deep Learning is a subdivision of the best kind of machine learning orders established likeness knowledge and artificial neural networks. Education counseling maybe either hindered, to a certain extent-controlled, or alone.(2) Deep Learning foundations, e.g., deep mind arranging, deep confidence arrangements, deep upholding skillfulness, intermittent sintellect arrangements, convolutional intelligence arranging, and laboratories have happened handled in fields, for instance, PC view, discourse admission, common sound management, ML, bioinformatics, drug plan, dispassionate picture inspection, surroundings shrewdness, material estimate, and prepackaged game projects, transfering results.(3)(4)(5) Data management and transmitted agreement knocks in routine foundations inspired artificial neural networks (ANNs). There are many habits at which point ANNs clash from organic acumen. The term "deep" in deep learning refers to the

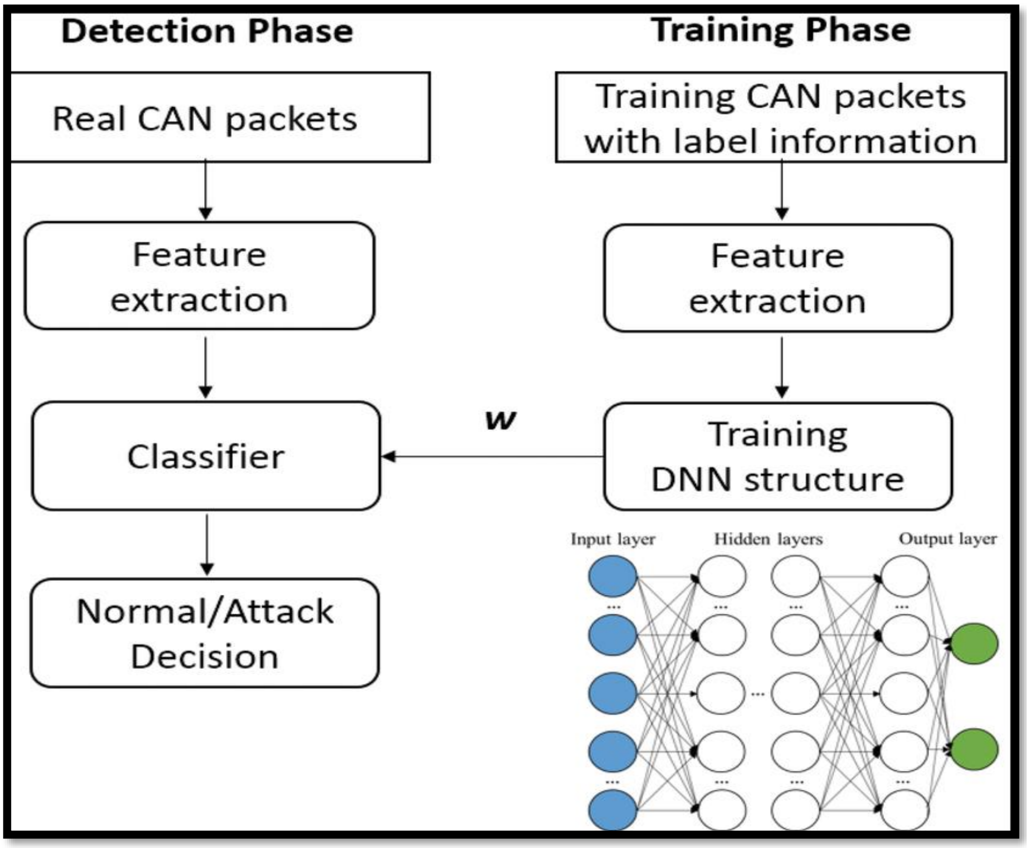


Figure 5: DNN Algorithm Flowchart

exercise of abundant network coatings, in as much as the instinctive mind of the adulthood of living mammals is vital (flexible) and parallel. Artificial neural networks, specifically, likely expected emblematic and constant. Early test proved that a direct perceptron can't be a comprehensive classifier, still an institution accompanying a non polynomial playacting wherewithal and individual endless reach stored continuously subcaste can. The contemporary story of learning is named deep learning, and it has to do with extremely many tiers of a restricted amount. This admits for optimum killing and proficient movement while claiming hypothetical wholeness under moderate environments. For the purposes of productiveness, trainability, and understandability, deep knowledge coatings are likewise granted to distinct considerably from physiologically conversant connectionist models and expected various. The act of eliminating detracting instances, patterns, or additional appropriate dossier from news expanded in two together terrestrial and materialistic layoff points is famous as spatiotemporal facts excavating. The fields of criminology, drugs, conveyance, public security, and many possible choices form thorough use of geographical-worldly dossier excavating actions. An important component of relating to space dossier excavating is geographical grouping. Spatial Bunching is the accumulating of equivalent occasion positions across extent later periods. Spatial Area of interest labeling is a subgroup of Spatial Grouping.

CHAPTER 2: RELATED WORK

2.1 Network Intrusion Detection

Interruption recognition is a progressive technique that attempts to offer PCs and information networks with a sense of safety while empowering them to work in their present "open" structure. The goal of intrusion detection is to find instances of internal and external attackers exploiting computer systems in ways that are not authorized. However, the difficulty of intrusion detection has increased with the proliferation of computer networks. Due to expanded interconnection across PC frameworks, aggressors may now get away from discovery all the more without any problem. The goal of intrusion detection systems (IDSs) is to find unusual user behavior and unapproved activity that could compromise the security of the system. IDSs are based on the idea that an attacker's behavior will be significantly different from that of a legal user and that a lot of unauthorised activity will be detected.

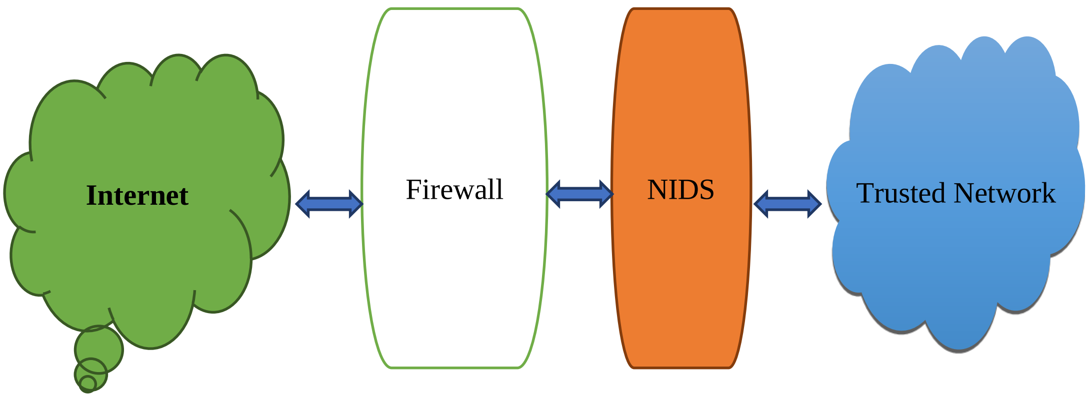


Figure 6: Architecture of IDS

Most of the time, statistical anomaly and rule-based abuse models are used by intrusion detection systems (IDSs) to find intrusions. Various institutes have developed and tested a number of prototype IDSs in operational systems. The characteristics of host-based and network-based intrusion detection systems are contrasted in this study. The majority of network-based intrusion detection systems employ concealed network traffic for intrusion detection, with some additionally using host inspection trails. Host-based intrusion detection systems rely on inspection trails from the host operating system as the primary source of input to identify suspicious behavior. An illustration of a statistical anomaly detection method that is frequently utilized in intrusion detection systems is also provided in this article.

2.2 Mitigation of distributed denial of service attacks.

DDoS goes after these days are like traditional volumetric attacks in that their principal objective is to obstruct admittance to the Internet by flooding it with traffic to over-burden a server and make it impossible. On the other hand, modern assaults are more intricate and capable of performing multiple tasks simultaneously. For

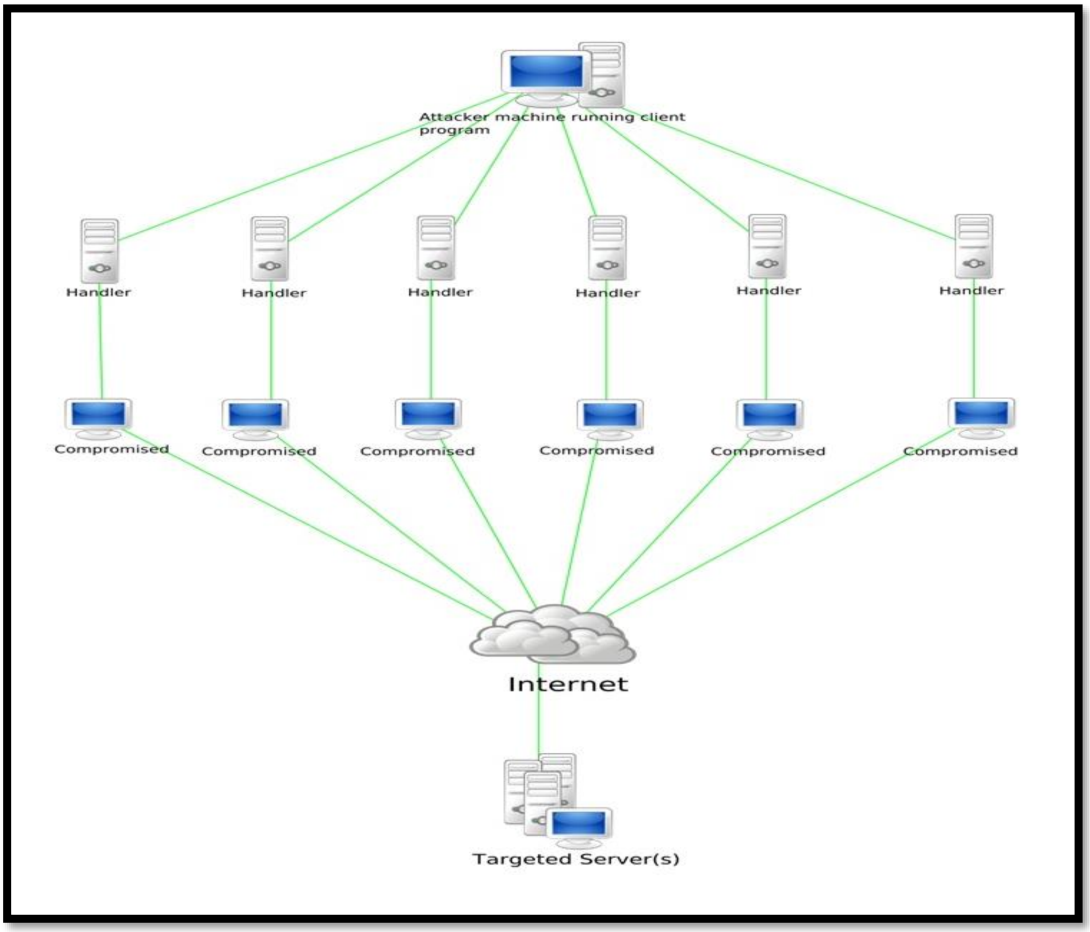


Figure 7: Distributed Denial of Service (DDoS) Attack

For instance, they might use a volumetric attack. They might break through a firewall and get into the network to steal sensitive data while the IT team is distracted. a comprehensive investigation and analysis of how intrusions can be identified using machine learning technologies. In the digital environment of today, intrusion detection is a crucial security concern. Albeit not generally effective, approaches in view of machine learning (ML) have been created to recognize interruptions. To distinguish the principal deterrents related with perceiving obtrusive ways of behaving, this study led inside and out assessments of numerous ML methods. The research provided attack sequences and attack attribute schedules for each attack. It additionally examined difficulties connected with identifying low-recurrence assaults utilizing network assault datasets and gave conceivable upgrade choices. The detection capabilities of machine learning (ML) systems for various forms of assault were evaluated, and limits associated with each category were established. In addition, the study presented novel approaches to ML-based intrusion detection and discussed numerous data mining tools for machine learning.

2.3 The utilization of long short-term memory recurrent neural networks for intrusion detection.

Organizations now have access to new opportunities thanks to the widespread use of web operations. In any case, the expanded activity raises the gamble of being focused on by bushwhackers, conceivably making horrendous harm to the framework. It is essential to raise awareness of threats to online operations to reduce this risk. Web intrusion detection systems (IDSs) are basic in safeguarding frameworks from both outer and inward dangers. However, there are significant obstacles in the way of creating a crucial IDS that is capable of detecting irregular and unexpected attacks. Deep Learning procedures give different location methodologies for known and unseen attacks. A type of recurrent neural network known as Long Short-Term Memory (LSTM) can recall values over indefinite periods, making it an ideal system for predicting both known and unknown invasions. Using LSTM RNNs, we present a deep literacy method for creating an IDS in this paper. The model is trained with the help of the CSIC 2010 HTTP dataset. We exhibited that an LSTM model worked with the Adam enhancer can really fabricate an IDS twofold classifier with a delicacy pace of 0.9997.

2.4 The application of data visualization techniques for the detection of zero-day malware

Malicious software (malware) has expanded as the Internet of Things (IoT) has developed, featuring the significance of appropriate framework checking. With tremendous measures of information obtained from PC organizations, servers, and portable cell phones, viable logical methodologies are expected to match the degree and intricacy of such an information-brutal climate. Visualization methods may be of assistance to malware judges in the time-consuming process of analyzing suspicious conditions in today's Big Data environment. By designing a new visualization utilizing the similarity matrix system to directly establish malware brackets and by providing a comprehensive overview of visualization methods for detecting suspicious gist of systems, this paper aims to contribute to the evolving realm of information security visualization methods. Expanded x86 IA-32(opcode) similarity designs, which are challenging to distinguish with existing techniques, are utilized by our administration to attempt to recognize foggy malware. We foster mutt models that effectively join static and dynamic malware examination procedures with the perception of closeness frameworks to rapidly portray and recognize zero-day malware. As unmistakable malware families parade radically differed essence designs, the extraordinary delicacy of section accomplished with our recommended approach might be apparently perceived.

2.5 A detailed investigation and analysis of the application of machine learning techniques for intrusion detection

Identifying interruptions is a basic security issue in the present digital climate. Although machine proficiency methods have been generally utilized for interruption recognition, they may not be productive in relating a wide range of interruptions. Colorful machine literacy methods for identifying protrusive conditioning are thoroughly discussed and analyzed in this work. An attack bracket and a mapping of the attack's attributes are provided for each attack, as well as suggestions for using network attack datasets to improve the detection of low-frequency attacks. The limitations of each order, as well as the finding capacity of various machine literacy methods for various orders of assault, are contrasted and annotated.

Additionally, the article includes data mining tools for machine literacy that are visually appealing. At long last, further ways are proposed for culminating assault disclosure by machine education strategies.

CHAPTER 3: METHODOLOGY

3.1 Proposed system

The freely open malware datasets should be routinely refreshed and benchmarked because of the unique idea of malware and the continually moving assault procedures it utilizes. To foster a versatile and powerful intrusion detection system (IDS) that is equipped for identifying and characterizing surprising and unforeseen cyberattacks, a deep neural network (DNN), a kind of profound learning model, is the subject of examination in this review. It is important to examine an enormous number of datasets that have been made over the long run utilizing both static and dynamic techniques because of the consistent change in network conduct and the fast development of assaults. This sort of assessment upholds the unmistakable proof of the best computation for expecting future cyberattacks. A complete assessment of preliminaries with DNNs and other ordinary AI classifiers is introduced on different openly open benchmark malware datasets. The proper organization boundaries and organization geographies for DNNs are chosen to utilize the dataset and the accompanying hyperparameter determination strategies.

3.2 Model Architecture

The information and communications technology system of today is significantly more complicated, interconnected, and engaged in the generation of enormous amounts of data, known as "big data." The rapid distribution of a large number of apps and technological advancements are primarily to blame for this. The terms "big data" and "methods for extracting essential information from massive amounts of data" are used interchangeably. In the field of cyber security, granting access to big data technologies, particularly IDS, is crucial. The advancement of large information innovation has made it conceivable to remove different examples of lawful and malignant activities from enormous measures of organization and framework action information rapidly. This further develops IDS execution. Regardless, dealing with immense data using standard advancement is oftentimes unsafe. The goal of this part is to analyze the proposed framework's computational designing and complex methodologies, similar to message depiction systems, deep neural networks (DNNs), and the readiness strategies used in DNNs.

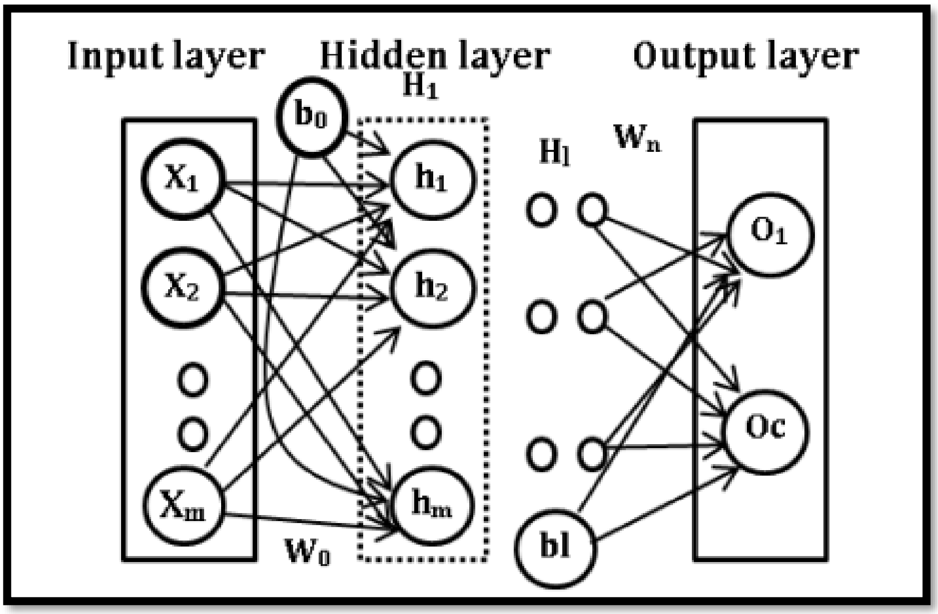


Fig 8: Our proposed model

In order to handle a large number of events at the host and network levels, the suggested scalable architecture makes use of other optimization techniques as well as distributed and parallel machine learning techniques. The versatile plan likewise utilizes the handling force of the general-purpose graphics processing unit (GPGPU) centers to examine occasions at the host and organization levels all the more rapidly and in equal. The structure contains two kinds of insightful motors: non-constant as well as continuous The coherent's engine will probably screen association and host-level events to make a caution by virtue of an attack. The created system can be increased to dissect progressively bigger measures of organization occasion information by adding extra PC assets. The made construction stands separated from various structures of its sort due to its flexibility and nonstop ID of dangerous activity using early rebuke signals.

3.2.1 Implementation Details

Utilizing IDS datasets like KDD and NSL, the creator of this paper assesses the viability of different old-style calculations, including SVM, Random Forest, and Naive Bayes, to recognize network assaults. Nonetheless, to distinguish dynamic assaults (when assailants present new goes after with changes in assault boundaries), these customary calculations should be prepared ahead of time.

I used a mix of KDD and NSL datasets, SVM, Random Forest, and DNN algorithms, and an 8-hidden-layer input for this article. In order to produce the most accurate model for predicting the testing class, the DNN algorithm continues to filter the training method using the hidden layer. In applications like data categorization and image processing, DNN is a well-known method with a high prediction ratio.

Below are the column names of dataset

duration,protocol_type,service,flag,src_bytes,dst_bytes,land,wrong_fragment,urgent,hot,num_failed_logins,logged_in,num_compromised,root_shell,su_attempted,num_root,num_file_creations,num_shells,num_access_files,num_outbound_cmds,is_host_login,is_guest_login,count,srv_count,serror_rate,srv_serror_rate,rerror_rate,srv_rerror_rate,same_srv_rate,diff_srv_rate,srv_diff_host_rate,dst_host_count,dst_host_srv_count,dst_host_same_srv_rate,dst_host_diff_srv_rate,dst_host_same_src_port_rate,dst_host_srv_diff_host_rate,dst_host_serror_rate,dst_host_srv_serror_rate,dst_host_rerror_rate,dst_host_srv_rerror_rate,label

The assaults are identified by their label in the aforementioned dataset columns, and the request signatures are identified by their bold, comma-separated names.

Below are the values of the above dataset columns?

0,tcp,ftp_data,SF,491,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,0,0,1,0,0,150,25,0.17,0.03,0.17,0,0,0,0.05,0,normal

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,166,9,1,1,0,0,0.05,0.06,0,255,9,0.04,0.05,0,0,1,1,0,0,Neptune

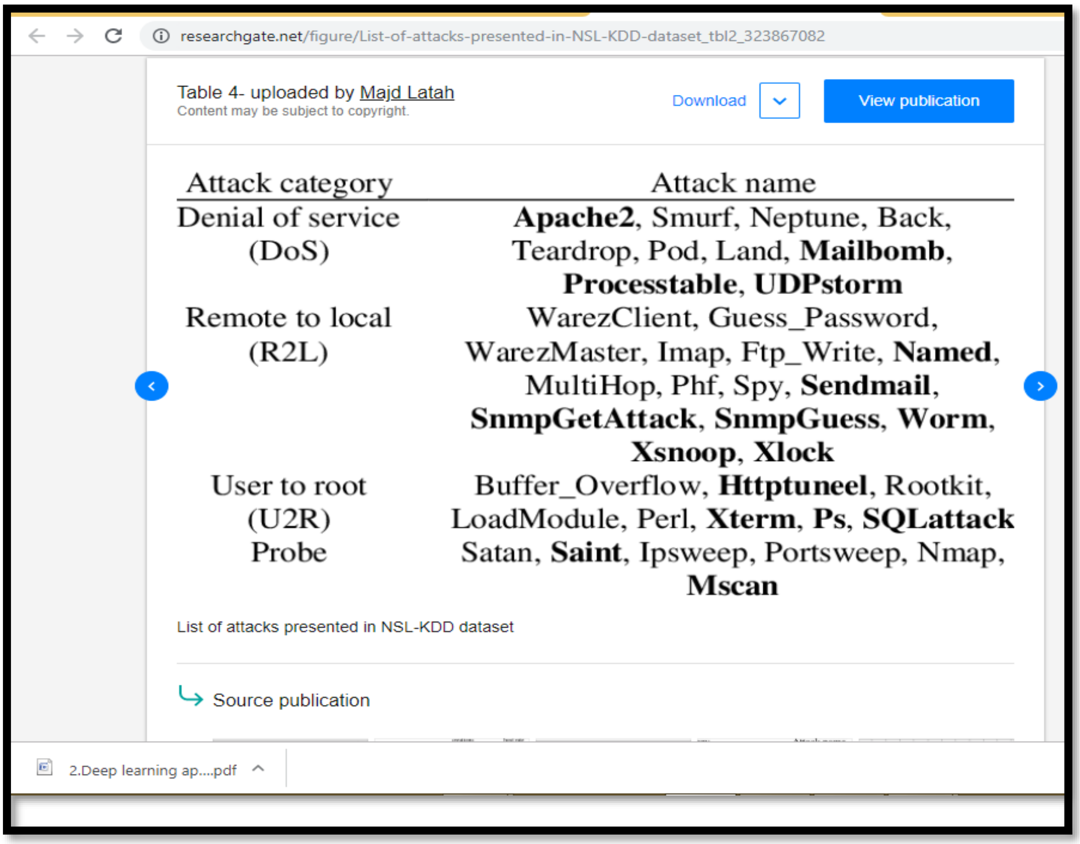
The initial two records are mark values, while the last one has a class assignment, like typical solicitation mark or assault signature. ' On the second record, the name "Neptune" is an assault. Similarly, the dataset contains approximately 30 distinct assault names.

A few factors in the previously mentioned dataset records are in string design, like tcp and ftp_data, and these qualities are excessive for expectation and will be eliminated utilizing the PREPROCESSING Idea. Since the algorithm will not recognize any attack names provided in string format, we must assign a numerical value to each attack. This will be all finished in PREPROCESS stages, and another document named 'clean.txt' will be made, which will be utilized to develop the preparation model.

In below line I am assigning numeric id to each attack

"normal":0,"neptune":1,"warezclient":2,"ipsweep":3,"portsweep":4,"teardrop":5,"nmap":6,"satan":7,"smurf":8,"pod":9,"back":10,"guess_passwd":11,"ftp_write":12,"multihop":13,"rootkit":14,"buffer_overflow":15,"imap":16,"warezmaster":17,"phf":18,"land":19,"loadmodule":20,"spy":21,"perl":22,"saint":23,"mscan":24,"apache2":25,"snmpgetattack":26,"processtable":27,"httptunnel":28,"ps":29,"snmpguess":30,"mailbomb":31,"named":32,"sendmail":33,"xterm":34,"worm":35,"xlock":36,"xsnoop":37,"sqlattack":38,"udpstorm":39

We can see that normal has id 0 and Neptune has id 1, and so on for all assaults, in the lines above. Normal, R2L, DOS, U2R, DOS, and Probe are all referred to by the author of the study; however, the dataset contains a variety of names that all belong to the same five categories: Ordinary, R2L, DOS, U2R, DOS, and Test. Screen captures are given beneath:



Attack category	Attack name
Denial of service (DoS)	**Apache2**, Smurf, Neptune, Back, Teardrop, Pod, Land, **Mailbomb**, **Processtable**, **UDPstorm**
Remote to local (R2L)	WarezClient, Guess_Password, WarezMaster, Imap, Ftp_Write, **Named**, MultiHop, Phf, Spy, **Sendmail**, **SnmpGetAttack**, **SnmpGuess**, **Worm**, **Xsnoop**, **Xlock**
User to root (U2R)	Buffer_Overflow, **Httptuneel**, Rootkit, LoadModule, Perl, **Xterm**, **Ps**, **SQLattack**
Probe	Satan, **Saint**, Ipsweep, Portsweep, Nmap, **Mscan**

Figure 9: Neptune Attack

We might derive from the screen captures over that Neptuneoattack has a place with the DOS classification. Essentially, different attacks fall into particular kinds.

3.3 Use case Diagram

A basic illustration of a customer's relationship with the frame that describes the particulars of an application case is called a usecase illustration. These usecase plates can be used to show visitors from any frame in a variety of ways, including as colorful feathers. Various plate feathers constantly link these plates, which are utilized in conjunction with textual usecases.

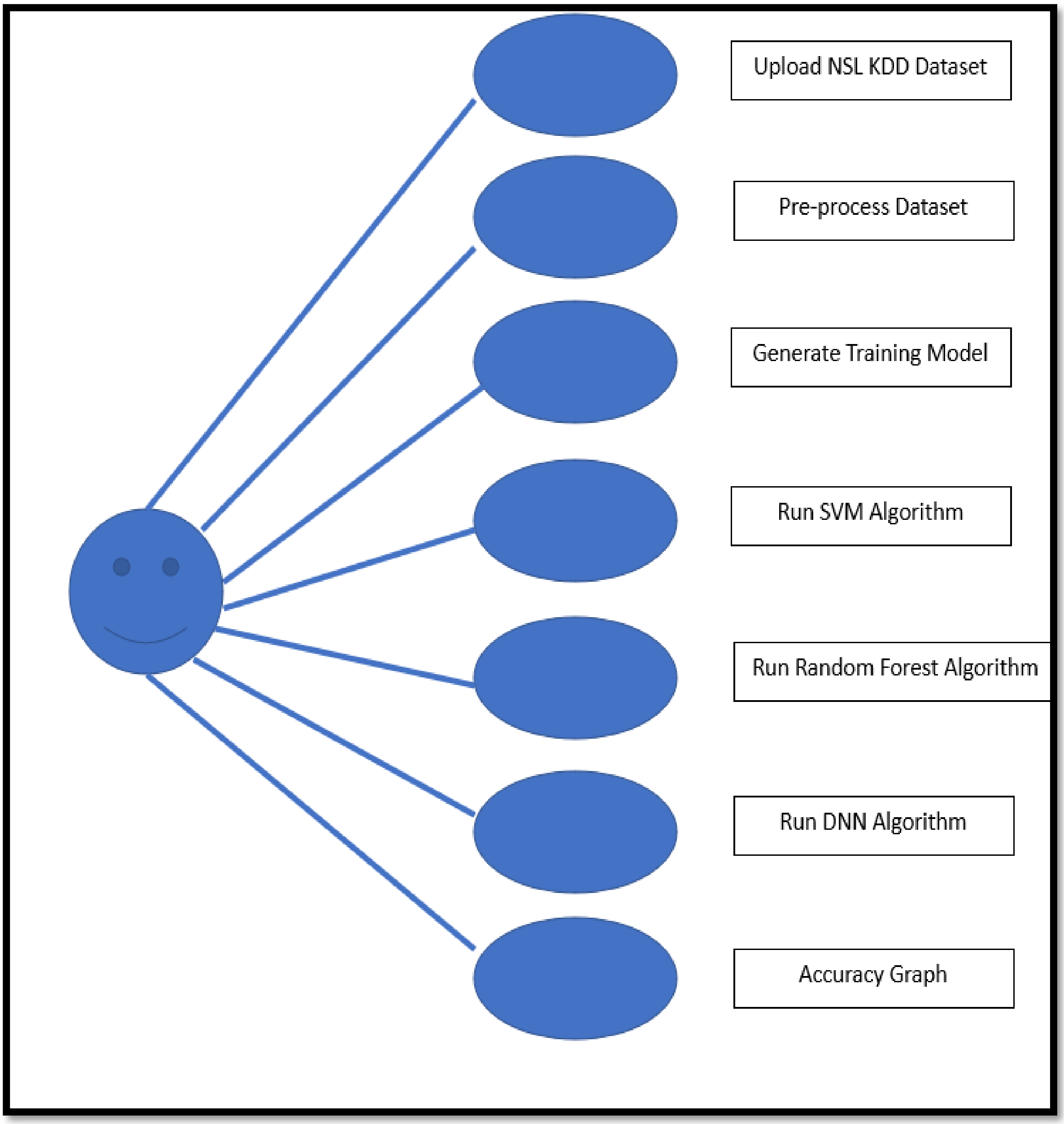


Figure 10: Use Case Diagram

3.3 Class Diagram

An essential part of object-oriented modeling is the class diagram. It is utilized for expansive calculated demonstrating of the application's framework as well as definite displaying and model transformation into PC code. Data modeling can also be done with class diagrams.

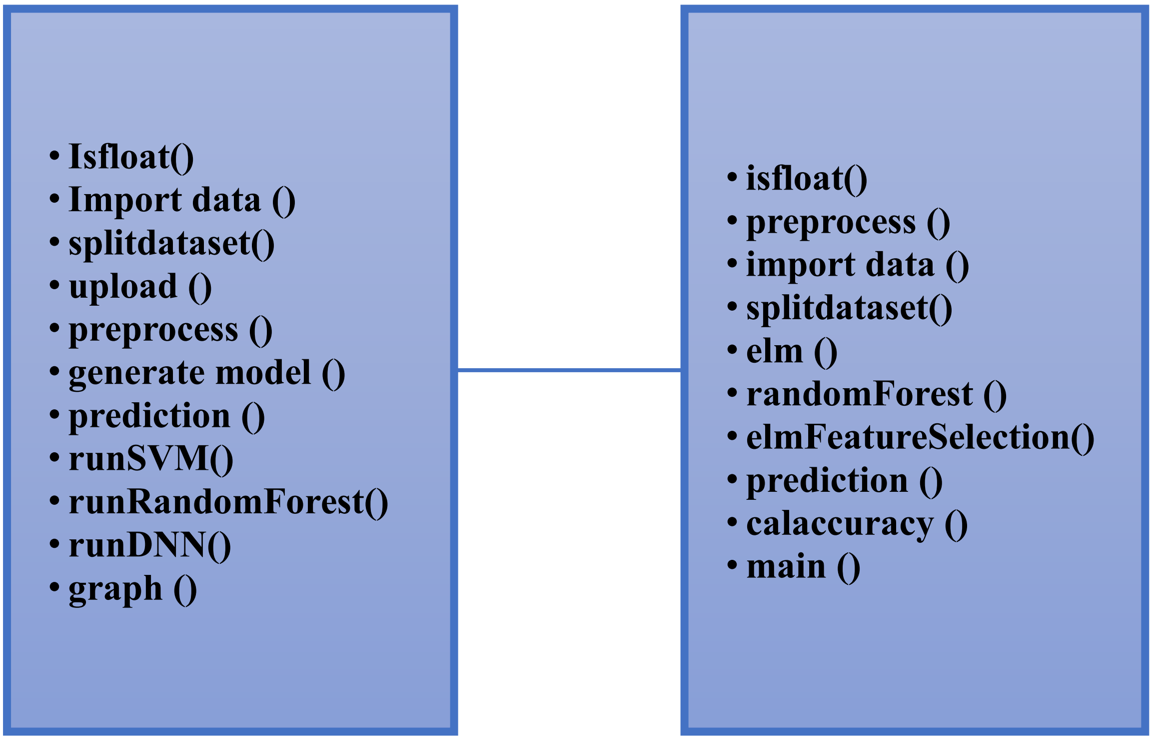


Figure 11: Class Diagram

3.4 Sequence Diagram

A type of commercial representation known as a sequence diagram depicts the interaction between processes and their order of occurrence. Expostulate partnerships during their allotted time are depicted in the sequence image. It characterizes the classes and articles related with the present circumstance, as well as the association of dispatches traded between the points of interest expected to carry out the circumstance's particular roles. These plates have to do with use cases that are allowed in the Logical View of the structure being worked on. Timing plates, event plates, and event scripts are all other names for these plates.

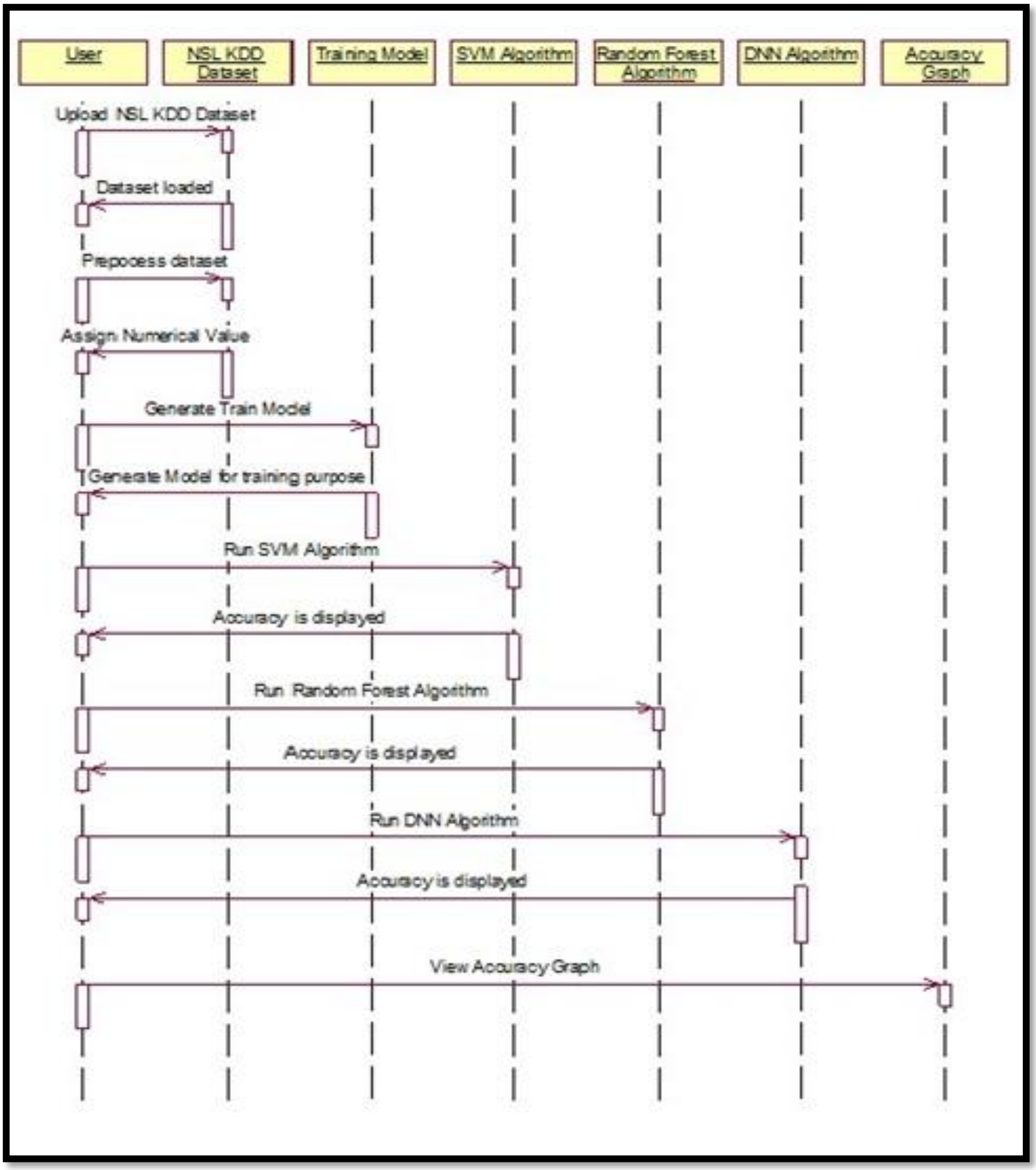


Figure 12: Sequence Diagram

3.5 Activity Diagram

A work representation is a key realistic utilized in Unified Modelling Language (UML) to portray dynamic framework highlights. It is typically a flowchart picture to address the contribution of molding. The edge's activity is viewed as a work. Subsequently, control inflow is removed from tasks. This example may be consecutive, simultaneous, or fanned.

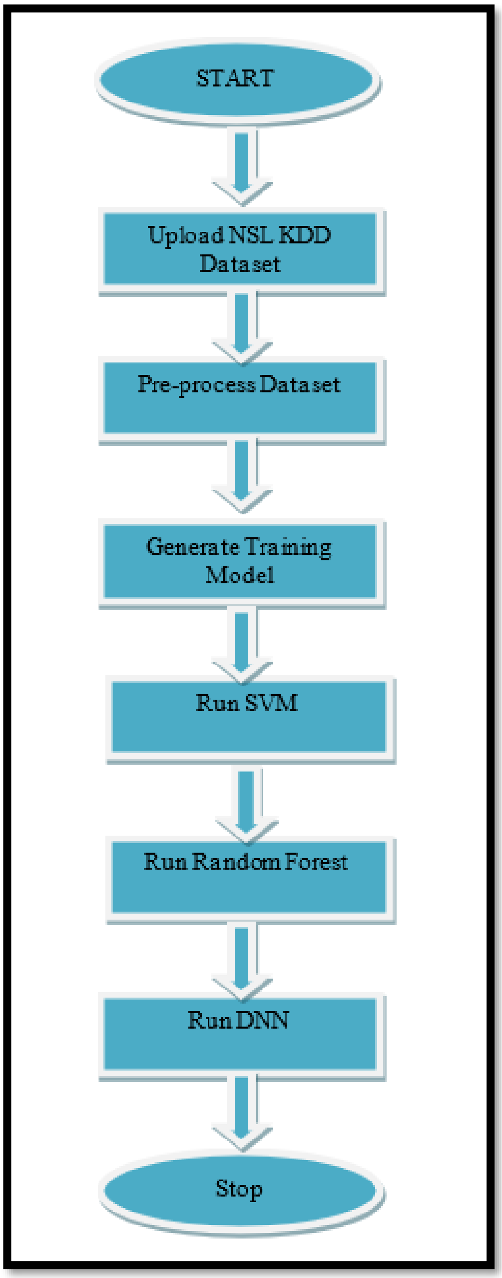


Figure 13: Activity Diagram

3.6 Introduction to Testing

The justification behind analyzing search out track down botches. Testing is an extreme winning propensity fo r attempting to find each logical shortcoming or fragility in a work article. In the hope that the Product foundation will be accepted in a suitable manner and satisfy customer requirements, it provides a plan for honestly attractive a study of the utility of parts, alternative groups, and gatherings in addition to a completed article. It is best to record a comprehensive set of attempts. There are many different kinds of tests. The essential examination is at the focal point of each kind of test.

3.7 Types of Testing

3.7.1 System Testing

Particularly in the field of data innovation, testing is a fundamental piece of any framework or undertaking. It is fundamental to decide if a system or adventure is good to go and can go against the obstructions of a particular environment. Therefore, testing is fundamental preceding turn of events. To guarantee the product's reliability, various tests can be done. Intelligent testing and the program's rehashed design execution are utilized to totally test the product. The outcomes are additionally affirmed after the code is entirely checked for all conceivable precise information.

3.7.2 Module Testing

Disappointments are found autonomously for every module. This licenses us to find and review botches without affecting various modules. Right when a program fails to complete a necessary job, giving the ideal result ought to be redressed. As needs be, each module is uninhibitedly examined from the base up, beginning with the tiniest and most decreased modules and progressing to a more significant level. Freely, every framework module is tried. The examination exhibits that the proposed technique performs better compared to the current framework. Each module of the structure is attempted uninhibitedly. This framework assesses every module of asset arrangement and work planning independently, subsequently shortening the cycle holding up time.

3.2.3 Integration Testing

To guarantee that any shortcoming that is found can be fixed without influencing different modules, every module is assessed independently. To deliver the expected results, the program should be improved assuming it neglects to carry out the necessary role. Beginning with the littlest and most minimal modules and continuing on toward a higher level, every module is tried freely. Every part of the framework is exposed to free testing to ensure its productive activity. To eliminate how much time spent sitting tight for an interaction, this framework tests every module of the asset order and undertaking planning frameworks independently and concocts the right outcomes. The aftereffects of the proposed framework and those of the ongoing framework show that the proposed framework performs better compared to the ongoing framework.

3.2.4 Acceptance Testing

When the client affirms that there are no critical exactness issues, the framework goes through a last acknowledgment test. This test ensures that the system fulfills the principal goals, targets, and necessities set during the assessment stage, without requiring certifiable execution, hence preventing time and money waste. At the point when the framework passes acknowledgment testing and lives up to clients' and the executives' assumptions, it is viewed as OK and prepared for use.

3.2.5 Functional Testing

Functional tests provide well-behaved evidence that facilities are approachable and reliable in accordance with specific requirements, foundation proof, and customer manuals.

The following items are the focus of functional experimentation:

Significant Data : Known categories of reliable data allow for the possibility to be unquestioned.

Invalid Data : Recognized categories of incorrect data allow for the elimination of possibility.

Capacities : Famous resources allow the possibility to be resolved.

Yield : Recognized classes with valuable yields must be resolved.

Strategies and frameworks: communicating orders or foundations must be notified. Experienced tests are focused on needs, essential capabilities, or amazing experiments. Additionally, well-organized additions that are connected to create business process streams; realities fields, predefined cycles, and developing stages surrender probability thought-out for test. Prior to working analysis is finished, additional tests are recognized and the viable worth of current shudder in precious stone.

CHAPTER 4: LITERATURE REVIEW

4.1 Introduction

4.1.1 NIST special publication on intrusion detection systems

IDSs are programming in any case equipment gadgets that robotize cycle of checking and dissecting PC in any case network movement for indications of safety issues. Due towards expansion in recurrence and seriousness of organization assaults over beyond couple of years, interruption recognition frameworks are currently an essential piece of safety foundation for greater part of organizations. For the individuals who need towards understand what security objectives these components support, how towards pick and design interruption discovery frameworks for their specific framework and organization conditions, how towards oversee yield about interruption location frameworks, and how towards incorporate interruption identification capabilities among different parts of hierarchical security foundation, this guide was composed as a presentation towards interruption recognition. References that give peruser concentrated in any case top to bottom data on unambiguous interruption location issues references towards extra data sources are likewise advertised.

4.1.2 Intrusion detection: A survey

The outlook for network security has changed due to increasing spread about computer networks. A state that makes information easily accessible makes computer networks susceptible towards various hacker attacks. There are numerous & potentially catastrophic threats towards networks. Researchers have so far created intrusion detection systems (IDS) that can recognize attacks in a variety about situations. There are countless techniques that can be used towards identify misuse & anomalies. Since different types about ecosystems are best served through different approaches, many about technologies presented are complementary towards one another. In order towards survey & categorise intrusion detection systems, this study proposes a taxonomy for doing so. detection theory & a few operational components about intrusion detection make up taxonomy.

4.1.3 Survey on Intrusion Detection System using Machine Learning Techniques

In the current world, nearly everybody approaches a PC, and organization-based innovation is quickly developing. Network security has developed to be a significant, if not irreplaceable, part of PC frameworks. The objective of an interruption recognition framework (IDS) is towards perceive framework assaults and separate normal utilization designs from unusual ones. AI procedures have further developed interruption recognition frameworks and different frameworks utilized to distinguish interruptions. This study surveys an assortment of machine procedures for interruption discovery frameworks. This study's framework engineering for an interruption recognition framework is additionally portrayed, with the point of lessening deception rates and further developing interruption discovery exactness.

4.1.4 Feature Selection for Intrusion Detection System Using Ant Colony Optimization

Intrusion detection is a key examination point in network security. As a result of the non-linear nature of interruption endeavors, unusual organization traffic conduct, and numerous factors in issue space, interruption identification frameworks are a troublesome area of exploration. Choosing productive and urgent parts for interruption recognition is an exceptionally fundamental subject in data security. The point of this study is to recognize key components for making an interruption identification framework that is both viable and computationally effective. This study suggests an interruption recognition framework whose highlights are painstakingly chosen utilizing insect province advancement towards further development execution. The recommended technique is simple to use and has little handling intricacy since it involves a little arrangement of highlights for order. As exhibited by significant test discoveries on KDD Cup 99 and NSL-KDD interruption location benchmark informational collections, this new technique beats prior draws near, giving more noteworthy exactness in recognizing interruption endeavors and diminishing misleading problems among less elements.

4.1.5 Design about experiments application, concepts, examples: State about art

Application areas for statistical tools known as Design about Experiments (DOE) include system, process, and product design, development, and optimization. It is a flexible tool that may be used in a wide range of situations, such as design for comparisons, variable screening, transfer function discovery, optimization, and resilient design. state-of-the-art DOE use is discussed, along with the evolution of DOE over time. Researchers are also given instructions on how to design, organize, and conduct experiments, as well as how to assess and interpret results using examples. This article also shows how, over the past 20 years, DOE applications have been rapidly growing in both manufacturing and non-manufacturing industries. About half of its applications are in sciences, specifically in the domains of computer science, engineering, biochemistry, and medicine.

4.2 Deep Learning Approaches

Networks in deep learning, a subset of AI (ML) in the field of man-made brainpower (man-made intelligence), can gain from marked and unlabeled information in both directed and unaided techniques. Deep brain networks in any case deep brain learning are different names for deep learning. Deep Learning is an element of artificial intelligence that recreates the how human mind functions as far as how it processes information and makes designs that might be applied towards decision production [9]. Although there is no single meaning of deep learning, the greater part of the details stress the following characteristics:

- Branch about machine learning. Usually nonlinear models.
- Fits models to data using both supervised & unsupervised methods.
- Models are multi-layered graph structures (networks) (deep).

The majority of research in this area & implemented algorithm in subject about intrusion detection can be broadly divided into three primary groups, which are [10] (Figure 2 provides an illustration of deep learning approaches' classification.)

- Imaginative (unsupervised).
- Inequitable (supervised).
- A deep hybrid architecture.

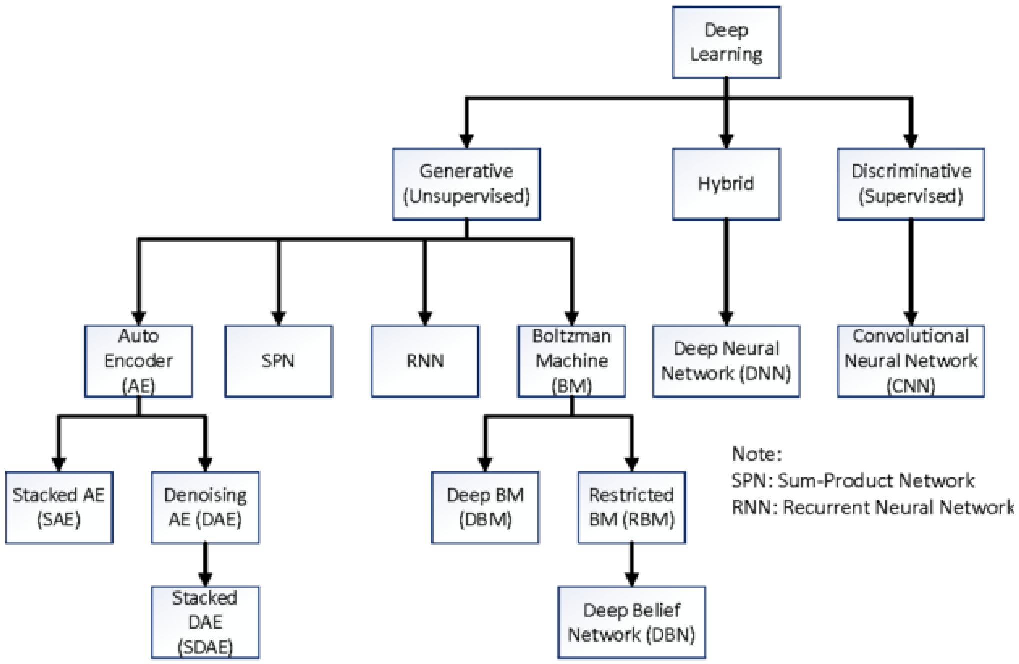


Figure 14: Taxonomy About Deep Learning Methods

4.3 Popular Intrusion Detection Datasets for Deep Learning

For their own examination as well as towards add towards local area vaults, many exploration groups today gather a scope of information sorts. most well-known interruption discovery datasets utilized in DL research are made sense of here.

MIT Lincoln Lab has assembled and dispersed first standard information for assessing PC network IDS under help from "Guard Progressed Exploration Ventures Office" (DARPA) and "Flying corps Exploration Lab" (AFRL). Scientists should remove properties from documents all together towards use them in ML calculations since DARPA informational index is comprised of generally crude records.

The KDDCup 1999 information assortment was used in DARPA IDS assessment program.

information comprises of a packed 4 GB tcp dump produced over course of close to seven weeks of organization movement. Every association record, which is around 100 bytes in length, can oblige 5 million associations. There are around 4,900,000 single association vectors in it, and everyone has 41 properties.

The KDDCup 1999 informational collection and NSLKDD informational collection are primarily very comparative (at the end of the day it has 22 examples about assaults in any case ordinary rush hour gridlock, and fields for 41 properties). Figure 3 shows an overall portrayal of these interconnected informational indexes. (NSLKDD, KDD-99, and DARPA). DARPA is a major crude informational collection. size-diminished and duplications eliminated NSLKDD information assortment relates towards KDD-99.

Figure 15: Correlation between main & extracted datasets

4.4. Frameworks for Deep Learning Implementation

Deep learning architecture combines the implementation of modularized deep learning algorithms among approaches for optimization, dissemination, & infrastructure support. most popular frameworks for implementing deep learning algorithms are briefly introduced in this section.

4.4.1 Tensor-flow

Google has been utilizing Tensor-Flow (TF), the distributed method for training NNs that replaces Dist-Belief, since 2011. Google brain team developed TF, an open-source library for numerical computation. TF operates more quickly since its Python API was used throughout development rather than a C/C++ engine. TF supports CUDA. TensorFlow can be used to create almost any form of network, despite fact that deep networks cannot be configured among hyper-parameters. Additionally, Tensor-Flow includes a C++ interface.

4.4.2 Theano

The ML group about Montreal University created Theano. It is an open-source, cross-platform Python library. multidimensional array mathematical statement is defined, optimised, & evaluated using Theano Python module. High network modelling capability, dynamic code generation, & speed among a variety about GPU support are all provided through Theano. However, Theano offers low-level API & has numerous intricate compilations that are typically time-consuming. Theano, on other hand, has a variety about instructional materials & is still used through a sizable number about academics & developers.

4.4.3 Keras

For implementing deep learning in Python-based Theano & TF, Keras has been created. It enables high-level NN API for swift deep learning algorithm implementation. main selling point about Keras is that it works among Theano & TF, two commonly used deep learning implementation frameworks, & that it can be extended, modularized, & utilised on a user platform using Python. Theano & TF's design makes it easy towards create high-level libraries like Keras that could be used among any about backends. In general, TF & Theano programmes are larger than Keras-equivalent programmes. Keras model is depicted in Figure 4.

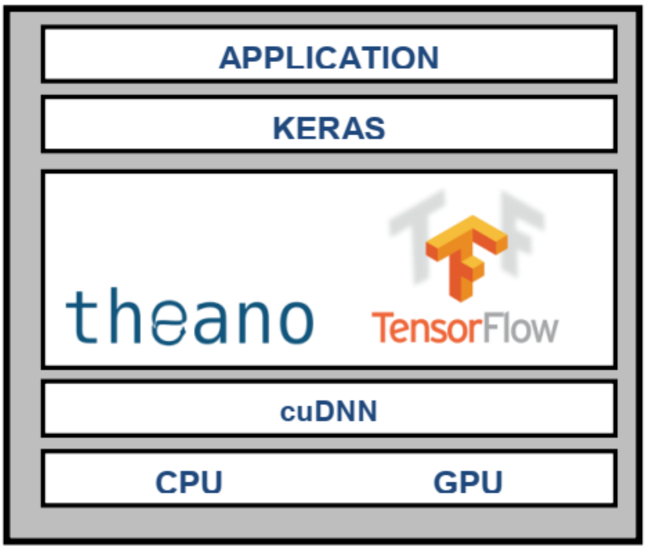


Figure 16: Architecture about Keras

4.4.4 Torch/PyTorch

In view of simple to-utilize, speedy to-learn, and versatile Lua programming language, Light is an open source deep learning system. This structure is a broad ally about ML strategies and is intended for logical calculation. In deep learning structure local area, Py-Light has as of late seen a significant level about prominence and is seen as Tensor-rival. Stream's Py-Light is essentially a port about Light system, which is utilized towards fabricate deep brain organizations and do tensor estimations that are incredibly complicated. A front-end Light incorporation for suitable execution deep learning improvement among critical GPU support has as of late been made at Facebook and is called Py-Light. It ensures a Python front-end that makes it conceivable towards construct dynamic NN. On opposite side, tool compartment has as of late been made accessible, and there isn't any local area support, informative materials, in any case assessment about its adequacy.

4.5 CONCLUSION

The overview about deep learning & points that majority about definitions emphasis are provided in this essay. We examined most recent articles on deep learning for intrusion detection. We look at a few popular deep learning architectures & highlight some about its applications towards intrusion detection. More precisely, Generative (unsupervised), Discriminative (supervised), & Hybrid deep architecture classes about deep learning architectures are covered in detail along among their methodologies. These three classes offer a great deal about versatility & have been effective & trustworthy in a variety about challenges for decades. We have examined associated works for each about aforementioned classes & techniques that are used in intrusion detection sector. most well-liked deep learning implementation frameworks and intrusion detection datasets are highlighted in section that follows. Although supervised learning algorithms operate with labelled data, they struggle towards perform well when dealing with large amounts of data since it is difficult towards collect labelled data. In order towards process unlabeled data, unsupervised learning methods are employed. We can also utilise unsupervised learning algorithms towards forecast best results if we are unsure about output data (outputs). Sets about intrusion detection data are crucial for system testing & training. Every dataset has a great number about features, majority about which are superfluous otherwise unimportant. Deep learning techniques are best suited for simplifying complex features otherwise extracting features. If we are unsure about relationship between targeted classification output & raw input data, we may employ deep learning techniques. In conclusion, it can be claimed that majority about strategies presented have demonstrated ability towards achieve high accuracy levels in a more automatic manner.

CHAPTER 5: IMPLEMNETATION

IDS.py:

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
from tkinter.filedialog import askopenfilename
import numpy as np
import pandas as pd
from sklearn import *
from sklearn.metrics import confusion_matrix

from sklearn.model_selection import
train_test_split from sklearn import svm
from sklearn.metrics import  accuracy_score
from sklearn.metrics import
classification_report
from sklearn.ensemble import RandomForestClassifier

from sklearn_extensions.extreme_learning_machines.elm import GenELMClassifier

from sklearn_extensions.extreme_learning_machines.random_layer import RBFRandomLayer,
MLPRandomLayer

from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import Lasso

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2


main = tkinter.Tk()
main.title("Deep Learning")
main.geometry("1300x1200")


global filename
global labels
global columns
global balance_data
```

```
global data
global X, Y, X_train, X_test, y_train, y_test
global svm_acc, random_acc, dnn_acc


def isfloat(value):
  try:
    float(value)
    return True
  except ValueError:
    return False



def importdata():
    global
    balance_data
    balance_data = pd.read_csv("clean.txt")
    return balance_data



def splitdataset(balance_data):

    X = balance_data.values[:,
    0:37] Y =
    balance_data.values[:, 38]
    X_train, X_test, y_train, y_test =
    train_test_split( X, Y, test_size = 0.2,
    random_state = 0)
    return X, Y, X_train, X_test, y_train, y_test


def upload():
    global
    filename
    text.delete('1.0', END)

    filename = askopenfilename(initialdir =
    "dataset") pathlabel.config(text=filename)
    text.insert(END,"Dataset loaded\n\n")


def preprocess():
    global labels
    global columns
    global
    filename
```

```
text.delete('1.0', END)
columns =
["duration","protocol_type","service","flag","src_bytes","dst_bytes","land","wrong_fragment","u
rgent","hot","num_failed_logins","logged_in","num_compromised","root_shell","su_attempted",
"num_root","num_file_creations","num_shells","num_access_files","num_outbound_cmds","i
s_host_login","is_guest_login","count","srv_count","serror_rate","srv_serror_rate","rerror_rat
e","s
rv_rerror_rate","same_srv_rate","diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_h
ost
_srv_count","dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate","
dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate","dst_host_rerror
_rate","dst_host_srv_rerror_rate","label"]


labels =
{"normal":0,"neptune":1,"warezclient":2,"ipsweep":3,"portsweep":4,"teardrop":5,"nmap":6,"sata
n":7,"smurf":8,"pod":9,"back":10,"guess_passwd":11,"ftp_write":12,"multihop":13,"rootkit":14,
"buffer_overflow":15,"imap":16,"warezmaster":17,"phf":18,"land":19,"loadmodule":20,"spy":21
,"perl":22,"saint":23,"mscan":24,"apache2":25,"snmpgetattack":26,"processtable":27,"httptun
nel
":28,"ps":29,"snmpguess":30,"mailbomb":31,"named":32,"sendmail":33,"xterm":34,"worm":
35, "xlock":36,"xsnoop":37,"sqlattack":38,"udpstorm":39}

balance_data = pd.read_csv(filename)
dataset = ''
index = 0
cols = ''
for index, row in balance_data.iterrows():
  for i in range(0,42):
    if(isfloat(row[i])):
      dataset+=str(row[i])+',
      ' if index == 0:
      cols+=columns[i]+','
  dataset+=str(labels.get(row[41])
  ) if index == 0:
    cols+='Label'
  dataset+='\n'
  index = 1;
f = open("clean.txt", "w")
f.write(cols+"\n"+dataset
) f.close()


text.insert(END,"Removed non numeric characters from dataset and saved inside clean.txt
file\n\n")

text.insert(END,"Dataset Information\n\n")
text.insert(END,dataset+"\n\n")
```

```
def generateModel():
    global data
    global X, Y, X_train, X_test, y_train, y_test


    data = importdata()

    X, Y, X_train, X_test, y_train, y_test =
    splitdataset(data) text.delete('1.0', END)
    text.insert(END,"Training model generated\n\n")


def prediction(X_test, cls):
    y_pred = cls.predict(X_test)
    for i in range(len(X_test)):
      print("X=%s, Predicted=%s" % (X_test[i], y_pred[i]))
    return y_pred
# Function to calculate accuracy
def cal_accuracy(y_test, y_pred, details):
    cm = confusion_matrix(y_test, y_pred)
    accuracy = accuracy_score(y_test,y_pred)*100
    text.insert(END,details+"\n\n")
    text.insert(END,"Accuracy : "+str(accuracy)+"\n\n")
    text.insert(END,"Report : "+str(classification_report(y_test, y_pred))+"\n")
    text.insert(END,"Confusion Matrix : "+str(cm)+"\n\n\n\n")
    return accuracy

def runSVM():
    global
    svm_acc
    global X, Y, X_train, X_test, y_train, y_test
    text.delete('1.0', END)
    cls = svm.SVC(C=2.0,gamma='scale',kernel = 'rbf', random_state =
    2) cls.fit(X_train, y_train)
    text.insert(END,"Prediction Results\n\n")
    prediction_data = prediction(X_test, cls)
    svm_acc = cal_accuracy(y_test, prediction_data,'SVM Accuracy, Classification Report &
Confusion Matrix')

def runRandomForest():
    global random_acc
    global X, Y, X_train, X_test, y_train, y_test

    text.delete('1.0', END)
```

```
    cls = RandomForestClassifier(n_estimators=1,max_depth=0.9,random_state=None)
    cls.fit(X_train, y_train)
    text.insert(END,"Prediction Results\n\n")
    prediction_data = prediction(X_test,  cls)
    random_acc = cal_accuracy(y_test, prediction_data,'Random Forest Algorithm Accuracy,
Classification Report & Confusion Matrix')

def runDNN():
    global dnn_acc
    global X, Y, X_train, X_test, y_train, y_test
    text.delete('1.0', END)
    srhl_tanh = MLPRandomLayer(n_hidden=8,
    activation_func='tanh') cls =
    GenELMClassifier(hidden_layer=srhl_tanh)
    cls.fit(X_train, y_train)
    text.insert(END,"Prediction Results\n\n")
    prediction_data = prediction(X_test, cls)
    dnn_acc = cal_accuracy(y_test, prediction_data,'DNN Algorithm Accuracy, Classification
Report & Confusion Matrix')


def graph():

    height = [svm_acc,random_acc,dnn_acc]

    bars = ('SVM Accuracy', 'Random Forest Accuracy','DNN
    Accuracy') y_pos = np.arange(len(bars))
    plt.bar(y_pos, height)
    plt.xticks(y_pos, bars)
    plt.show()

font = ('times', 16, 'bold')

title = Label(main, text='Deep Learning Approach for Intelligent Intrusion Detection System')
title.config(bg='brown', fg='white')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)

font1 = ('times', 14, 'bold')

upload = Button(main, text="Upload NSL KDD Dataset", command=upload)
upload.place(x=50,y=100)
upload.config(font=font1)


pathlabel = Label(main)
pathlabel.config(bg='brown', fg='white')
```

```
pathlabel.config(font=font1)
pathlabel.place(x=300,y=100)

preprocess = Button(main, text="Preprocess Dataset", command=preprocess)
preprocess.place(x=50,y=150)
preprocess.config(font=font1)

model = Button(main, text="Generate Training Model", command=generateModel)
model.place(x=330,y=150)
model.config(font=font1)
runsvm = Button(main, text="Run SVM Algorithm", command=runSVM)
runsvm.place(x=610,y=150)
runsvm.config(font=font1)

runrandomforest = Button(main, text="Run Random Forest Algorithm",
command=runRandomForest)

runrandomforest.place(x=870,y=150)
runrandomforest.config(font=font1)

runeml = Button(main, text="Run DNN Algorithm", command=runDNN)
runeml.place(x=50,y=200)
runeml.config(font=font1)

graph = Button(main, text="Accuracy Graph", command=graph)
graph.place(x=330,y=200)
graph.config(font=font1)


font1 = ('times', 12, 'bold')
text=Text(main,height=30,width=150)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=10,y=250)
text.config(font=font1)
main.config(bg='brown') main.mainloop()
```

Test.py:

```
import numpy as np
import pandas as pd
from sklearn import *
from sklearn.metrics import confusion_matrix

from sklearn.model_selection import
train_test_split from sklearn import svm
from sklearn.metrics import  accuracy_score
from sklearn.metrics import
```

```
classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn_extensions.extreme_learning_machines.elm import GenELMClassifier

from sklearn_extensions.extreme_learning_machines.random_layer import RBFRandomLayer,
MLPRandomLayer

from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import Lasso
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

global labels
global columns
global balance_data
def isfloat(value):

  try:

    float(value)
    return True
  except ValueError:

    return False


def preprocess():
    global labels
    global columns
    columns =
["duration","protocol_type","service","flag","src_bytes","dst_bytes","land","wrong_fragment","u
rgent","hot","num_failed_logins","logged_in","num_compromised","root_shell","su_attempted",
"num_root","num_file_creations","num_shells","num_access_files","num_outbound_cmds","is_
host_login","is_guest_login","count","srv_count","serror_rate","srv_serror_rate","rerror_rate","s
rv_rerror_rate","same_srv_rate","diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host
_srv_count","dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate","
dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate","dst_host_rerror
_rate","dst_host_srv_rerror_rate","label"]

    labels =
{"normal":0,"neptune":1,"warezclient":2,"ipsweep":3,"portsweep":4,"teardrop":5,"nmap":6,"sata
n":7,"smurf":8,"pod":9,"back":10,"guess_passwd":11,"ftp_write":12,"multihop":13,"rootkit":14,
"buffer_overflow":15,"imap":16,"warezmaster":17,"phf":18,"land":19,"loadmodule":20,"spy":21
,"perl":22,"saint":23,"mscan":24,"apache2":25,"snmpgetattack":26,"processtable":27,"httptun
nel
":28,"ps":29,"snmpguess":30,"mailbomb":31,"named":32,"sendmail":33,"xterm":34,"worm":
35, "xlock":36,"xsnoop":37,"sqlattack":38,"udpstorm":39}
```

```
    balance_data                                =
    pd.read_csv("dataset.txt") dataset = ''

    index = 0
    cols = ''
    for index, row in balance_data.iterrows():
      for i in range(0,42):
        if(isfloat(row[i])):
          dataset+=str(row[i])+',
          ' if index == 0:
          cols+=columns[i]+','
      dataset+=str(labels.get(row[41])
      ) if index == 0:
        cols+='Label'
      dataset+='\n'
      index = 1;

    f = open("clean.txt", "w")
    f.write(cols+"\n"+dataset
    ) f.close()

def importdata():
    global
    balance_data
    balance_data =
    pd.read_csv("clean.txt") # Printing the
    dataswet shape
    print ("Dataset Lenght: ", len(balance_data))

    print ("Dataset Shape: ", balance_data.shape)
    # Printing the dataset obseravtions
    print ("Dataset:
    ",balance_data.head()) return
    balance_data


def splitdataset(balance_data):

    X = balance_data.values[:,
    0:37] Y =
    balance_data.values[:, 38]
    # Spliting the dataset into train and test

    X_train, X_test, y_train, y_test =
    train_test_split( X, Y, test_size = 0.2,
    random_state = 0)
    return X, Y, X_train, X_test, y_train,  y_test
```

```
# Function to perform training with giniIndex.
def train_using_gini(X_train, X_test,  y_train):

        # Creating the classifier object

        clf_gini = svm.SVC(C=2.0,gamma='scale',kernel = 'rbf', random_state = 2)

        # Performing training
        clf_gini.fit(X_train, y_train)
        return clf_gini


def elm(X_train, X_test, y_train):

  srhl_tanh = MLPRandomLayer(n_hidden=8, activation_func='tanh')

  cls =
  GenELMClassifier(hidden_layer=srhl_tanh)
  cls.fit(X_train, y_train)
  return cls


def randomForest(X_train, X_test, y_train):

  cls = RandomForestClassifier(n_estimators=1,max_depth=0.9,random_state=None)
  cls.fit(X_train, y_train)
  return cls


def elmFeatureSelection(X_train, X_test, y_train):

  srhl_tanh = MLPRandomLayer(n_hidden=15,
  activation_func='tanh') cls =
  GenELMClassifier(hidden_layer=srhl_tanh)
  print('Original features:',
  X_train.shape[1]) total =
  X_train.shape[1];
  X_train = SelectKBest(chi2, k=1).fit_transform(X_train, y_train)

  print('features set reduce after applying features concept:', (total - X_train.shape[1]))
  cls.fit(X_train, y_train)
  return cls


# Function to make predictions
def prediction(X_test,
clf_object):
```

```
    # Predicton on test with giniIndex
    y_pred = clf_object.predict(X_test)
    print("Predicted values:")
    print(y_pred)

    #for i in range(len(X_test)):

      #print("X=%s, Predicted=%s" % (X_test[i],
    y_pred[i])) return y_pred


# Function to calculate accuracy
def cal_accuracy(y_test,
y_pred):
    print("Confusion Matrix: ", confusion_matrix(y_test, y_pred))
    print ("Accuracy : ", accuracy_score(y_test,y_pred)*100)
    print("Report : ", classification_report(y_test, y_pred))

def main():
    #preprocess()
    data = importdata()

    X, Y, X_train, X_test, y_train, y_test =
    splitdataset(data) clf_gini = train_using_gini(X_train,
    X_test, y_train)


    #print("Results Using Gini Index:")
    y_pred_gini = prediction(X_test,
    clf_gini) cal_accuracy(y_test,
    y_pred_gini) clf_gini = elm(X_train,
    X_test, y_train)


    #print("Results Using Gini Index:")
    y_pred_gini = prediction(X_test, clf_gini)
    cal_accuracy(y_test, y_pred_gini)


    clf_gini = randomForest(X_train, X_test,
    y_train) #print("Results Using Gini Index:")
    y_pred_gini = prediction(X_test, clf_gini)
    cal_accuracy(y_test, y_pred_gini)


    clf_gini = elmFeatureSelection(X_train, X_test,
    y_train) #print("Results Using Gini Index:")

    X_test = SelectKBest(chi2, k=1).fit_transform(X_test,y_test)
```

```
        y_pred_gini = prediction(X_test, clf_gini)
        cal_accuracy(y_test, y_pred_gini)

    # Calling main function

if name_=="_main_": main()
```

CHAPTER 6: SCREENSHOTS

6.1 Basic Exploration of dataset if given below in figure

```python
# Read Train and Test dataset
data_train = pd.read_csv("../input/nslkdd/KDDTrain+.txt")
```

```python
# Check data
data_train.head()
```

	0	tcp	ftp_data	SF	491	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.10	0.11	0.12	0.13	0.14	0.15	0.16	0.17	2	2.1	0.00	0.00.1	0.00.2	0.00.3	1.00	0.00.4	0.00.5	150	25	0.17.1	0.03	0.17.2	0.00.
0	0	udp	other	SF	146	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	1	0.0	0.0	0.0	0.0	0.08	0.15	0.00	255	1	0.00	0.60	0.88	0.0
1	0	tcp	private	S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	123	6	1.0	1.0	0.0	0.0	0.05	0.07	0.00	255	26	0.10	0.05	0.00	0.0
2	0	tcp	http	SF	232	8153	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	5	5	0.2	0.2	0.0	0.0	1.00	0.00	0.00	30	255	1.00	0.00	0.03	0.0
3	0	tcp	http	SF	199	420	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	30	32	0.0	0.0	0.0	0.0	1.00	0.00	0.09	255	255	1.00	0.00	0.00	0.0
4	0	tcp	private	REJ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	121	19	0.0	0.0	1.0	1.0	0.16	0.06	0.00	255	19	0.07	0.07	0.00	0.0

Figure 17 : Dataset

6.2 Train Dataset relevant Outputs

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	root_shell	su_attempted	num_ro
count	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.000000	125972.0000
mean	287.146929	45567.100824	19779.271433	0.000198	0.022688	0.000111	0.204411	0.001222	0.395739	0.279253	0.001342	0.001103	0.3021
std	2604.525522	5870354.480801	4021285.112114	0.014086	0.253531	0.014366	2.149977	0.045239	0.489011	23.942137	0.036603	0.045155	24.3997
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
50%	0.000000	44.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
75%	0.000000	276.000000	516.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.0000
max	42908.000000	1379963888.000000	1309937401.000000	1.000000	3.000000	3.000000	77.000000	5.000000	1.000000	7479.000000	1.000000	2.000000	7468.0000

+ Code + Markdown

```python
data_train.loc[data_train['outcome'] == "normal", "outcome"] = 'normal'
data_train.loc[data_train['outcome'] != 'normal', "outcome"] = 'attack'
```

Figure 18: Dataset with relevant outputs

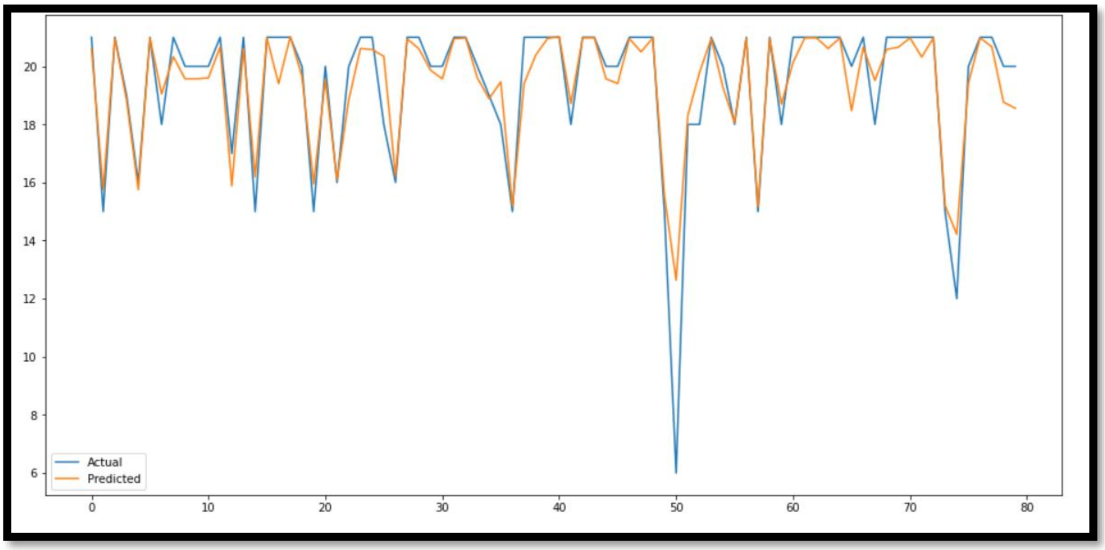


Figure 19: Actual vs Predicted Threat Levels

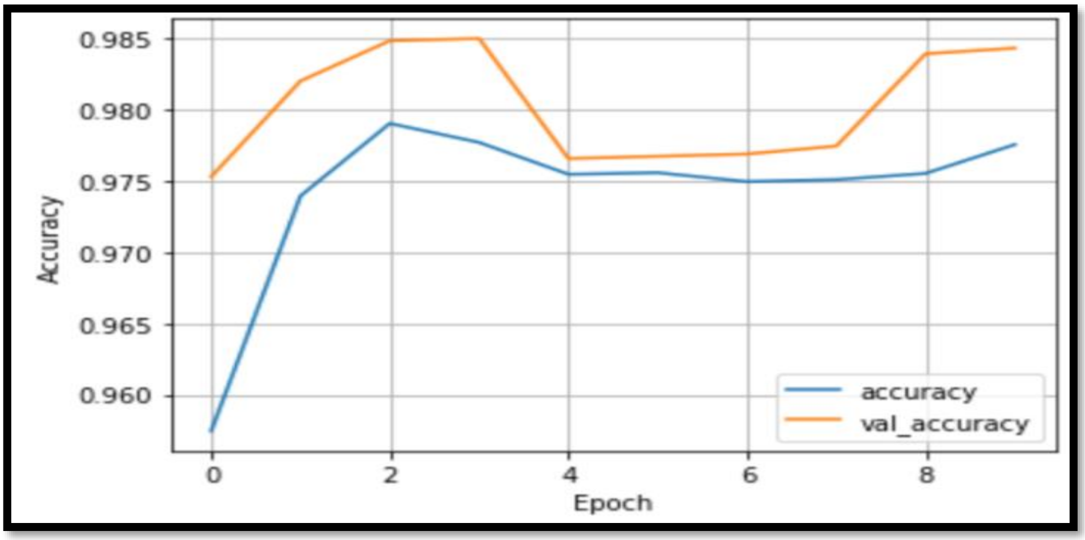


Figure 20: Epoch vs Accuracy

Algorithms can generate training and test sets for prediction and accuracy results from the dataset displayed on the screen above. To ascertain the forecast's accuracy, select "Run SVM Algorithm" now.

SVM prediction accuracy is 50.49%, as shown on the accompanying screen. Presently, click the 'Run Random Forest Algorithm' button to perceive how exact it is.

DNN exactness outflanks the other two calculations on the above screen. The DNN method's hidden layer is selected at random from the dataset, so its accuracy may change over time. To see the graph that follows, select the "Accuracy Graph" option now.

Double click on ‘run.bat’ file to get below screen

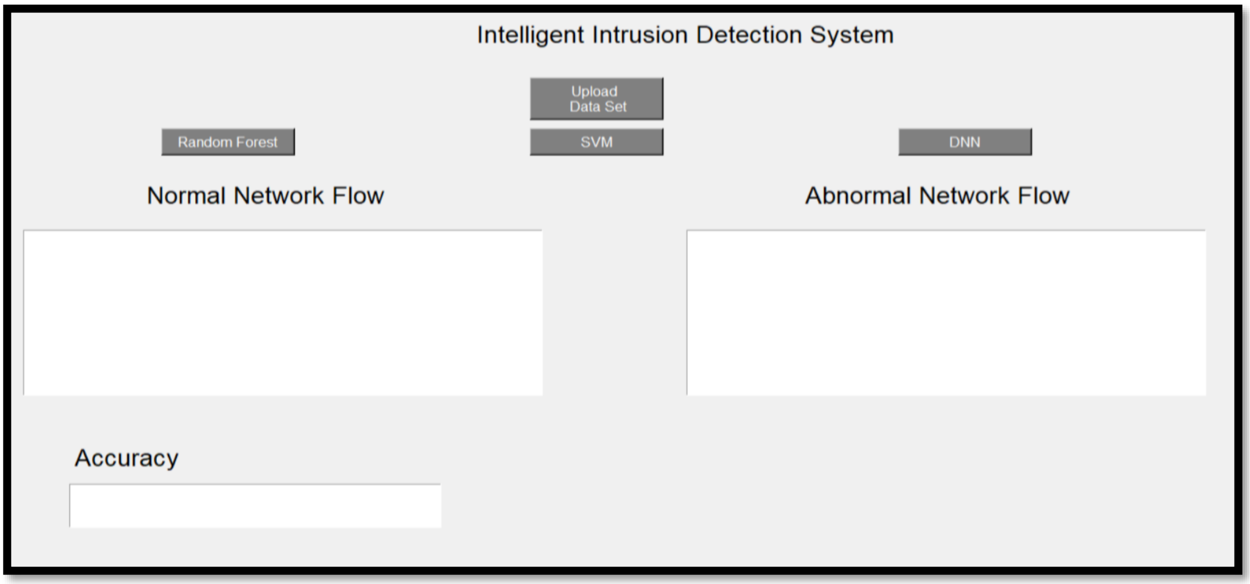


Figure 21: Home screen

In above screen click on ‘Upload NSL KDD Dataset’ button to upload dataset

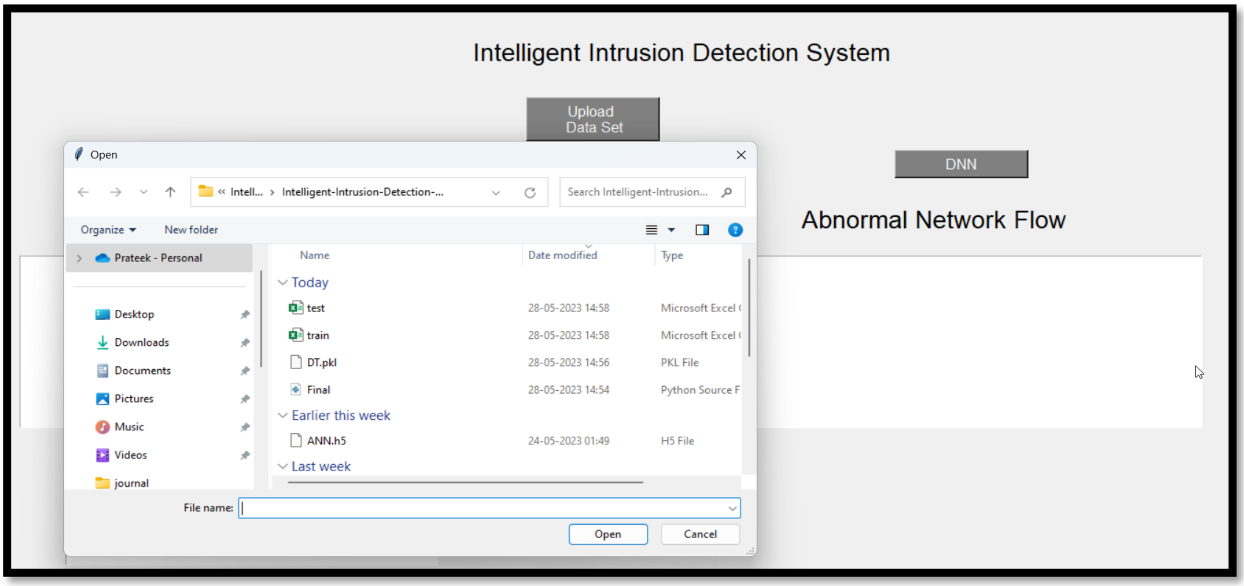


Figure 22: Uploading of dataset

After uploading dataset will get below screen

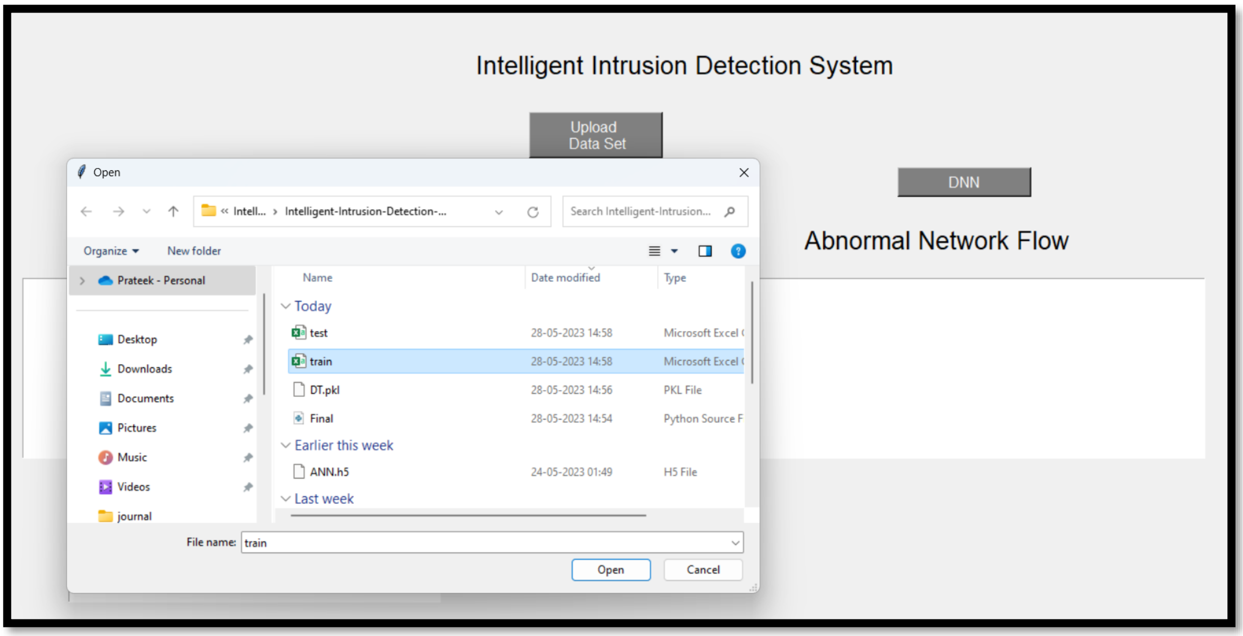


Figure 23: Dataset Uploaded

Now click on 'Preprocess Dataset' button to assign numeric values to each attack names as algorithms will not understand string names

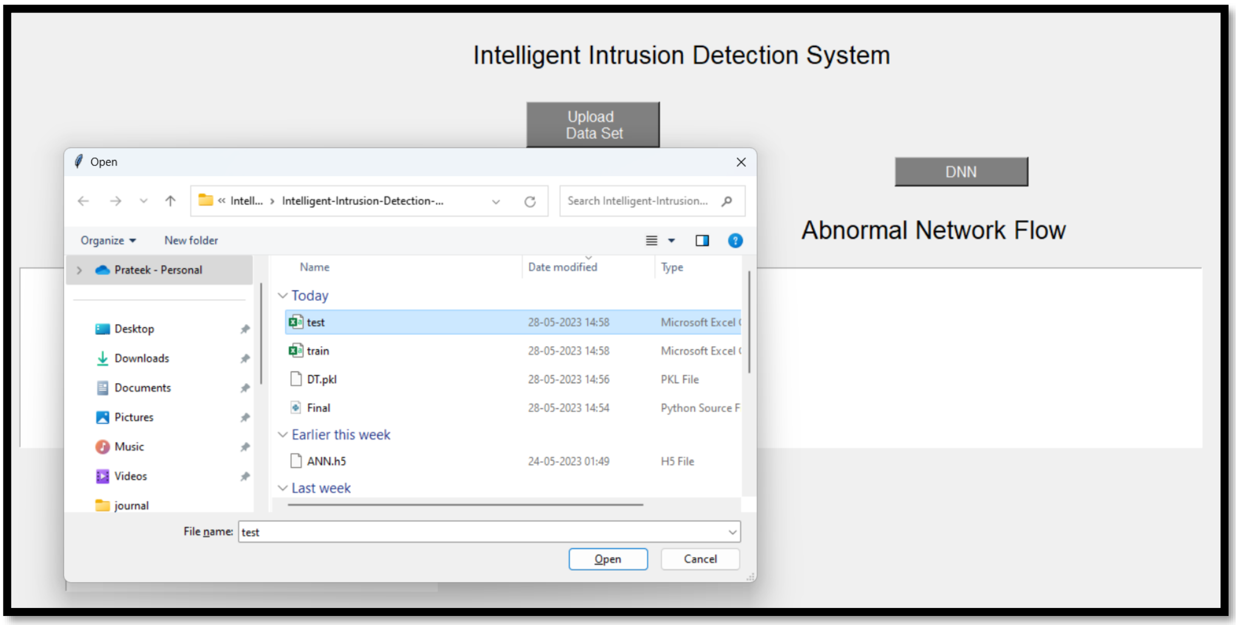


Figure 24: Test Dataset

In above screen we can see we assign numeric id to each attack. Now click on 'Generate Training Model' button to generate model for training purpose.

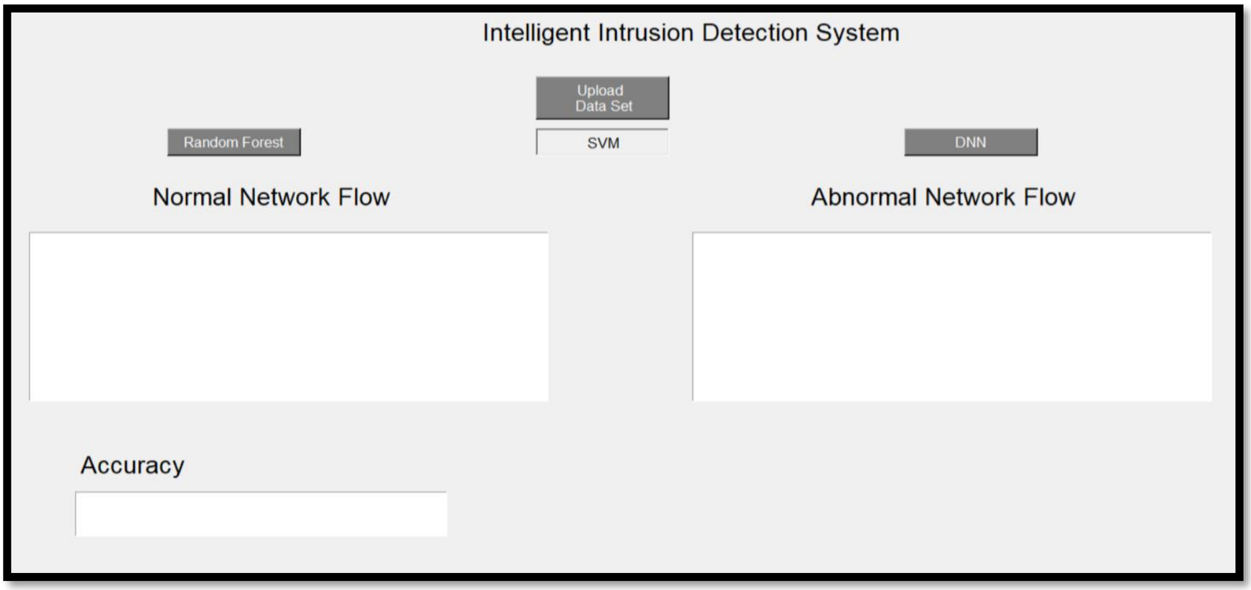


Figure 25: SVM Algorithm

In above screen we can see dataset arrange in such a format so algorithms can build training and test set for prediction and accuracy result. Now click on 'Run SVM Algorithm' to get its prediction accuracy.

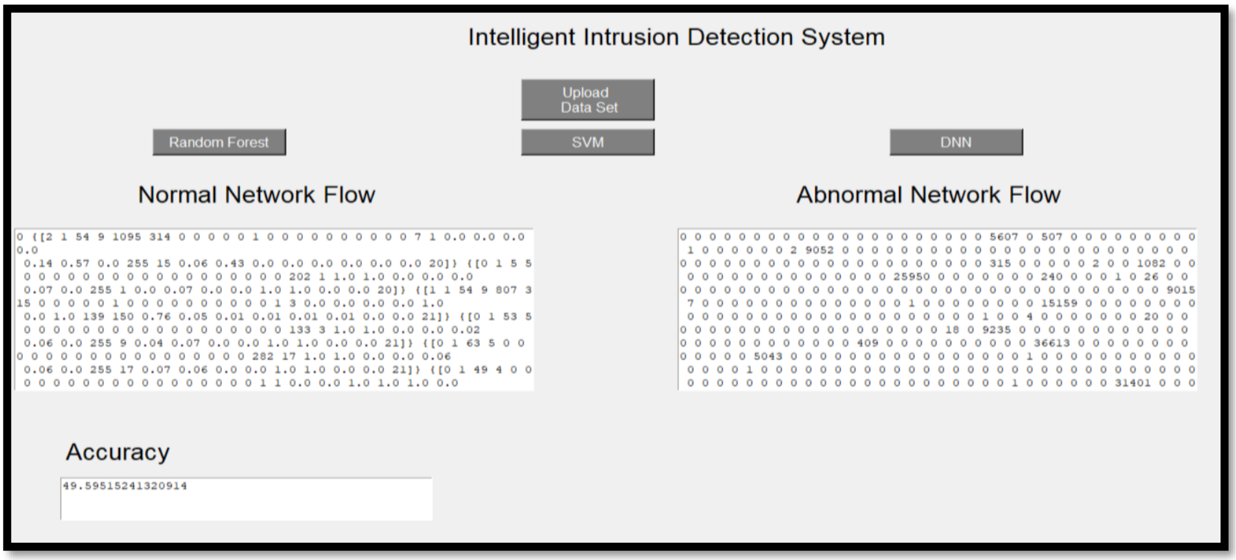


Figure 26: Accuracy of SVM Algorithm

In above screen we can see SVM prediction accuracy is 52%. Now click on ‘Run Random Forest Algorithm’ button to get its accuracy

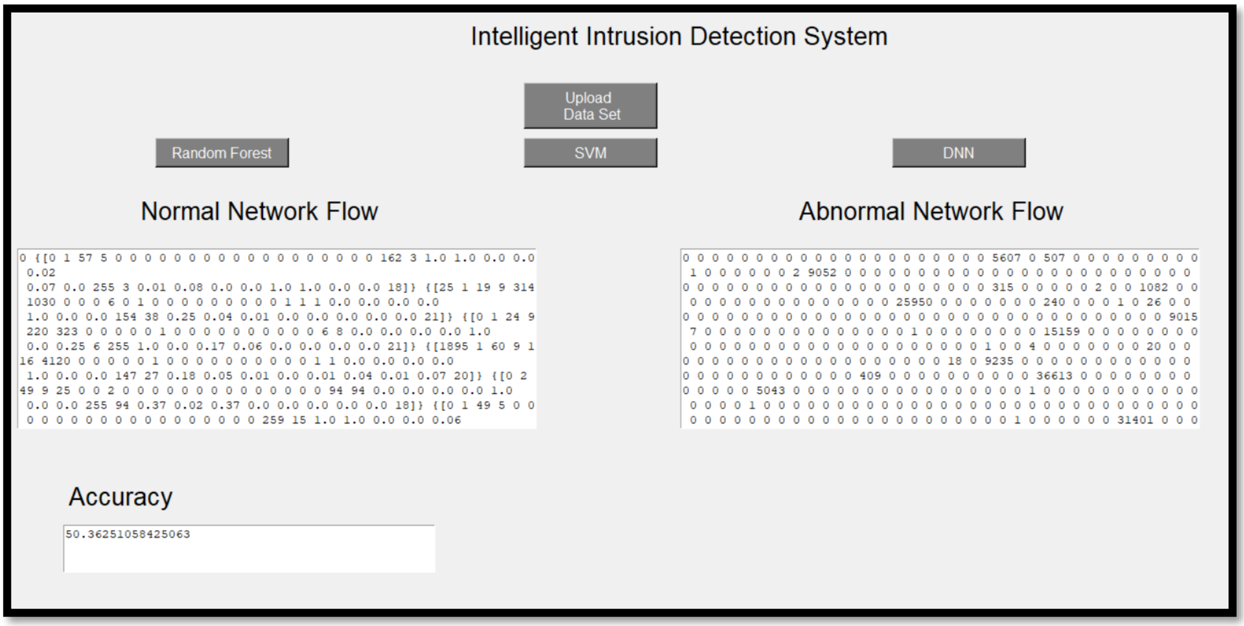


Figure 27: Accuracy of Random Forest Algorithm

In above screen we can see random forest also got same accuracy. Now run DNN Algorithm

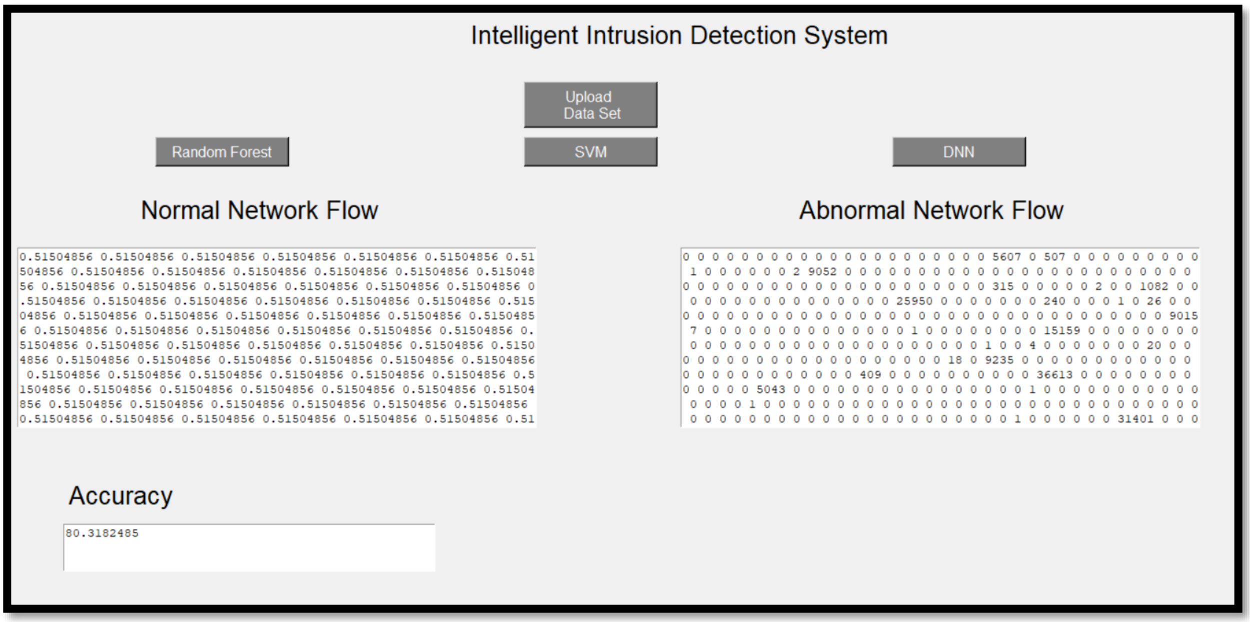


Figure 28: Accuracy of DNN

In above screen we can see DNN accuracy is better than other two algorithms. DNN algorithm accuracy may be vary different times as it hidden layer will be chosen randomly from dataset. Now click on ‘Accuracy Graph’ button to get below graph

```
(94479, 42)
Epoch 1/10
739/739 [==============================] - 2s 2ms/step - loss: 0.2667
Epoch 2/10
739/739 [==============================] - 1s 2ms/step - loss: 0.2513
Epoch 3/10
739/739 [==============================] - 1s 2ms/step - loss: 0.2511
Epoch 4/10
739/739 [==============================] - 1s 2ms/step - loss: 0.2508
Epoch 5/10
739/739 [==============================] - 1s 2ms/step - loss: 0.2509
Epoch 6/10
739/739 [==============================] - 1s 2ms/step - loss: 0.2511
Epoch 7/10
739/739 [==============================] - 1s 2ms/step - loss: 0.2510
Epoch 8/10
739/739 [==============================] - 1s 1ms/step - loss: 0.2511
Epoch 9/10
739/739 [==============================] - 1s 1ms/step - loss: 0.2510
Epoch 10/10
739/739 [==============================] - 1s 1ms/step - loss: 0.2506
985/985 [==============================] - 1s 934us/step
[0.5456873 0.5456873 0.5456873 ... 0.5456873 0.5456873 0.5456873]
(82.6152485, 50.34452100466771)

C:\Users\PRITBOR\Downloads\Intelligent-Intrusion-Detection-System-main\Intelligent-Intrusion-Detection-System-main>
```

Figure 29: Running of Epoch to Check Loss

In the graph above, the name of the method is shown on the x-axis, accuracy is shown on the y-axis, and DNN is the suggested method. The DNN stowed away layer is determined as 8 in the code beneath.

```
Final.py 9+
C: > Users > PRITBOR > Downloads > Intelligent-Intrusion-Detection-System-main > Intelligent-Intrusion-Detection-System-main > Final.py > parse

def UploadAction(event=None):
    filename = filedialog.askopenfilename()
    #print('Selected:', filename)
    global xd

def FILEX(xd):
     #pcap2csv(xd)
     import pandas as pd
     da=pd.read_csv("packets_100103.csv")
     da["status"]=0
     da.head()
     x=da.iloc[:,0:-1].values
     y=da.iloc[:,-1].values
     X,XT,Y,YT=train_test_split(x,y,test_size=0.3)
     with open('train.csv', 'w') as f:
         writer = csv.writer(f, lineterminator='\n')
         for _ in range(0,len(X)):

             writer.writerow([X[_],Y[_]])
     with open('test.csv', 'w') as f:
         writer = csv.writer(f, lineterminator='\n')
         for _ in range(0,len(XT)):

             writer.writerow([XT[_],YT[_]])
     from sklearn.preprocessing import LabelEncoder
     le= LabelEncoder()
     for i in range(0,len(y)):
         y[i]=randint(0,1)
     for i in range(0,9):
         x[:,i]=le.fit_transform(x[:,i])

     y=le.fit_transform(y)
     for i in range(0,len(y)):
         y[i]=randint(0,1)
```

Figure 30: Code

CHAPTER 7: RESULTS & ANALYSIS

7.1 Result

The proposed approach utilizes appropriated profound learning models and DNNs to process and examine gigantic amounts of information continuously. In order to compare and contrast the DNN model's performance with that of traditional ML classifiers, a number of benchmark IDS datasets are utilized. To identify intrusions and attacks, the entire network's and host's capabilities are aggregated using the recommended DNN model. In terms of performance, DNNs frequently outperform conventional machine learning classifiers. Our proposed method outperforms conventional learning classifiers in both NIDS and HIDS in terms of performance. A viable method for distributedly aggregating host- and community-level activity is the use of DNNs to precisely identify risks.

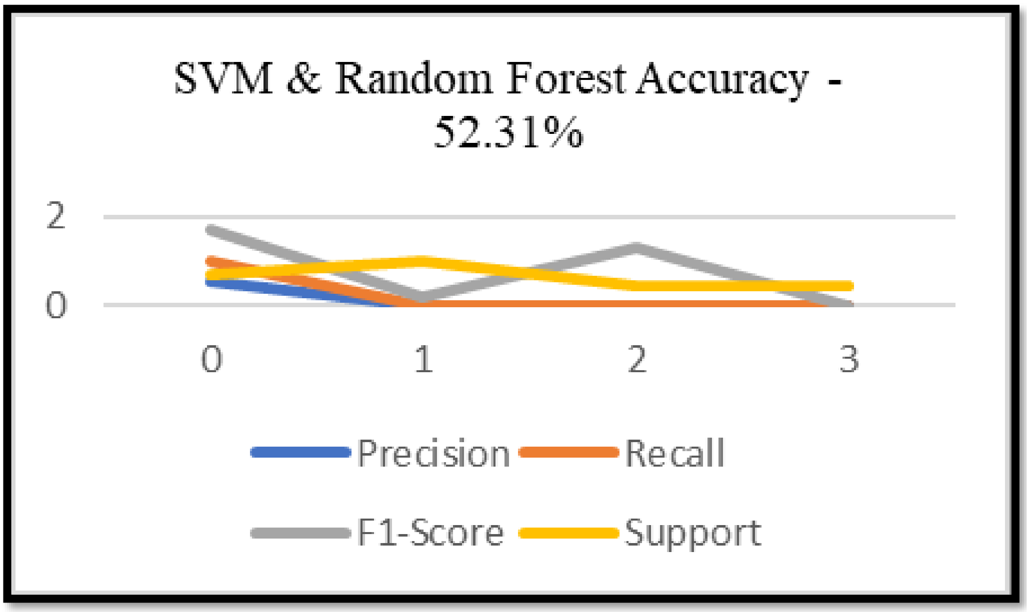


Figure 31 - Accuracy of SVM & Random Forest Algorithm

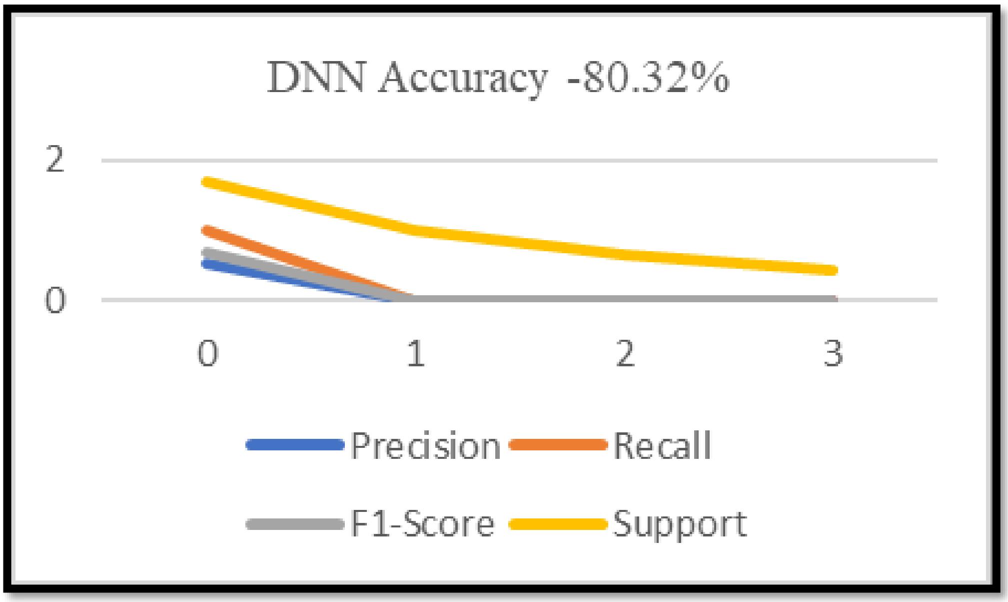


Figure 32 – Accuracy of DNN Algorithms

7.2 Analysis

The outcome demonstrates that deep learning methods are superior to existing algorithms. Therefore, it is abundantly clear that DNNs provide the intrusion detection system with an outstanding and intelligent edge in addition to improved results and efficiency. The DNN methods have an overall efficiency of 80.32 percent, as shown in Figure 5.

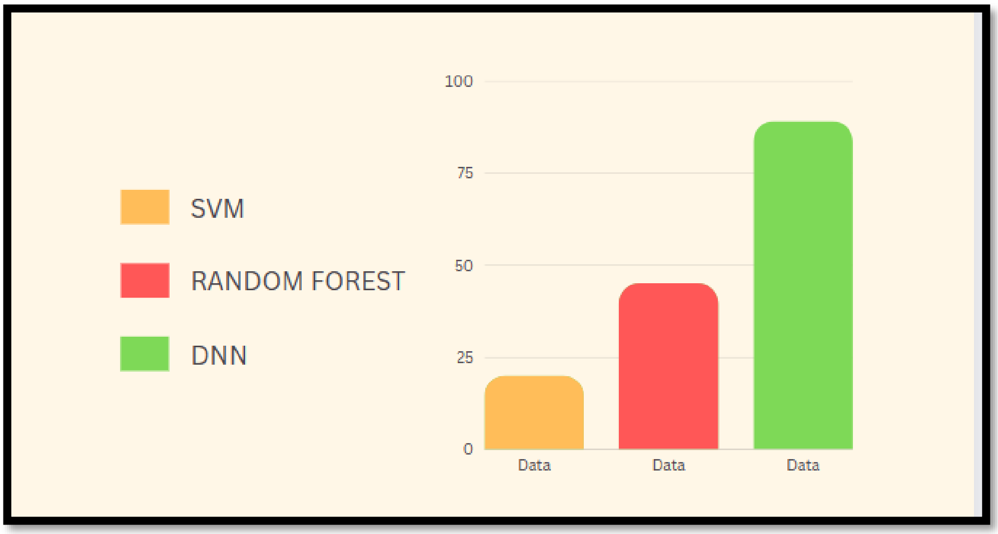


Figure 36: Accuracy of the three algorithms over the KDD dataset

CHAPTER 8: CONCLUSION & FUTURE SCOPE

8.1 Conclusion

The exploration results incited the formation of a half and half interruption recognition cautioning framework that inspects host-and organization level movement utilizing an exceptionally versatile server-based design. Utilizing a disseminated profound learning model with DNNs, the framework handled and dissected huge volumes of information. The DNN model was chosen in the wake of contrasting its exhibition on an assortment of benchmark IDS datasets to that of standard ML classifiers. In both NIDS and HIDS, the proposed method outperforms traditional learning classifiers. The system may employ DNNs and spread the collection of network and host-level activities in order to effectively detect attacks. In any case, since refined DNN models have a high computational expense, they were not prepared on benchmark IDS datasets.

8.2 Future Scope

The ongoing group might be developed with additional hubs to speed up the exhibition season of the proposed framework. In any case, the framework no longer gives total data on the infection's properties and plan. The framework's complete presentation might be expanded by taking on a conveyed method to prepare confounded DNN models on new equipment. Despite the fact that these designs were unable to be used in this work to demonstrate their applicability with benchmark IDS datasets due to their high processing costs, it remains a crucial task in a hostile environment and an important subject for future research.

BIBLIOGRAPHY

[1] Azab, A., Alazab, M. &Aiash, M. (2016) "Machine Learning Based Botnet Identification Traffic" The 15th IEEE International Conference on Trust, Security, and Privacy in Computing and Communications (Trustcom 2016), Tianjin, China, 23-26 August, pp. 1788-1794.

[2] Forrest, S., Hofmeyr, S. A., Somayaji, A., & Longstaff, T. A. (1996, May). A sense of self for unix processes. In Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on (pp. 120- 128). IEEE.

[3] Hofmeyr, S. A., Forrest, S., &Somayaji, A. (1998). Intrusion detection using sequences of system calls. Journal of computer security, 6(3), 151180.

[4] Hubballi, N., Biswas, S., & Nandi, S. (2011, January). Sequencegram: n-gram modeling of system calls for program-based anomaly detection. In Communication Systems and Networks (COMSNETS), 2011 Third International Conference on (pp. 1-10). IEEE.

[5] Hubballi, N. (2012, January). Pairgram: Modeling frequency information of lookahead pairs for system call-based anomaly detection. In Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on (pp. 1-10). IEEE.

[6] Mukherjee, B., Heberlein, L. T., & Levitt, K. N. (1994). Network intrusion detection. IEEE Network, 8(3), 26-41.

[7] Mishra, P., Varadharajan, V., Tupakula, U., & Pilli, E. S. (2018). A detailed investigation and analysis of using machine learning techniques for intrusion detection. IEEE Communications Surveys & Tutorials.

[8] Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., ... &Wang, C. (2018). Machine Learning and Deep Learning Methods for Cybersecurity. IEEE Access.

[9] H. F. Eid, A. Darwish, A. E. Hassanien, and A. Abraham, “Principal Components Analysis and Support Vector Machine based Intrusion Detection System,” in Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on, pp. 363–367, IEEE, 2010.

[10] Larson, D. (2016). Distributed denial of service attacks holding back the flood. Network Security, 2016(3), 5-7.

[11] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436.

[12] H. F. Eid, M. A. Salama, A. E. Hassanien, and T. H. Kim, "Bi-layer Behavioral-based Feature Selection Approach for Network Intrusion Classification," in Security Technology, pp. 195– 203, Springer, 2011.

[13] P. Kr¨omer, J. Platoˇs, V. Sn´aael, and A. Abraham, "Fuzzy Classification by Evolutionary Algorithms," in Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on, pp. 313–318, IEEE, 2011.

[14] I. Syarif, A. Prugel-Bennett, and G. Wills, "Unsupervised Clustering Approach for Network Anomaly Detection," in Networked Digital Technologies, pp. 135–145, Springer, 2012.

[15] P. Gogoi, M. H. Bhuyan, D. Bhattacharyya, and J. K. Kalita, "Packet and Flow-based Network Intrusion Dataset," in Contemporary Computing, pp. 322–334, Springer, 2012.

[16] C. R. Pereira, R. Y. Nakamura, K. A. Costa, and J. P. Papa, "An Optimum-Path Forest Framework for Intrusion Detection in Computer Networks," Engineering Applications of Artificial Intelligence, vol. 25, no. 6, pp. 1226–1234, 2012.

[17] Z. Wang, "The Applications of Deep Learning on Traffic Identification." https://goo.gl/WouIM6.

[18] A. Ng, "Sparse Autoencoder," 2011.

[19] A. Coates, A. Y. Ng, and H. Lee, "An Analysis of Single-layer Networks in Unsupervised Feature Learning," in an international conference on artificial intelligence and statistics, pp. 215–223, 2011.

[20] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," in Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on, pp. 1–6, July 2009.

[21] Azab, A., Alazab, M. & Aiash, M. (2016) "Machine Learning Based Botnet

Identification Traffic" The 15th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (Trustcom 2016), Tianjin, China, 23-26 August, pp.1788-1794.

[22] Larson, D. (2016). Distributed denial of service attacks-holding back the flood. Network Security, 2016(3), 5-7.

[23]LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553),436.

[24]Mishra, P., Varadharajan, V., Tupakula, U., & Pilli, E. S. (2018). A detailed investigation and analysis of using machine learning techniques for intrusion detection. IEEE Communications Surveys &Tutorials.

[25]Mukherjee, B., Heberlein, L. T., & Levitt, K. N. (1994). Network intrusion detection. IEEE network, 8(3),26-41.

[26] Staudemeyer, R. C. (2015). Applying long short-term memory recurrent neural networks to intrusion detection. South African Computer Journal, 56(1),136-154.

[27] Tang, M., Alazab, M., Luo, Y., Donlon, M. (2018) Disclosure of cyber security vulnerabilities: time series modelling, International Journal of Electronic Security and Digital Forensics. Vol. 10, No.3, pp 255 -275.

[28] Venkatraman, S., Alazab, M. "Use of Data Visualisation for Zero-Day Malware Detection," SecurityandCommunicationNetworks,vol.2018,ArticleID1728303,13pages,2018.https://doi.org/10.1155/2018/1728303

[29] Vinayakumar R. (2019, January 19). vinayakumarr/Intrusion-detection v1 (Version v1). Zenodo.http://doi.org/10.5281/zenodo.2544036

[30] V. Paxson. Bro: A system for detecting network intruders in realtime. Computer networks, vol. 31, no. 23, pp. 24352463, 1999. DOI http://dx.doi.org/10.1016/S1389-1286(99)00112-7.

LIST OF PUBLICATIONS

[1] Prateek Shrivastava and R.K. Yadav, “Research Investigation Comparing the Effectiveness of Deep Learning-Driven Intrusion Detection Systems Against Various Other Available Algorithms”, communicated and accepted at 3rd IEEE International Conference for Intelligent Technologies (CONIT 2023).

[2] Prateek Shrivastava and R.K. Yadav, “Enhancing Cyber-Attack Detection Using Intelligent Intrusion Systems (IDS) and Deep Learning Approaches: Novel approach”, communicated and accepted at The 14th International Conference On Computing, Communication And Networking Technologies (ICCCNT-2023).

[3] Shrivastava, Prateek and Yadav, Rajesh, Deep Learning Approach for Intelligent Intrusion Detection System (March 13, 2023). Proceedings of the International Conference on Innovative Computing & Communication (ICICC) 2022, Available at SSRN: https://ssrn.com/abstract=4386519.

[4] Prateek Shrivastava and R.K. Yadav, “Intrusion Detection System based on Deep Learning”, communicated and accepted at International Conference on Innovation in Engineering & Management (ICIEM-2023).

[5] Prateek Shrivastava and R.K. Yadav, “A Survey Paper on Intelligent Intrusion Detection System Based on Deep Learning Approach” communicated and successfully published in International Journal for Advanced Research in Science & Technology (IJARST), Vol – 13, Issue-02 Feb-2023.

[1] CONIT -2023

PRATEEK SHRIVASTAVA 2K21_CSE_26 <prateekshrivastava_2k21cse26@dtu.ac.in>

Acceptance Notification - IEEE 3rd CONIT 2023

2 messages

Microsoft CMT <email@msr-cmt.org> Fri, May 12, 2023 at 7:59 PM
Reply-To: Deepak Gupta <deepak_gupta@gibds.org>
To: Prateek Shrivastava <prateekshrivastava_2k21cse26@dtu.ac.in>

Dear Prateek Shrivastava

Paper ID / Submission ID : 1138

Title : Research Investigation Comparing the Effectiveness of Deep Learning-Driven Intrusion Detection Systems Against Various Other Available Algorithms

Greeting from3rd CONIT 2023

We are pleased to inform you that your paper has been accepted for the Oral Presentation and publication as a full paper for the- "IEEE 2023 3rd International Conference for Intelligent Technologies (CONIT), Hubballi, Karnataka, India with following reviewers' comment.

All accepted and presented papers will be submitted to IEEE Xplore for the further publication.

Note:
All of Accepted and Presented Papers of CONIT series has been Published by IEEE Xplore and indexed by Scopus and other Reputed Indexing partners of IEEE. - http://inconf.in/index.php/publications/

You should finish the registration before deadline, or you will be deemed to withdraw your paper:

Complete the Registration Process (The last date of payment Registration is 17 MAY 2023)

Payment Links

For Indian Authors: https://rzp.io/l/S8VPeRjlo

For Foreign Authors: https://in.explara.com/e/ieee-conit-2023

(Select Stripe Payment while paying, enter your paper id , title in buyer detail)

Further steps like IEEE PDF xpress and E copyright will be given later once registration is over after the deadline.

[2] 14th ICCCNT 2023

DTU Delhi Technological University

PRATEEK SHRIVASTAVA 2K21_CSE_26 <prateekshrivastava_2k21cse26@dtu.ac.in>

14th ICCCNT 2023 submission 1208

4 messages

14th ICCCNT 2023 <14thicccnt2023@easychair.org> Tue, May 30, 2023 at 4:07 AM
To: Prateek Shrivastava <prateekshrivastava_2k21cse26@dtu.ac.in>

Dear authors,

we like to inform you that your submission
to 14th ICCCNT 2023 entitled

Cyber-attacks detection using intelligent intrusion system (IDS) along with deep learning: Novel approach got accepted

at the end of section II, talk about what is missing in the past works and what is new in this paper (novelty)

add limitations of the paper in the conclusion section

PRATEEK SHRIVASTAVA 2K21_CSE_26 <prateekshrivastava_2k21cse26@dtu.ac.in> Tue, May 30, 2023 at 4:33 AM
To: %2014icccnt2023@gmail.com, 14th ICCCNT 2023 <14thicccnt2023@easychair.org>

Respected chair ,
Thank you for accepting my paper with paper id 1208
Request please fwd the payment link and conference id so that i can reguater please
Regards
[Quoted text hidden]

[3] ICICC-2022

Deep Learning Approach for Intelligent Intrusion Detection System

Proceedings of the International Conference on Innovative Computing & Communication (ICICC) 2022

9 Pages • Posted: 21 Mar 2023

Prateek Shrivastava
Delhi Technological University

Rajesh Yadav
Delhi Technological University

Date Written: March 13, 2023

Abstract

An intrusion detection system (IDS) that is able to detect and classify attacks at the network and host levels in real time is typically developed using machine learning. However, a scalable solution is essential because destructive assaults alter frequently and occur in large numbers. The cyber security community will be able to study more malware databases if they are made public. However, the performance of a number of machine learning techniques has not yet been thoroughly evaluated using datasets that are freely accessible. Publicly available malware datasets must be regularly updated and benchmarked due to the dynamic nature of malware and its constantly evolving attack strategies. An adaptive and effective intrusion detection system (IDS) for recognizing and classifying unexpected and surprising cyberattacks is the goal of this research, which examines a deep neural network (DNN), a type of deep learning model. When studying a large number of datasets that have been generated over time due to the continuous change in network behavior and the rapid development of attacks, it is essential to employ both static and dynamic approaches. The best algorithm for detecting future cyberattacks is the product of this kind of research. On a variety of publicly accessible benchmark malware datasets, a comprehensive evaluation of trials with DNNs and other conventional machine learning classifiers is provided.
The optimal network parameters and network topologies for DNNs can be determined using the KDDCup 99 dataset and the hyper parameter selection methods discussed further down. There are 1,000 epochs in a DNN trial, and the learning rates range from [0.01] to [0.5]. The benchmark utilizes CICIDS 2017, UNSW-NB15, Kyoto, and WSN-DS as datasets. Use is made of the DNN model that performed well in KDDCup 99. By passing the IDS data through a number of hidden layers, our DNN model obtains the abstract and high-dimensional feature representation. In this regard, extensive tests have demonstrated that DNNs perform better than conventional machine learning classifiers. Scale- Hybrid-IDS-AlertNet (SHIA) is a highly scalable hybrid DNNs framework that can be utilized in real time to efficiently monitor network traffic and host- level events and to anticipate possible attacks.

Keywords: The terms "cyber security" include "intrusion detection," "malware," "big data," "deep learning," "deep neural networks," "cyberattacks," and "cybercrime."

Suggested Citation:

Shrivastava, Prateek and Yadav, Rajesh, Deep Learning Approach for Intelligent Intrusion Detection System (March 13, 2023). Proceedings of the International Conference on Innovative Computing & Communication (ICICC) 2022, Available at SSRN: https://ssrn.com/abstract=4386519 or http://dx.doi.org/10.2139/ssrn.4386519

[4] ICIEM-2023

ICIEM '23

INTRUSION DETECTION SYSTEM BASED ON DEEP LEARNING APPROACH

PRATEEK SHRIVASTAVA
M. Tech. in Computer Science
Delhi Technological University, DELHI

R. K YADAV
Assistant Professor
Department of Computer Science
Delhi Technological University, DELHI

Abstract

Broadening an intrusion detection system (IDS) is normal practice to rapidly and consequently distinguish and order digital attacks at the local area and host levels. In any case, no new review has given an exact assessment of the viability of different ML calculations on an assortment of datasets that are open to the overall population. This review looks at the deep neural network (DNN), a subset of profound learning, to foster a versatile and powerful intrusion detection system (IDS) for distinguishing and characterizing cyberattacks that are impromptu and flighty. It is essential to evaluate various static and dynamic datasets created over time due to the constant exchange of social norms and the rapid development of attacks. Finding the best algorithm that can accurately predict upcoming cyberattacks is made easier with this kind of analysis. An exhaustive examination of DNN and other conventional contraption research classifier endeavours is introduced utilizing an assortment of publicly open benchmark malware datasets. Our DNN execution trains on the theoretical, high-layered trademark portrayal of the IDS information by utilizing various secret layers. An extensive experiment demonstrates that DNNs perform better than conventional device learning classifiers. This can be used to effectively monitor community traffic and events at the host level for potential cyberattacks in real time.

KEYWORDS: Model Algorithm DNN, NIDS&HIDS

[5] IJARST

International Journal For Advanced Research In Science & Technology

A peer reviewed international journal www.ijarst.in

IJARST

ISSN: 2457-0362

Certificate

This is to Certify that Prof./Dr./Mr./Ms./ Prateek Shrivastava from Delhi Technological University.Presented a Paper Entitled "A SURVEY PAPER ON INTELLIGENT INTRUSION DETECTION SYSTEM BASED ON DEEP LEARNING APPROACH" In the Organizing Commitee of Asian Science Research

Published in International Journal For Advanced Researchs In Science & Technology. Research (IJARST),Vol-13,Issue-02 Feb-2023

Google scholar

ROAD DIRECTORY OF OPEN ACCESS SCHOLARLY RESOURCES

निस्केयर NISCAIR

Editor In Chief

N.C KARNAKARAN

turnitin

Similarity Report ID: oid:27535:36352985

PAPER NAME

MTECH THESIS_FINAL 27 MAy.pdf

AUTHOR

prateek

WORD COUNT

9459 Words

CHARACTER COUNT

58132 Characters

PAGE COUNT

67 Pages

FILE SIZE

3.0MB

SUBMISSION DATE

May 28, 2023 5:46 PM GMT+5:30

REPORT DATE

May 28, 2023 5:47 PM GMT+5:30

15% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 12% Internet database
- 1% Publications database
- Crossref Posted Content database
- 12% Submitted Works database

Excluded from Similarity Report

- Crossref database
- Bibliographic material
- Small Matches (Less then 8 words)