**Introduction**

## Purpose

The Project concept is to achieve the new method for extracting the information from the Dataset that we have collected using Wireshark network capturing tool that will help to automatically detect the attack like Dos. How such attack are Detected by Machine Learning and ANN module and provide the security from the any anomaly data which attacks your system and provide security to the server.

## Scope

One of the most important issues about our proposed architecture is detecting the attack using intrusion detection system, in order to verify predictions of the system, to determine the requirements of the intrusion detection system and to evaluate the use of neural network and reinforcement learning within the problem domain.

## Product Perspective

This feature will give the user a system which can detect the upcoming attack and which type of attack it is. Hence, giving the user a more secure system to detect the attack and learning the machine using reinforcement learning model. With reinforcement learning model we are not training our machine with one specific model but our machine will train itself even on new database records and propose a model by itself which is best for detecting.

## Product Functions

In this extraction framework, intermediate output of IDS is stored so that only the improved component has to be deployed to the entire database. Extraction is then performed on both the previously processed data from the unchanged components as well as the updated data generated by the improved component by different ANN and compare it with Reinforcement Deep Q-Learning Algorithm. Performing such kind of incremental extraction can result in a tremendous reduction of processing time. The project provides the following functions:

1. Determine the requirements of an intrusion detection system
2. Evaluate the use of artificial neural networks within the problem domain.
3. Determine the most appropriate type of artificial neural network to use
4. Develop an intrusion detection system using the techniques that are found to be best suited to the problem.
5. Train and test the intrusion detection system using test data.
6. Test the system on unseen data.

## Intended Audience and Users

This document is intended to be read by the customers like net developers, project managers, staﬀ, users, testers and documentation writers. This is a technical document and the terms should be understood by the customers. This document is intended for: Developers: In order to be sure they are developing the right project that fulﬁll requirements that provided in this document. Testers: In order to have an exact list of the features and functions that has to respond according to requirements and provided diagrams. Users: In order to get familiar with the idea of the project and suggest other features that would make it even more functional. Documentation Writers: To know what features and in what way they have to explain. What security technologies are required, how the system will response on each users action etc. Advanced end users, end users/desktop and system administrators: In order to know exactly what they have to expect from the system, right inputs and outputs and response in error situations.

## Limitations

While deploying a flexible and efficient NIDS for unknown future attacks we have faced two challenges. Firstly, proper attribute selection of network traffic dataset is difficult for unknown attacks. The attribute selected for one kind of attack may not be compatible for detecting another kind of attack. When this happens, NIDS treats that traffic as normal or show some error. Second challenge is unavailability of labelled real time traffic dataset to develop a NIDS system. It would need an immeasurable effort to label a real time traffic, a raw data with respect to different attacks. Available datasets which are labelled are quite old and contains less attributes to detect any new or unknown attack.

## Assumptions and Dependencies

The system will attempt to classify a particular HTTP request or an HTTP session on a Web Server as normal or anomalous. However, if users are behind a Proxy Server, accurate identifications of the session user may not be possible. Therefore, for the purpose of this project we must assume that a session from a single IP address is the session of a single user.

The HTTP protocol version 1.0 specifies that a single TCP connection will carry a single HTTP message [n3]. HTTP/1.1 allows multiple messages over a single connection this system will assume we are using the HTTP/1.0 protocol.

Fragmentation provides a powerful mechanism for bypassing Intrusion Detection Systems. This system will assume these kinds of measures do not occur and that the received packet has not been fragmented. This project will not implement re-assembly routines.

## Definitions

**Wireshark**

Wireshark is an open-source packet analyser, which is used for education, analysis, software development, communication protocol development, and network troubleshooting. It is used to track the packets so that each one is filtered to meet our specific needs. It is commonly called as a sniffer, network protocol analyser, and network analyser. It is also used by network security engineers to examine security problems. Wireshark is a free to use application which is used to apprehend the data back and forth. It is often called as a free packet sniffer computer application. It puts the network card into an unselective mode, i.e., to accept all the packets which it receives.

**Flowtbag**

The purpose of this tool is to calculate flow statistics from a given capture file. Flowtbag was designed with offline processing as the primary focus**.**

**Hping**

hping is a command-line oriented TCP/IP packet assembler/analyser. The interface is inspired to the ping(8) Unix command, but hping isn’t only able to send ICMP echo requests. It supports TCP, UDP, ICMP and RAW-IP protocols, has a traceroute mode, the ability to send files between a covered channel, and many other features. It is used as a security tool to secure network and host.

**TShark**

TShark, a well-known and powerful command-line tool and is used as a network analyser. It is developed by Wireshark. It’s working structure is quite similar to Tcp dump, but it has some powerful decoders and filters. TShark is capable of capturing the data packets information of different network layers and display them in different formats. TShark is used to analyze real-time network traffic and it can read .pcap files to analyze the information, dig into the details of those connections, helping security professionals to identify their network problem.

**Python 3**

Python is a dynamic object-oriented programming language that can be used for many kinds of software development and other fields such as data science. It offers strong support for integration with other languages and tools, comes with extensive standard libraries, and can be learned in a few days. Many Python programmers report substantial productivity gains and feel the language encourages the development of higher quality, more maintainable code. Python 3.0 (a.k.a. "Python 3000" or "Py3k") is a new version of the language that is incompatible with the 2.x line of releases. The language is mostly the same, but many details, especially how built-in objects like dictionaries and strings work, have changed considerably, and a lot of deprecated features have finally been removed. Also, the standard library has been reorganized in a few prominent places.

**Anaconda**

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012. As an Anaconda, Inc. product, it is also known as Anaconda Distribution or Anaconda Individual Edition, while other products from the company are Anaconda Team Edition and Anaconda Enterprise Edition, which are both not free. Anaconda offers a text-mode and GUI mode, so users can install on a wide range of systems. It is designed to be easily portable and supports a wide range of hardware platforms (IA-32, Itanium, DEC Alpha, IBM ESA/390, PowerPC, ARMv8). It supports installing from local storage devices like CD-ROM drives and hard disks as well as from network resources via FTP, HTTP, or NFS. Installations can be automated with the use of a kickstart file, that automatically configures the installation, allowing users to run it with minimal supervision. Before starting the OS installation process, the installer checks the system hardware and resource requirements. Only if the requirements are satisfied does it start the installation process.

**Jupyter Notebook**

JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data. JupyterLab is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning. JupyterLab is extensible and modular: write plugins that add new components and integrate with existing ones. The Jupyter Notebook is an opensource web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and much more.

**Spyder**

Spyder is a powerful scientific environment written in python, for python and designed by and for scientists, engineers and data analysis. It offers a unique combination of the advanced editing, analysis, debugging and profiling of a comprehensive development tool with data exploration and interactive execution, deep inspection and beautiful visualization capabilities of a scientific package. Beyond its many built in features its abilities can be extended even further via its plugin-system and API. Furthermore Spyder can also be used as a PYQt5 extension library allowing developers to build upon its functionalities and embedded its components such as interactive console in its own PYQt5 Software.

**Gym**

Gym is a toolkit for developing and comparing reinforcement learning algorithms. It makes no assumptions about the structure of your agent, and is compatible with any numerical computation library, such as TensorFlow or Theano. The gym library is a collection of test problems — environments — that you can use to work out your reinforcement learning algorithms. These environments have a shared interface, allowing you to write general algorithms. Reinforcement learning (RL) is the subfield of machine learning concerned with decision making and motor control. It studies how an agent can learn how to achieve goals in a complex, uncertain environment. However, RL research is also slowed down by two factors:

• The need for better benchmarks. In supervised learning, progress has been driven by large labelled datasets like ImageNet. In RL, the closest equivalent would be a large and diverse collection of environments. However, the existing open-source collections of RL environments don’t have enough variety, and they are often difficult to even set up and use.

• Lack of standardization of environments used in publications. Subtle differences in the problem definition, such as the reward function or the set of actions, can drastically alter a task’s difficulty. This issue makes it difficult to reproduce published research and compare results from different papers.

**TensorFlow**

TensorFlow offers multiple levels of abstraction so you can choose the right one for your needs. Build and train models by using the high-level Keras API, which makes getting started with TensorFlow and machine learning easy. If you need more flexibility, eager execution allows for immediate iteration and intuitive debugging. For large ML training tasks, use the Distribution Strategy API for distributed training on different hardware configurations without changing the model definition. TensorFlow has always provided a direct path to production. Whether it’s on servers, edge devices, or the web, TensorFlow lets you train and deploy your model easily, no matter what language or platform you use.

**Sckit**

Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms. It’s built upon some of the technology you might already be familiar with, like NumPy, pandas, and Matplotlib. The functionality that scikit-learn provides include:

• Regression, including Linear and Logistic Regression

• Classification, including K-Nearest Neighbors

• Clustering, including K-Means and K-Means++

• Model selection

• Preprocessing, including Min-Max Normalization

**Keras**

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research. Keras is the high-level API of TensorFlow 2.0: an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity. Keras empowers engineers and researchers to take full advantage of the scalability and cross-platform capabilities of TensorFlow 2.0: you can run Keras on TPU or on large clusters of GPUs, and you can export your Keras models to run in the browser or on a mobile device.

**NumPy**

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

**Pandas**

Pandas is used for data manipulation, analysis and cleaning. Python pandas is well suited for different kinds of data, such as:

• Tabular data with heterogeneously-typed columns

• Ordered and unordered time series data

• Arbitrary matrix data with row & column labels

• Unlabelled data

• Any other form of observational or statistical data sets

**Matplot Library**

matplotlib. pyplot is a plotting library used for 2D graphics in python programming language. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits. There are several toolkits which are available that extend python matplotlib functionality. Some of them are separate downloads, others can be shipped with the matplotlib source code but have external dependencies

## 1.9 Acronyms and Abbreviations

1. NIDS Network Intrusion Detection System

2. ANN Artificial neural network

3. CNN Convolutional Neural Network

4. KNN K Nearest Neighbor

5. GRU Gated Recurrent Unit

6. LSTM Long Short-Term Memory

7. DNN Deep Neural Network

# Requirements

## 2.1 Overall Description

##### **2.1.1 Product perspective**

This feature will give the user a system which can detect the upcoming attack and which type of attack it is. Hence, giving the user a more secure system to detect the attack and learning the machine using reinforcement learning model. With reinforcement learning model we are not training our machine with one specific model but our machine will train itself even on new database records and propose a model by itself which is best for detecting.

##### **2.1.2 Product functions**

In this extraction framework, intermediate output of IDS is stored so that only the improved component has to be deployed to the entire database. Extraction is then performed on both the previously processed data from the unchanged components as well as the updated data generated by the improved component by different ANN and compare it with Reinforcement Deep Q-Learning Algorithm. Performing such kind of incremental extraction can result in a tremendous reduction of processing time. The project provides the following functions:

1. Determine the requirements of an intrusion detection system
2. Evaluate the use of artificial neural networks within the problem domain.
3. Determine the most appropriate type of artificial neural network to use
4. Develop an intrusion detection system using the techniques that are found to be best suited to the problem.
5. Train and test the intrusion detection system using test data.
6. Test the system on unseen data.

## 2.2 System Features

##### **2.2.1 System feature 1**

Functional Requirements:

1.The user will be able collect the necessary training data using an appropriate tool.

2. The user will be able to train the neural network based analysis engine.

3. The system must be able to be started and stopped by a privileged user of the system.

4. The system is required to notify a system administrator when a sequence of events that is likely to be an attack is encountered.

5. Statistics of the systems operation must be able to be collected for analysis and review.

##### **2.2.2 System feature 2**

Quality and Eﬃciency: Using the training and testing set automatically detect the attacks. Our approach minimizes the need of reprocessing the entire collection of attack in the presence of new extraction goals and deployment of improved processing speed of the server.

## 2.3 Other Non-functional Requirements

##### **2.3.1 Performance requirements**

The system must be easy to train for new users of the software. The mechanism for training the system used must be easy to understand and allow someone with a reasonable background in computer networks to capture the correct data and use the training software to create an analysis engine for their network. Users of the system must be able to start and stop the system easily. Users of the system must be able to configure any parameters the system may use in an easily manageable way.

##### **2.3.2 Software quality attributes**

1.Adaptability: The compiler must be able to accommodate changes to the language implementation as well changes in the machine architecture.

2.Correctness: The compiler generated code should give the exact output as that of the output of the script run using the interpreter.

False negative identifications of attacks should be minimized heavily. This means a higher focus on detecting false negatives should be given to the project rather than trying to reduce false positives, although a minimized false positive rate is desirable as a secondary concern.

**2.3.3 Performance**

Packets must be captured fast enough to enable real-time analysis of the data, and a system administrator should be notified as soon as possible. When the service becomes slow due to heavy traffic, the system should not affect the performance of the rest of network. The system should express a high degree of tolerance to internal faults and external faults with the system itself as well as its host system.

### **2.3.4 Supportability**

Dynamic configuration of the system will allow the network administrators to add a different weight matrix to the system with minimal disruption to the service, and without disrupting any other services on the network.

## 2.3.5 Reliability

System administrators should not be required to re-start the system at any point unless they wish to do so for maintenance reasons. False negative identifications of attacks should be minimized heavily. This means a higher focus on detecting false negatives should be given to the project rather than trying to reduce false positives, although a minimized false positive rate is desirable as a secondary concern. Intrusion Detection Systems are often themselves the target for attacks, so the system should attempt to resist subversion. So, a careful, security conscious view to the development is necessary.

## 2.3.6 Scalability

Scalability is not really an issue in this project, however thought will be given to making this or future system able to scale up to larger identification tasks.

## 2.3.7 Supportability

Dynamic configuration of the system will allow the network administrators to add a different weight matrix to the system with minimal disruption to the service, and without disrupting any other services on the network.

## 2.4 Problem Statement

Our main aim is to provide an additional approach for identifying/ stopping an attack with the use of Reinforcement Learning techniques. The objective of work is to analyse various types of attacks on a network system and improve the detection accuracy rate with new type of dataset by machine learning in network security domain.

Our main research questions are as follows:

• Problemstatement1.1: How to enhance the network intrusion detection system with the use of Reinforcement Learning algorithm by obtaining the optimal policy with the maximum reward?

• Problemstatement1.2: What type of dataset are we going to use while training the model? How the data set will aﬀect the performance of Machine Learning Algorithms?

## 2.5 Technical Model

A technical model is designed to provide an overview of all the functionalities and the features. The Reinforcement algorithms will be developed and combined to work together with the NIDS system in order to obtain the optimal policy and detect the network intrusions and to save the system resources. The technical model can be described as the following steps

• System monitoring

• Decision with changes in the NIDS system like reordering or updating the variables/attributes.

It is necessary to ﬁnd the cause of network intrusion thus, it is important to monitor the system resources parameter, such as CPU and RAM to ﬁnd out which resource metric that could be the bottleneck that makes packets to be dropped or intruded which may be an active attacks which cause interruption, modiﬁcation, or Fabrication. And among passive attacks like Release of message content or Traﬃc Analysis. The most important prerequisites that must be in order before conducting any implementation of the Deep Q Learning algorithm are ﬁnding the right system resource parameter to monitoring part and deployment of the underlying infrastructure.

### **2.5.1 Domain**

The best advantage for a system is when an IDS serves localhost and users over the internet, which is installed on the server side as shown in ﬁgure 3.1. There are four main important things in the system which are the system administrator, monitor, user, and network. The User sends a request to the server over the internet or Local Area Network and the Intrusion Detection System will analyse the packets which are received by the server. This Intrusion Detection System can detect intrusions both internally and externally. It alerts system administrator as soon as it detects an intrusion.

## 2.6 Implementation and Experiment Stage

In pursuance of building the environment and implement the testing, a range of different tools has to be used and combined like installing and conﬁguring the environment, as well for developing the algorithms. After reviewing the tools for implementation, the following tools was chosen.

• Gym: Reinforcement algorithm libraries. More speciﬁcally this toolkit used for developing and comparing Deep Q network algorithm. The main task is to supports the teaching agents. To update the Q-value we have used Gym.

• Tensorﬂow: Numerical computation using data ﬂow graphs and convolution neural network building. We used tensorﬂow to build 4 hidden layers to compute.

• Scikit-learn: Machine learning module, an effective tool for data mining and data analysis. We have used for analysing our data.

• Python: as the scripting language for automating experiments and extracting data.

• Keras: Keras API integrates very easy with the TensorFlow workﬂows. It’s very useful in Reinforcement Learning algorithm.

## 2.7 System Description

Network Intrusion Detection system (NIDS) is a system which monitors network intrusion. Intrusion may be detected by techniques like anomaly detection, signature pattern matching etc. Anomaly detection is a method in which normal network behaviour is captured and any abnormality in the network is detected such as a sudden increase in network traffic rate (number of IP packets per second). Signature pattern matching is a method in which network data is compared with the known attack techniques that are saved in a database. Intrusion is detected and system administrator is alerted about the kind of intrusion when any one of the following events takes place • 1. If a foreign entity has been detected in a log entry.

• 2. If user tries to access information which is beyond his/her access.

• 3. Baseline for critical system resources is measured such as CPU utilization, ﬁle entries, disk activity, user logins etc. Then the system can trigger when there is a deviation from this baseline.

## 2.8 Development of Deep Reinforcement Learning Algorithm

Reinforcement Learning indicates a Machine Learning method in which the agent receives a delayed reward in the next time step to evaluate its previous action. Q-Learning is a value-based Reinforcement Learning algorithm. The goal is to maximize the value function “Q”, which is an expected future reward given a state and action.

## 2.8.1 Creation of Q-Table

Create a table to calculate the maximum expected future reward, for each action in each state in the environment. Each Q-table score will be the maximum expected future reward that an agent will get if it takes an action at a state with the best policy given. To calculate the values for each element of the Q table, we use the Q learning algorithm.

## 2.8.2 Q-Learning Algorithm

Learning the Action Value Function the Action Value Function (or “Qfunction”) takes two inputs “state” and “action.” It returns the expected future reward of that action at that state. Before we explore the environment, the Qtable gives the same arbitrary ﬁxed value (most of the time 0). As we explore the environment, the Q-table will give us a better and better approximation by iteratively updating Q(s,a) using the Bellman Equation.

## 2.8.3 Q-Learning Algorithm Process

At the end of the training, we get “Good Q-table” with best possible results.[6] We can see this Q function as a reader that scrolls through the Q-table to ﬁnd the line associated with our state, and the column associated with our action. It returns the Q value from the matching cell. This is the “expected future reward.” But before we explore the environment, the Q-table gives the same arbitrary ﬁxed value (most of the time 0). As we explore the environment, the Q-table will give us a better and better approximation by attractively updating Q(s,a) using the Bellman Equation. At the end of the training, we get “Good Q-table” with best possible results. 1. Initialise Q- values Q(s,a) arbitrary for all state action pairs. 2. For Life or until learning is stopped. 3. Choose the action (a) in the current world state (s) based on current Qvalue estimates (Q(s, .)). 4. Take the action (a) and observe the outcome state (s’) and reward (r).

## 2.8.4 Steps in designing the algorithm

• Specify Neural network Architecture

• Build Neural network

• Based on the estimated probabilities to select a random action

• Set up the training of the Neural network Using Policy Gradient

– n\_iterations = 10 number of training iterations 10-100

– n\_max\_steps = 100 max steps per episode 100 -1000

– n\_games\_per\_update = 10 train the policy every 10 episodes

– discount\_rate = 0.95 discount rate

• Executing the graph

• Evaluation of the result

## 2.9 Measurement, Analysis and Comparison Stage

This is the last stage and one of the most signiﬁcant task of the thesis work where all of the data is analysed. The tasks done in this stage is outlined below • Executing the algorithm

• Analysing the obtained data and ﬁgures

## 2.9.1 Executing the Deep Q Algorithm

After implementation of Deep Q algorithm, during execution, following are the factors to consider Parameters

• action\_index: object, reward, episode\_over information

• Returns – tuple ob (object) an environment-speciﬁc object representing your observation of the environment.

– reward (ﬂoat) amount of reward achieved by the previous action. The scale varies between environments, but the goal is always to increase your total reward.

– episode\_over (bool) whether it’s time to reset the environment again.

– info (dict) diagnostic information useful for debugging. It can sometimes be useful for learning (for example, it might contain the raw probabilities behind the environment’s last state change).

## 2.10 Analysing the Obtained Data and Figures

The implementation of the above-mentioned experiments is performed on a physical serverrunningUbuntuLinux16.04. After the implementation, the data analysis and plotting will follow. The last stage involves analysing and veriﬁcation of the data that was extracted during the experiment in-depth. The data will be analysed in an attempt to see if the objectives set in this project is met. Analysis connected to the Deep Q Learning algorithms how show well the algorithm performed by comparing the all the attributes in the detection of the network attacks. An additional analysis involves ﬁnding the system resources (CPU, RAM) that cause the network intrusion. This consists of examining the different system resources and analyse which is likely to be the bottleneck. When performing the ﬁnal evaluation and execution of the datasets (explained in the next section) with the Deep Q Learning algorithm, the execution is done over GPU’s which are highly compatible with high performance. The ﬁnal analysis is to check how efficient the Deep Q Learning algorithm performs. Since the network traffic load can differ from time to time, it is important to have an analysis of the performance of the algorithm and how it reacts in response to the network traffic load.

## 2.11 System Architecture

## 2.11.1 UML- Diagrams

## 2.11.1.1 Class Diagram

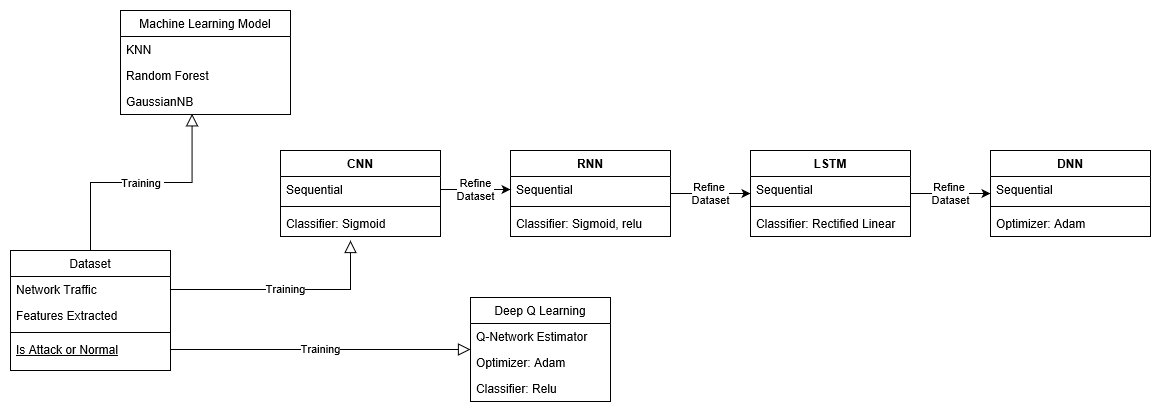


Fig 2.1 Class Diagram of NIDS

## 2.11.1.2 Use-Case Diagram

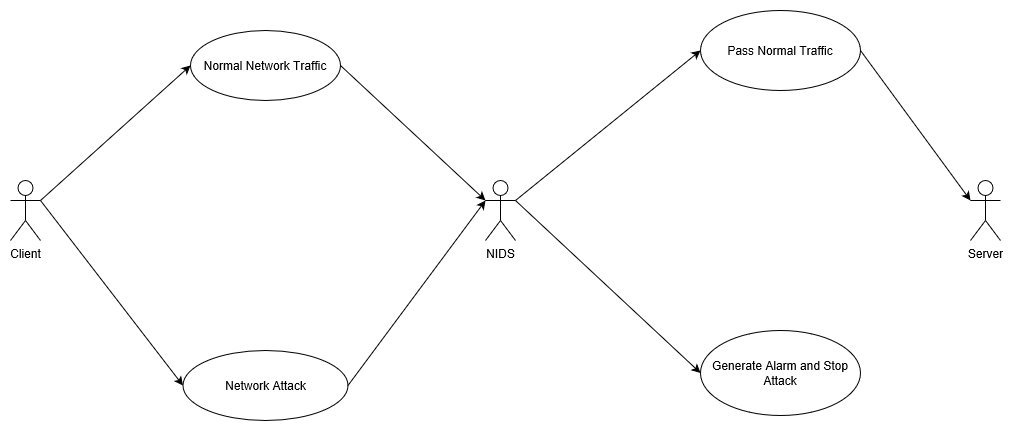


Fig 2.2 Use-Case Diagram of NIDS

## 2.11.1.3 Sequence Diagram

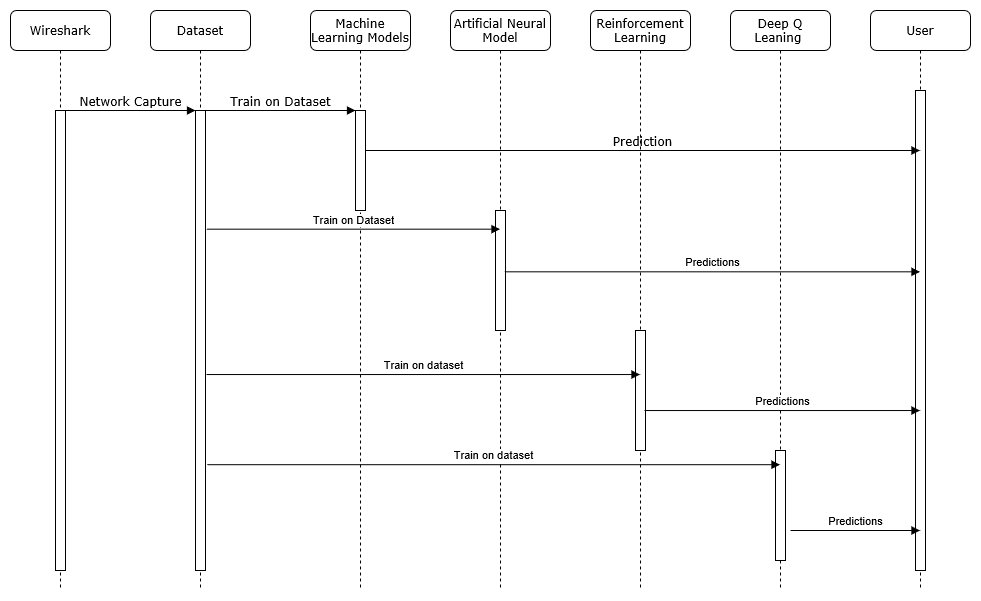


Fig 2.3 Sequence Diagram of NIDS

## 2.11.1.4 Component Diagram

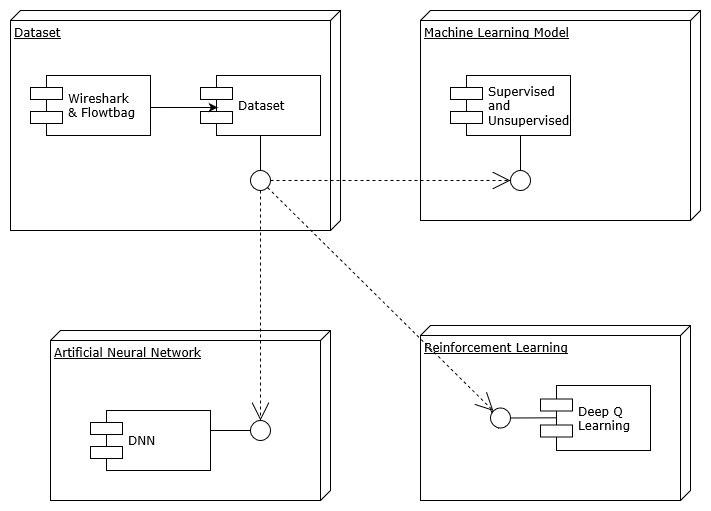


Fig 2.4 Component Diagram of NIDS

## 2.11.1.5 Activity Diagram

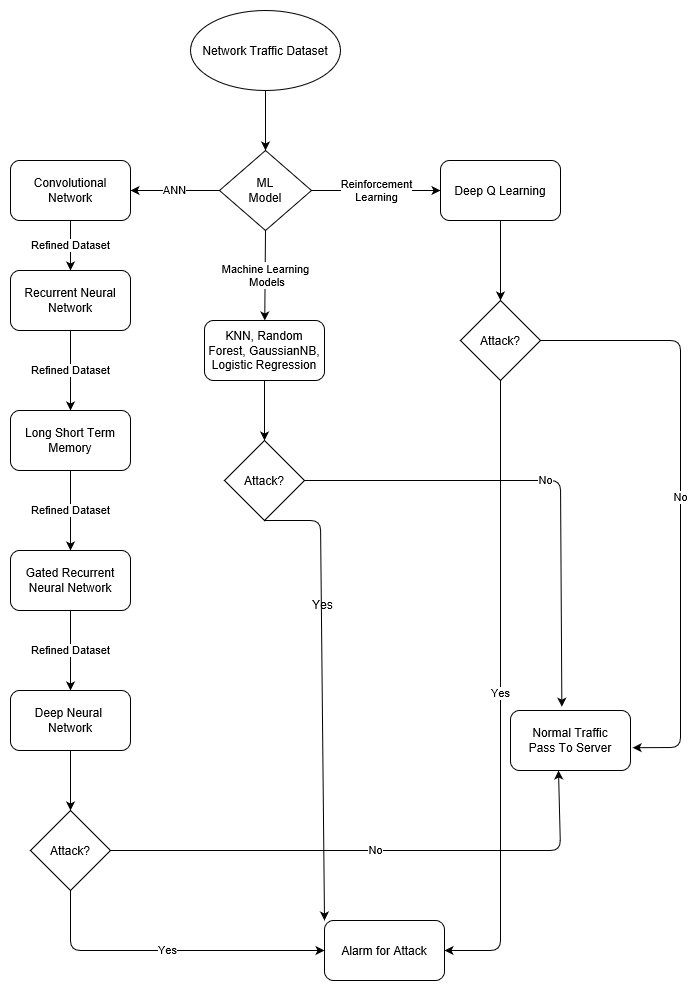


Fig 2.5 Activity Diagram of NIDS

## 2.12 Architecture of NIDS

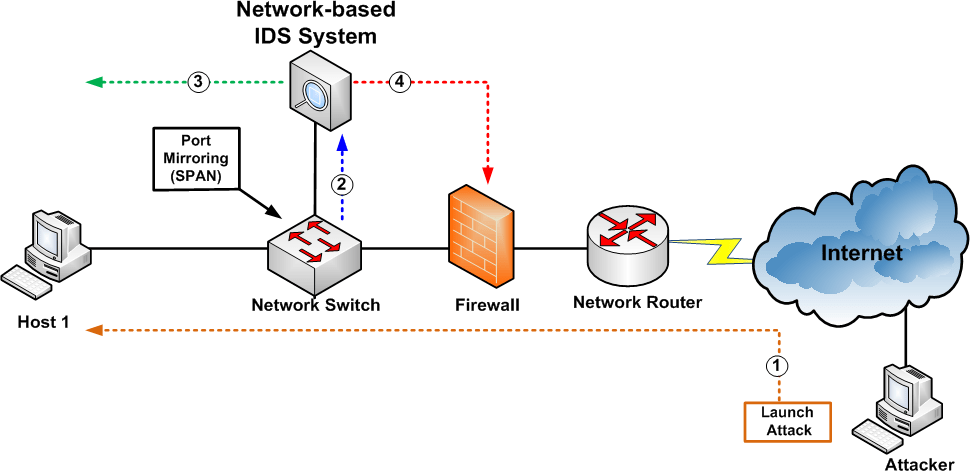


Fig 2.6 Architecture of NIDS

## 2.13 Hardware Requirements

* Pentium IV or higher, (PIV-300GHz recommended)
* 256 MB RAM
* 1 Gb hard free drive space
* P-IV and Higher versions Microprocessor
* 1.44 MB FDD Disk Drives
* HP CD-Rom

**2.14 Software Requirements**

* Operating System: Windows 10 and Linux
* Wireshark
* Flowtbag
* Hping
* Tshark
* Python 3 and above versions
* Anaconda
* Jupyter Notebook or Spyder

**2.14.1 Machine Learning Libraries and Tools**

* + Gym
  + TensorFlow
  + Sckit
  + Keras
  + Numpy
  + Pandas
  + Matplot Libraries

## 2.15 Interfaces

## 2.15.1 User Interface

The user will be capable of connecting to the service in order to administer it. This will include updating parameters, starting and stopping the service. This can be graphical or text based. The user must also be capable of training the network provided they have the data sets in the correct place for the system to find them. This can again be text or graphical based.

## 2.15.2 Software Interface

The system will interface with the packet capture library PCap to enable raw packet capture and the raw sockets API. The system will be implemented in the Python programming language and so its standard API will be used.

## 2.15.3 Communication Interface

An Ethernet card must be installed on the machine that is running the software. With the addition of PCap the system will eavesdrop on the network in order to analyze its activity.

# Verification

## 3.1 Introduction

Verification is process of ensuring that software function as per user needs.

Accuracy: Deﬁned as the percentage of correctly classiﬁed records over the total number of records. This is the percentage of correctly classiﬁed item over total item. In our code we have done like this

accuracy =(tp + tn)/totalrecords

• tp=true positive

• tn=true negatives

Precision is the ratio of true positive value and the combination of true positive value and false positive.

precision = tp / (tp + fp)

Sensitivity: Its also called recall. This is the ration of true positive and sum of true positive and false negative.

Specificity: This is the ration of true negative and sum of true negative and false positive.

speciﬁcity = tn / (tn + fp)

## 3.2 Types of software Verification:

1. White Box Testing It is a test case design philosophy that uses the control structure described as part of component level design to derive test cases.

• Basis Path Testing- Basis path method enables designer to derive a logical complexity measure of a procedural design and use this measure as a guide for deﬁning a basis set of execution paths.

• Control Structure Testing- It tests control structures of program.

1. Black Box Testing Black box testing enables the software designer to derive sets of input conditions that will fully exercise all functional requirements of a program.

• Graph Based Testing Method

• Equivalence Partitioning

• Boundary Value Analysis

• Orthogonal Array Testing

1. Integration Testing

Integration Testing is a systematic technique for constructing software architecture while at the same time conducting tests to uncover errors associated with interfacing.

1. Regression Testing

Each time a new model is added as part of integration testing, the software changes. At this time regression testing is applied.

## 3.3 Testing on KDD Dataset of Network Intrusion

We have trained our algorithms on the dataset which we have captured from Wireshark in which we have done DoS attack using hping3 and then we extracted 32 features using Flowtbag tool and hence we made a dataset which is equal to KDD dataset in terms of number of features. So now its time to test our algorithms on KDD dataset but our model is trained with our dataset only.

## 3.3.1 KDDCUP Dataset

Computer network intrusion detection, this database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment. Intelligent Intrusion detection systems can only build if there is an availability of a valid data set. A data set with the sizable amount of data which mimics the real-time can only help to train and test an intrusion detection system.

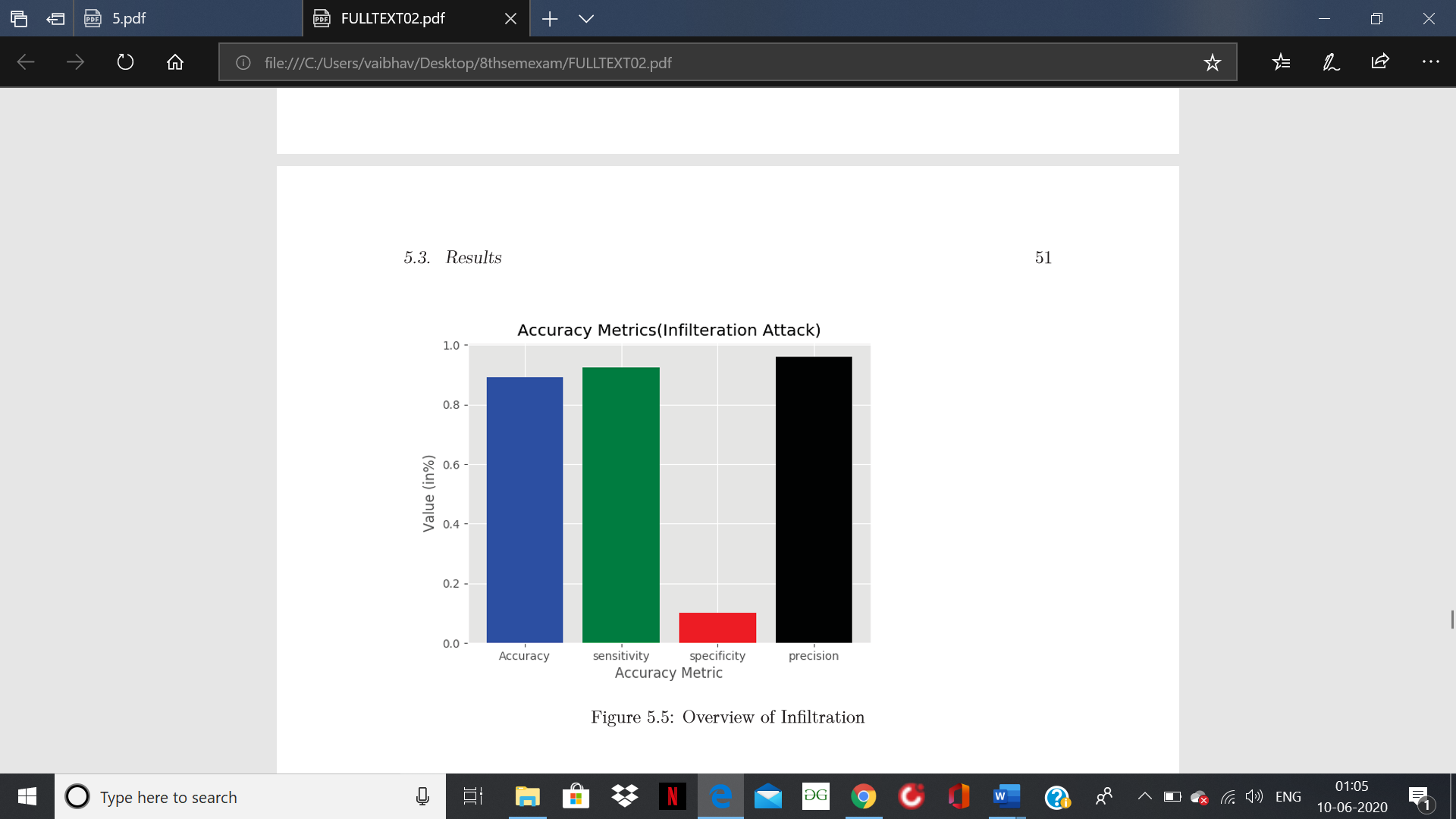


Fig 3.1 Accuracy Metric for KDD

Result using KDD-Dataset with Machine Learning-Model ( Random Forest)

• Accuracy = 89.245%

• Precision=95.93%

• Sensitivity=92.34%

• Speciﬁcity=10.1%

In the result above is the confusion matrix including the value of true posititve and true false with true false and true positive. Then accuracy of predicting the attack with its precision and its recall value and F\_score.

## 3.4 Testing on NSL-KDD Dataset

## 3.4.1 NSL-KDD

NSL-KDD is a data set suggested to solve some of the inherent problems of the KDD'99 data set. Furthermore, the number of records in the NSL-KDD train and test sets are reasonable. This advantage makes it affordable to run the experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research work will be consistent and comparable. The NSL-KDD data set has the following advantages over the original KDD data set:

* It does not include redundant records in the train set, so the classifiers will not be biased towards more frequent records.
* There is no duplicate records in the proposed test sets; therefore, the performance of the learners are not biased by the methods which have better detection rates on the frequent records.
* The number of selected records from each difficultylevel group is inversely proportional to the percentage of records in the original KDD data set. As a result, the classification rates of distinct machine learning methods vary in a wider range, which makes it more efficient to have an accurate evaluation of different learning techniques.
* The number of records in the train and test sets are reasonable, which makes it affordable to run the experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research works will be consistent and comparable.

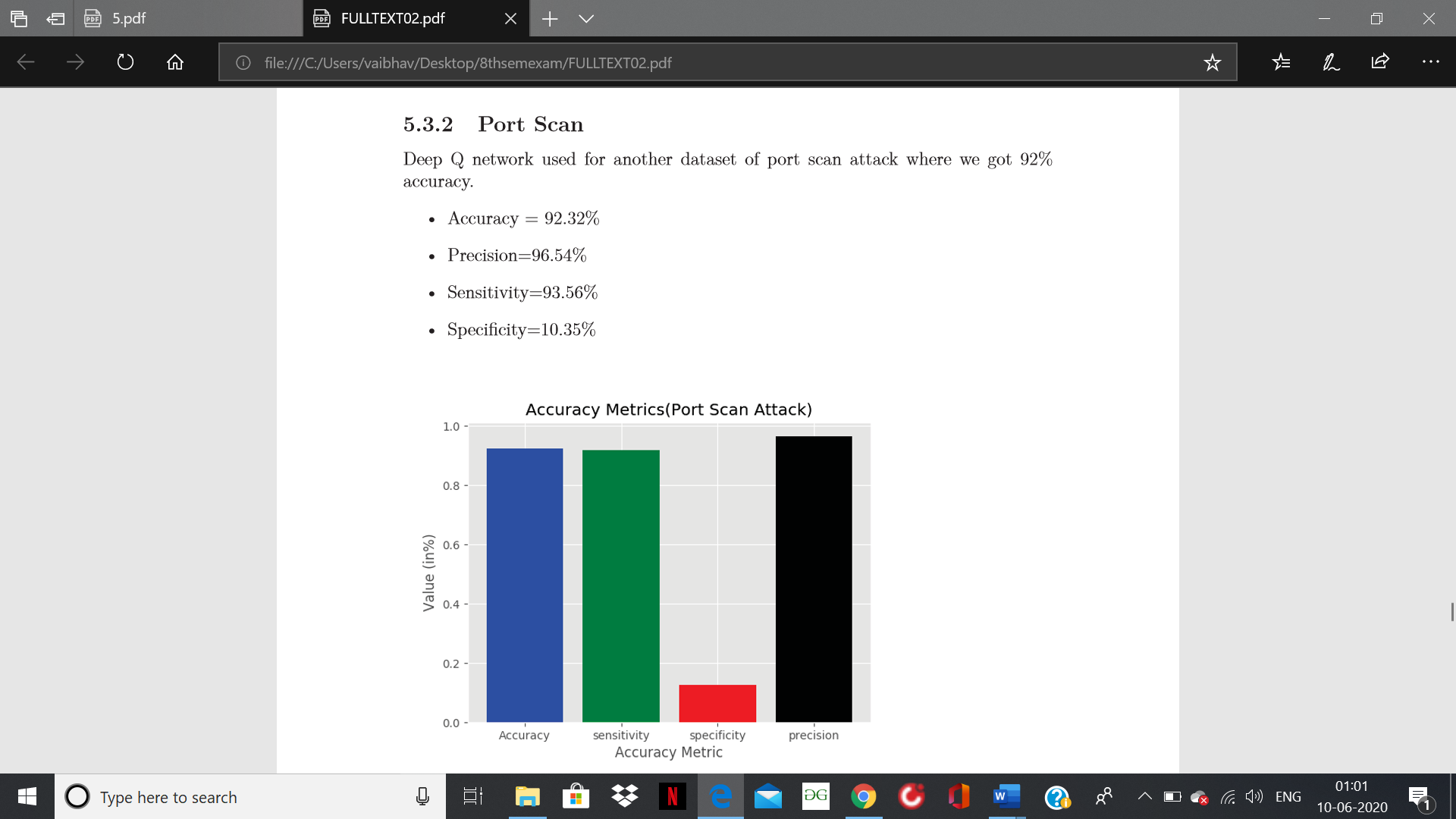


Fig 3.2 Accuracy Metric for NSL-KDD

Result using NSL-KDDCup Dataset using ANN

• Accuracy = 92.32%

• Precision=96.54%

• Sensitivity=93.56%

• Speciﬁcity=10.35%

## 3.5 Verification on CICIDS 2017 Dataset

## 3.5.1 CICIDS Dataset

KDDcup99 is 20 years older dataset and NSL-KDD is 10 years older. Network packet attributes have changed and are upgraded. KDDcup99 has only 41 attributes. CICIDS 2017 dataset is used which has 85 attributes. The size and entry of the dataset are 20 times more that KDDcup99 or NSL-KDD. Moreover, tuning the hyperparameters could improve the rate of accuracy because the prediction of the model relies on the dataset.

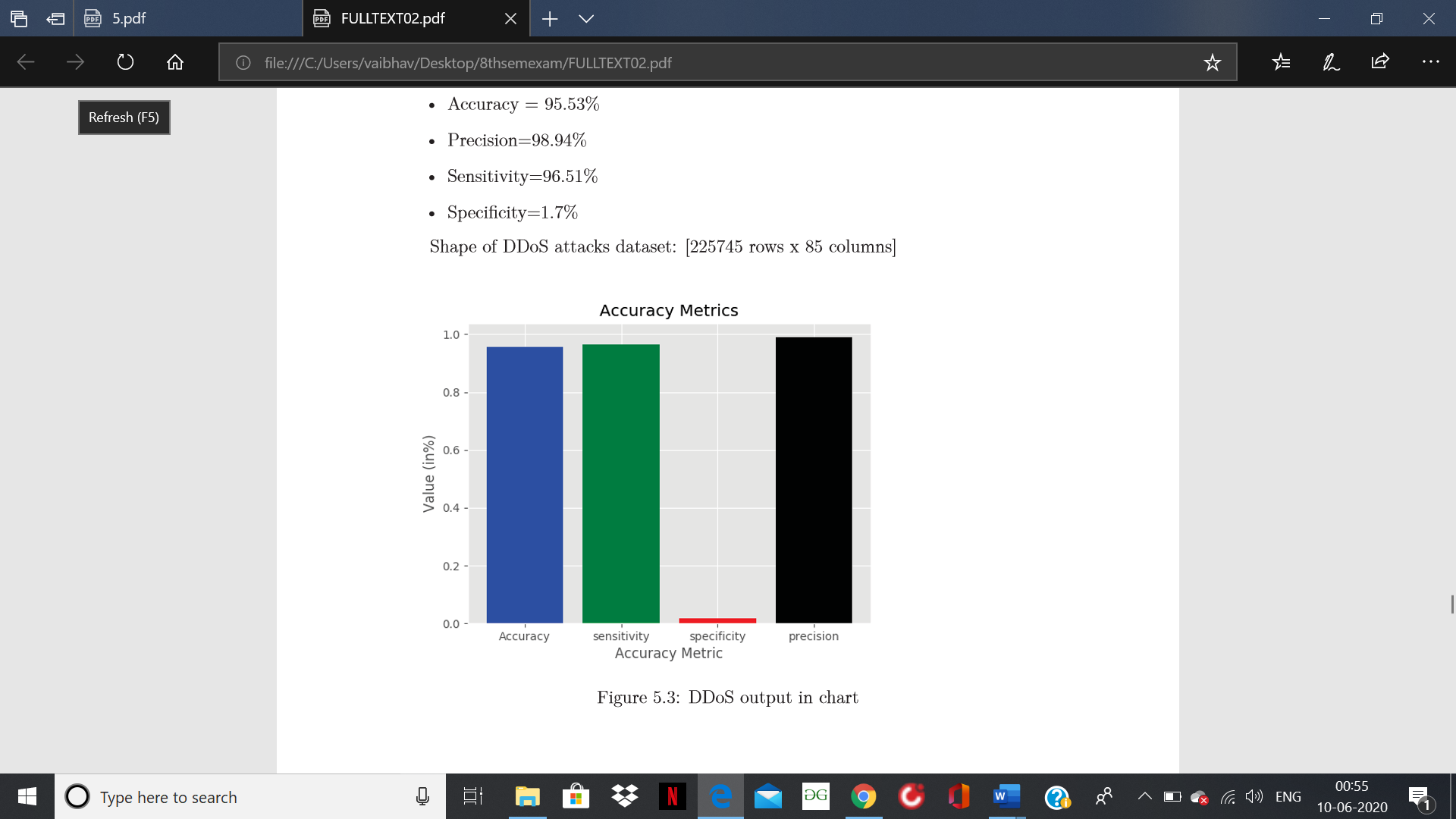


Fig 3.3 Accuracy Metric for CICIDS

Result using Deep Q learning Model on CICIDS Dataset

• Accuracy = 95.53%

• Precision=98.94%

• Sensitivity=96.51%

• Speciﬁcity=1.7%

We have evaluated three datasets. We got most consistent precision and sensitivity performance. Using DQN our highest outcome percentage is almost 99% precision and sensitivity is more than 96%. Our result reﬂects thatDQN is best suitable for unknown attacks detection compare to other ML algorithm and minimise the false alarm rate. We have used DQN for detection unknown attacks. While testing our model with new dataset, the input is random and unknown to the model. Using DQN we have got the maximum accuracy 95% and precision of 98%. We have shown the comparison with other algorithm and got acceptable result according to other algorithm.

# Conclusions

Prevention of security breaches completely using the existing security technologies is unrealistic. As a result, intrusion detection is an important component in network security. IDS offers the potential advantages of reducing the man power needed in monitoring, increasing detection efficiency, providing data that would otherwise not be available, helping the information security community learn about new vulnerabilities and providing legal evidence. In this system we proposed a new intrusion detection system in which we have refined our results by combining different artificial neural networks having 3-4 hidden layers to get the best accuracy for predicting the attacks which are already detected. We trained our model first with CNN and passed the result in RNN then GRU through LSTM and at last passed the refined dataset to Deep Neural Network with 5 hidden layers. We achieved better accuracy after this combined model than using a single model for the detection but still we faces the problem of unknown attacks detection and older dataset problem so we used reinforcement learning which is model-free approach of training the machine on new datasets and can also predict unknown attacks. We also examined that the Deep Q Network is able to classify the attack types with a highest accuracy of 97%. In opposition, there are other algorithms we have shown that has high accuracy level. Moreover, tuning the hyperparameters could improve the rate of accuracy because the prediction of the model relies on the dataset. The Deep Q network algorithm provides a very good accuracy compared with other models with a low featured dataset. For all the research and experimentation of NIDS with machine learning algorithm approach, these datasets are used for Q network and hence we are comparing our result with these available models on the internet. In application, this model could be implemented for the detection of any unknown attack and unknown data type. One of the most important concern to consider is that to make sure the data is clean. From the results and analysis, Deep Reinforcement Learning algorithm with NIDS would be a fair consideration for the enhancement of the improved accuracy of intrusion detection in the network with high performance metrics. On the other hand, prior to the known attacks, we can also detect the unknown attacks. Deep Reinforcement Learning techniques help in acquiring the best possible outcome with better accuracy i.e. the maximum reward. Advantage of Reinforcement Learning technique is doesn’t need any prior knowledge or past experience to take any decision, when the possibility of occurrence of anomaly packets is high; when there is an unknown attack, it also detects random data.

# References

[1] A. M. Chandrasekhar, K. Raghuveer “ Intrusion Detection Techniques by using K-means, Fuzzy Neural network and SVM classifier”, ICCCI2013, Jan. 04-06, 2013, Coimbatore, INDIA.

[2] Sumaiya Thaseen, Ch. Aswani Kumar: “analysis of supervised Tree based classifiers for intrusion detection system”, In Proceedings of the 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering (PRIME) in February 21-22.

[3] Nidhi Srivastav, Rama Krishna Challa “Novel Intrusion Detection System integrating Layered Framework with Neural Network” 978-1-4673- 4529, in IEEE, 2012.

[4] Zhao Y ongli, Zhang Yungui, Tong Weiming, Chen Hongzhi, “an improved feature selection algorithm based on MAHALANOBIS distance for network intrusion detection”, international conference on sensor network security technology and privacy communication system (SNS & PCS), 2013.

[5] . Fengli Zhang, Dan Wang “an effective feature selection approach for network intrusion detection” IEEE Eighth international conference on networking, architecture and storage, 2013.

[6] Y. Li, B. Fang, Y. Chen, and L. Guo, “a lightweight intrusion detection model based on feature selection and maximum entropy model,” Proc. International Conference on Communication Technology (ICCT '06), IEEE Press, Nov. 2006, pp.14, doi: 10.1109/ICCT.2006.341771.

[7] . Preecha Somwang, Woraphon Lilakiatsakun: “intrusion detection technique by using fuzzy ART on computer network security” 2011, 978-1- 4577-2119, in IEEE. [8] Bauer, D.S., Eichelman II., F.R., Herrera, R.M. and Irgon, A.E. (1989) 'Intrusion detection: an application of expert systems to computer security', Proceedings of the International Carnahan Conference on Security Technology, pp. 97-100.

[9] A novel intrusion detection system based on hierarchical clustering and support vector machines SJ Horng, MY Su, YH Chen, TW Kao, RJ Chen, JL Lai… - Expert systems with Applications, 2011.

[10] Richard, K. and V. Giovanni (2002). Intrusion Detection: A Brief History and Overview. Security and Privacy: 27-30.

[11] Adebayo, O.A., S.O. Falaki, O.S. Adewale and Boniface K. Alese, 2008.. Network Intrusion Detection Based On Rough Set and K-Nearest Neighbour.

[12] Abadeh, M. S., Mohamadi, H. and Habibi, J. “Design and analysis of genetic fuzzy systems for intrusion detection in computer networks”, Expert Systems with Applications, 38(6), pp. 7067–7075, 2011.

[13] Intrusion detection using fuzzy association rules A Tajbakhsh, M Rahmati, A Mirzaei - Applied Soft Computing, 2009