

# **A NOVEL ARCHITECTURE FOR DETECTION OF VARIOUS ATTACKS IN CLOUD COMPUTING ENVIRONMENT**

Thesis Submitted for the Award of the Degree of

**DOCTOR OF PHILOSOPHY**

in

**Computer Science and Engineering**

By

**Pooja Rana**

**Registration Number: 41800914**

**Supervised By**

**Dr. Isha Batra (17451)**

**Computer Science and Engineering**

**(Professor)**

**Lovely Professional University**

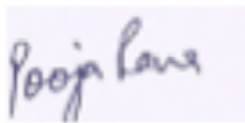


**LOVELY PROFESSIONAL UNIVERSITY, PUNJAB  
2025**

## DECLARATION

---

I, hereby declared that the presented work in the thesis entitled “**A Novel Architecture for Detection of Various Attacks in Cloud Computing Environment**” in fulfilment of degree of Doctor of Philosophy (Ph.D.) is outcome of research work carried out by me under the supervision of Dr. Isha Batra, working as Professor, in the Computer Science & Engineering of Lovely Professional University, Punjab, India. In keeping with general practice of reporting scientific observations, due acknowledgements have been made whenever work described here has been based on findings of other investigator. This work has not been submitted in part or full to any other University or Institute for the award of any degree.



(Signature of Scholar)

Name of the scholar: Pooja Rana

Registration No.: 41800914

Department/school: Computer Science & Engineering

Lovely Professional University

Punjab, India

## CERTIFICATE

---

This is to certify that the work reported in the Ph. D. thesis entitled “**A Novel Architecture or Detection of Various Attacks in Cloud Computing Environment**” submitted in fulfillment of the requirement for the award of degree of **Doctor of Philosophy (Ph.D.)** in the Computer Science & Engineering , is a research work carried out by Pooja Rana, 41800914, is bonafide record of his/her original work carried out under my supervision and that no part of thesis has been submitted for any other degree, diploma or equivalent course.



**(Signature of Supervisor)**

Name of supervisor: Dr. Isha Batra

Designation: Professor

Department/school: Computer Science & Engineering

University: Lovely Professional University

## Abstract

---

Cloud Computing is a platform that virtually shared servers provide software, platforms, infrastructure, policies and other functionalities. This technology is seen as a solution for minimizing costs and complexities for users. Its increasing popularity stems from benefits like on-demand service access, flexible resource management, robust fault tolerance and scalability. Cloud environments with their distributed nature can become appealing targets for intruders. To ensure reliable and secure services within these environments, implementing an Intrusion Detection System (IDS) is a highly effective approach. An IDS helps to identify the attacks that could compromise the security of cloud infrastructures. Conventional security measures, like firewalls, are insufficient to tackle the increasingly complex and dynamic security issues faced in cloud computing environments. These methods primarily focus on filtering unauthorized access but fall short in identifying sophisticated and dynamic threats. To overcome these limitations, implementing a robust system like an Intrusion Detection System (IDS) is essential. An IDS effectively monitors network traffic and system activities, detecting and responding to potential attacks in real-time. Its advanced capabilities make it a critical component in safeguarding cloud infrastructures against emerging security threats.

An Intrusion Detection System (IDS) is an automated solution designed to detect inappropriate events, such as intrusion attacks, occurring within computer systems. An IDS equipped with effective countermeasures plays a vital role in identifying and addressing such threats. The main goal is to identify different forms of malicious network activity and keep track of system behaviour, tasks that conventional security methods often struggle to handle effectively.

The necessity for robust Intrusion Detection Systems (IDS) has grown exponentially as the attacks pose significant risks to data integrity, availability, and confidentiality, resulting in substantial financial losses, reputational damage and operational

disruptions of the cloud. Consequently, there has been a significant surge in research dedicated to developing effective IDS to counteract attacks. This report introduces a novel architecture that combines the modified firefly algorithm with a hybrid classifier to enhance IDS performance.

The swift expansion of internet connectivity and digitalization has resulted in a significant rise in cyberattacks. This has driven the need for advanced IDS capable of real-time identification and mitigation of various cyber threats. The primary goal of an IDS is to detect unauthorized access and malicious activities within a network, thereby protecting sensitive data and ensuring the continuous availability of network services.

The proposed architecture of research integrates the modified firefly algorithm for feature selection with a hybrid classifier for detection of attacks. Hybridized Firefly Algorithm with Decision Tree Algorithm is used for feature selection. This algorithm was compared with Particle Swarm Optimization (PSO) and Genetic Algorithm (GA). The hybrid classifier, which combines the strengths of Neural Network with Decision Tree utilized to enhance detection capabilities further.

Simulations were conducted in CloudSim, a robust platform for modelling and simulating cloud computing environments and services. This realistic approach allowed for testing the robustness and scalability of the IDS under various conditions. The proposed algorithm was also tested on simulated data generated within the CloudSim environment, in addition to validation over the CSE-CIC-IDS 2018 dataset. This two-step validation method verifies both the accuracy and efficiency of the modified firefly algorithm paired with the hybrid classifier in controlled datasets. Additionally, it highlights its practical utility in real-world, cloud-based applications. The simulation outcomes emphasize the algorithm's ability to detect and counter cyber threats effectively, showcasing its potential to improve network security across various computing environments.

The increasing sophistication and frequency of cyber threats demand the development of robust and effective Intrusion Detection Systems. The proposed architecture, which

combines the proposed feature selection with a hybrid classifier, offers a novel approach to improving IDS performance. Utilizing the extensive CSE-CIC-IDS 2018 dataset and performing simulations in CloudSim, notable enhancements in classification accuracy, precision, recall, and overall performance have been achieved. This study not only contributes to the advancement of IDS research but also offers a practical and scalable approach to safeguarding digital infrastructures. As cyber threats grow increasingly sophisticated, continued innovation and research in IDS will be vital for maintaining the security and reliability of network systems, thereby protecting critical data and services from malicious activities.

At 150,000 samples, the proposed architecture achieved a precision of 0.9658, recall of 0.9681, F-Measure of 0.9669, and accuracy of 94.99%. These metrics consistently showed superior performance across various sample sizes compared to other architectures. For instance, in terms of precision, PSO combined with a hybrid classifier achieved 0.95188 at 100,000 samples, whereas the proposed architecture attained a higher precision of 0.9658. Similarly, recall for PSO and GA combined with a hybrid classifier at 100,000 samples was 0.83272 and 0.85425, respectively, while the proposed architecture achieved 0.9884, highlighting its effectiveness in identifying actual positives.

At 1,00,000 samples of simulated dataset, the precision was 0.9658, accuracy 91.5%, recall 0.988 and F-Measure 0.977. These results indicate the proposed architecture's strong ability to accurately identify true positives while minimizing false positives. The high recall rate signifies the proposed algorithm's robust capacity to detect nearly all true attack instances, thus significantly reducing false negatives. This capability to detect a high number of true positives is crucial in an IDS context, as missing actual threats can lead to severe security breaches with high F-Measure demonstrates that the proposed architecture not only identifies a large number of true positives but also does so with a low rate of false positives, ensuring a balanced and effective detection capability.

This high recall rate signifies the proposed algorithm's robust capacity to detect nearly all true attack instances, thus significantly reducing false negatives. This capability to detect a high number of true positives is crucial in an IDS context, as missing actual threats can lead to severe security breaches.

The successful application of proposed architecture in both controlled datasets and simulated real-world environments further highlights its robustness and practical relevance in contemporary cybersecurity practices. The proposed architecture not only advances IDS research but also provides a practical and scalable solution for protecting digital infrastructures. This makes it an invaluable tool in the fight against cyber threats, ensuring that IDS can effectively identify and mitigate potential attacks, thus protecting network integrity and maintaining the availability of essential services. The successful application of this architecture in both controlled datasets and simulated real-world environments further highlights its robustness and practical relevance in contemporary cybersecurity practices.

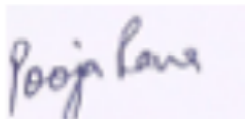
## Acknowledgement

---

I thank the Lord Almighty, the source of wisdom, who has provided me the with ability, the resources, the opportunity, and his kind support. I would like to acknowledge and give my warmth thanks to my supervisor Dr. Isha Batra who made this work. Her advice guided through all stages of my research work. I am deeply indebted to her for shaping my research path by guiding me with her extensive knowledge and discussions. I express my deep gratitude to my supervisor for her motivation, persistent encouragement, and keen involvement that persuaded me to complete this research. I am highly thankful to her. Without her guidance, help, and patience, I would never have accomplished this thesis work. She has been an incredible mentor to me.

I wish to extend my thanks to all faculty members of the Ph.D. (CSE) for attending my seminars and for their insightful comments and constructive suggestions to improve the quality of this research work.

I thank all the faculty members of the CSE department, Lovely Professional University, for rendering excellent cooperation. I will remain indebted to my family for providing me the confidence and comfort to undertake this thesis. I find no words to thank my husband and parents for their dedicated support and prayers offered to complete my research work.



Pooja Rana



## Table of Contents

---

DECLARATION.....	<b>Error! Bookmark not defined.</b>
CERTIFICATE .....	<b>Error! Bookmark not defined.</b>
Abstract .....	iii
Acknowledgement.....	vii
Table of Contents .....	viii
List of Tables.....	xi
List of Figures .....	xiii
List of Abbreviations.....	xv
CHAPTER 1: INTRODUCTION .....	1
1.1 Cloud Computing .....	1
1.2 Characteristics of Cloud Computing .....	2
1.3 Services of Cloud Computing .....	3
1.4 Deployment Models of Cloud Computing .....	4
1.5 Intrusion Detection System (IDS) .....	6
1.5.1 Types of IDS Based on Monitoring System.....	8
1.5.2 Detection Techniques of IDS .....	9
1.5.3 IDS Related to Cloud Computing .....	12
1.5.4 Examples of Real-Time Security Breaches Related to Cloud Computing.....	13
1.6 Attacks Affecting Network .....	14
1.7 Machine Learning.....	18
1.8 Swarm Intelligence.....	17
1.9 Research Motivation.....	19

1.10 Organization of Thesis .....	24
1.11 Summary .....	25
CHAPTER 2: LITERATURE REVIEW .....	26
2.1 Cloud Computing .....	26
2.2 Feature Selection .....	28
2.3 Attacks Affecting Cloud Computing.....	32
2.4 Similarity Comparison of Benchmark Dataset with Existing Dataset .....	33
2.5 Intrusion Detection System .....	36
2.6 Comparative Analysis .....	45
2.7 Problem Statement .....	55
2.8 Research Gaps .....	57
2.9 Summary .....	58
CHAPTER 3: RESEARCH METHODOLOGY .....	59
3.1 Research Objectives .....	59
3.2 Research Methodology of Research Work.....	59
3.3 Key Challenges While Implementing the IDS .....	68
3.4 Summary .....	70
CHAPTER 4: PREPROCESSING AND FEATURE EXTRACTION ON SIMULATED DATASET AND CSE CIC IDS 2018 DATASET .....	71
4.1 Preprocessing.....	71
4.1.1 Generation and Preprocessing of Simulated Dataset.....	71
4.1.1.1 Generation of Simulated Dataset by Simulating Cloud Computing Environment.....	84
4.1.1.2 Mapping of Simulated Dataset with Actual Dataset.....	95
4.1.1.3 Parameters Analysis for Analyzing the Impact of Attacks .....	100

4.1.1.4 Preprocessing on Simulated Dataset .....	101
4.1.2 Preprocessing of CSE CIC IDS 2018 Dataset.....	103
4.2 Feature Selection Technique .....	104
4.2.1 Types of Feature Selection Techniques.....	106
4.2.2 Proposed Feature Selection Algorithm.....	107
4.3 Summary .....	113
CHAPTER 5: ARCHITECTURE FOR DETECTION OF ATTACKS IN CLOUD COMPUTING ENVIRONMENT .....	101
5.1 Proposed Architecture for Detection of Attacks in Cloud Computing Environment Using Optimized Classifier .....	116
5.2 Computation Complexity Analysis .....	127
5.3 Summary .....	127
CHAPTER 6 RESULTS AND DISCUSSIONS .....	128
6.1 Implementation Details .....	128
6.2 Performance Metrics and Hypothesis Testing .....	129
6.2.1 Performance Metrics .....	129
6.2.2 Hypothesis Testing .....	129
6.3 Feature Selection Analysis .....	130
6.4 Analysis Using CSE CIC IDS 2018 Dataset.....	139
6.5 Analysis Using Simulated Dataset .....	151
6.6 Summary .....	160
CHAPTER 7: CONCLUSION AND FUTURE SCOPE .....	161
7.1 Conclusion.....	161
7.2 Future Scope.....	166
References .....	168

List of Publications.....	178
---------------------------	-----

## List of Tables

---

Table 1.1 Comparison of Detection Techniques of Intrusion Detection Syste .....	11
Table 1.2 Comparison of Active Attack and Passive Attack .....	14
Table 2.1 Comparison of related work to similarity measure.....	45
Table 2.2 Comparative Analysis of Literature Review Related to IDS .....	47
Table 3.1 Comparison of IDS Datasets .....	61
Table 4.1 Attributes of VMs.....	76
Table 4.2 Cloud Sim Configurations Used for Cloud Simulation.....	76
Table 4.3 Cloudsim Node Communication Simulation Algorithm.....	78
Table 4.4 Result of Similarity Calculation .....	93
Table 4.5 Parameters Analyzed Before Attack .....	96
Table 4.6 Parameters Analyzed After Attack.....	97
Table 4.7Psuedocode for Hybridized Firefly Algorithm with Decision TreeAlgorithm	100
Table 4.8 Parameter Analysis Before and After Implementation of IDS .....	110
Table 5.1 Psuedocode for Hybridized Neural Network with Decision Tree.....	123
Table 6.1Implementation Details .....	128
Table 6.2 Precision Comparison of Feature Selection Algorithms .....	130
Table 6.3 Accuracy Comparison of Feature Selection Algorithms.....	133
Table 6.4 Recall Comparison of Feature Selection Algorithms.....	135
Table 6.5F-Measure Comparison of Feature Selection Algorithms.....	137
Table 6.6 Precision Comparison for CSE CIC IDS 2018 Dataset .....	140
Table 6.7 Accuracy Comparison for CSE CIC IDS 2018 Dataset.....	142

Table 6.8 Recall Comparison for CSE CIC IDS 2018 Dataset .....	145
Table 6.9 F-Measure Comparison for CSE CIC IDS 2018 Dataset .....	148
Table 6.10 Precision Comparison for Simulated Dataset.....	151
Table 6.11 Accuracy Comparison for Simulated Dataset .....	153
Table 6.12 Recall Comparison for Simulated Dataset .....	155
Table 6.13 F-Measure Comparison for Simulated Dataset.....	158

## List of Figures

---

Figure 1.1 Characteristics of Cloud Computing .....	2
Figure 1.2 Services of Cloud Computing .....	3
Figure 1.3 Deployment Models of Cloud Computing .....	5
Figure 1.4 CIA Related to Security.....	7
Figure 1.5 Types of IDS Based on Monitoring Environment.....	8
Figure 1.6 Signature-Based IDS .....	10
Figure 1.7 Pictorial Representation of DoS attack and DDoS attack .....	16
Figure 1.8 Pictorial Representation of Botnet attack .....	18
Figure 1.9 Types of Machine Learning.....	19
Figure 1.10 Types of Swarm Intelligence.....	21
Figure 3.1 Research Methodology of Research Work.....	60
Figure 4.1 Calculation of similarity measures and threshold .....	92
Figure 4.2 Comparison of CPU Utilization .....	97
Figure 4.3 Comparison of Bandwidth Utilization.....	98
Figure 4.4 Comparison of CPU Load .....	98
Figure 4.5 Flowchart of Proposed Feature Selection Algorithm .....	111
Figure 5.1 Proposed Architecture for Detection of Attacks in Cloud Computing Environment.....	115
Figure 6.1 Comparison of Accuracy for Feature Selection Algorithms .....	132
Figure 6.2 Comparison of Precision for Feature Selection Algorithms.....	134
Figure 6.3 Comparison of Recall for Feature Selection Algorithms .....	136
Figure 6.4 Comparison of F-Measure for Feature Selection Algorithms .....	138

Figure 6.4 Comparison of Precision for CSE CIC IDS 2018 Dataset .....	141
Figure 6.5 Comparison of Accuracy for CSE CIC IDS 2018 Dataset.....	144
Figure 6.6 Comparison of Recall for CSE CIC IDS 2018 Dataset.....	147
Figure 6.7 Comparison of F-Measure for CSE CIC IDS 2018 Dataset.....	149
Figure 6.8 Comparison of Precision for Simulated Dataset .....	152
Figure 6.9 Comparison of Accuracy for Simulated Dataset.....	154
Figure 6.10 Comparison of Recall for Simulated Dataset .....	157
Figure 6.11 Comparison of F-Measure for Simulated Dataset .....	159



## List of Abbreviations

---

Acronym	Term
CC	Cloud Computing
CSPs	Cloud Service Providers
SaaS	Software as a Service
IaaS	Information as a Service
PaaS	Platform as a Service
IDS	Intrusion Detection System
DM	Data Mining
ML	Machine Learning
NN	Neural Network
HIDS	Host-Based Intrusion Detection System
NIDS	Network-Based Intrusion Detection System
DIDS	Distributed Intrusion Detection System
DT	Decision Tree
SVM	Support Vector Machine
FS	Feature Selection
FFA	Firefly Algorithm
AI	Artificial Intelligence
ACO	Ant Colony Optimization

ABC	Artificial Bee Colony
PSO	Particle Swarm Optimization
FFA	Firefly Algorithm
GA	Genetic Algorithm
DT	Decision Tree
SVM	Support Vector Machine
KNN	K-Nearest Neighbor
VM	Virtual Machine

# CHAPTER 1 INTRODUCTION

---

The chapter offers an in-depth overview of the cloud computing environment, addressing the associated security concerns. It also examines various network attacks that threaten cloud computing security. The intrusion detection systems used for detection of attacks are also discussed in detail in this chapter.

## 1.1 Cloud Computing

Today Cloud Computing has become highly valued as it offers on-demand and scalable services that meet user needs along with reducing costs and complexities. Resources can be deployed more easily with minimal management and reduced interaction with different service providers by using cloud computing. This improves the accessibility of the resources and allows the infrastructure to be utilized on a Pay-per-Use-On-Demand basis which results in cost savings [S. Kumar and R. H. Goudar,2012].

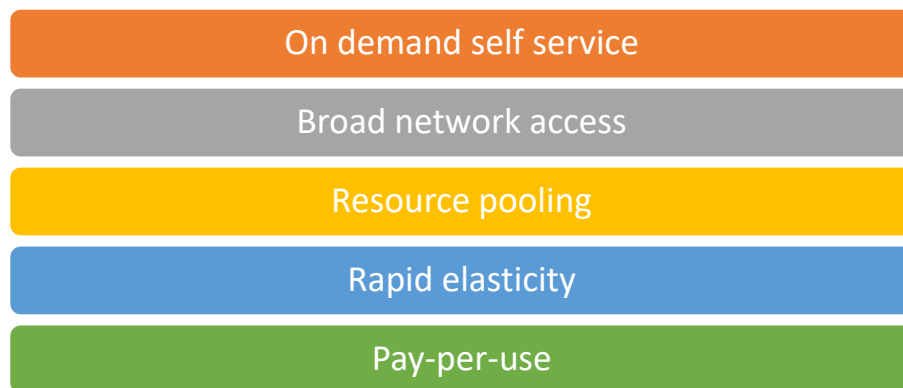
The cloud environment can support a large number of users because it is scalable. The key benefits of switching to cloud computing include lower costs, less reliance on staff, resilient scalability and others. Cloud computing allows for the dynamic easing of congestion or addition of capabilities without investing in additional hardware, employing more employees or obtaining software licences. It increases IT potential.

Cloud Computing has revolutionised information processing by providing a technology platform that is affordable, efficient and scalable. From an administrative standpoint, cloud computing offers greater storage and processing capacity at a lower cost. To fully unlock the potential of cloud computing, CSPs must offer flexible service delivery that accommodates diverse consumer needs, all

while keeping users abstracted from the underlying infrastructure [A. Beloglazov et al., 2012].

## 1.2 Characteristics of Cloud Computing

The cloud computing model is defined by five key characteristics: broad network access, rapid elasticity, resource pooling, on-demand self-service, and measured service. These features highlight the distinct advantages of cloud computing over traditional computing models [M. I. Alam et al., 2015]. Figure 1.1 provides a visual representation of these characteristics.



**Figure No.1.1 Characteristics of Cloud Computing**

**1.On-demand self-service:** Cloud computing services allow users for provisioning, monitoring and managing resources independently without the requirement of human administrators.

**2.Broad network access:** Cloud services are delivered over wide-area networks and can be accessed through various heterogeneous devices.

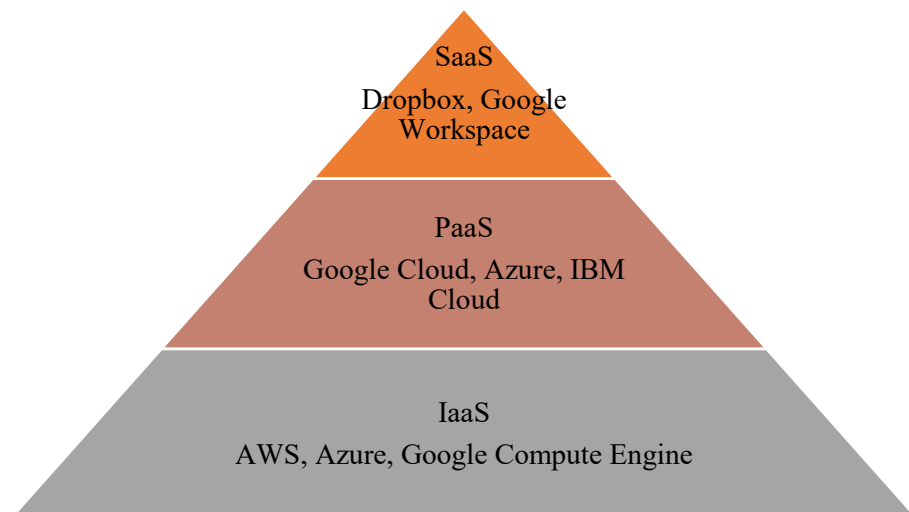
**3.Rapid elasticity:** These services provide the capability to swiftly scale resources up or down based on user demand.

**4.Resource pooling:** Resources such as networks, servers, storage, applications, and services are shared among multiple users and applications. This allows different clients to utilize the same physical resources efficiently.

**5.Measured service:** Resource usage is tracked for each application and user, providing both the service provider and the user with detailed records of consumed resources. This is crucial for purposes like billing and ensuring optimal resource utilization.

### 1.3 Services of Cloud Computing

Cloud computing is comprised of three service models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [M. Kavis, 2014]. Figure 1.2 depicts the services provided by these models.



**Figure No.1.2 Services of Cloud Computing**

**1.Software as a Service (SaaS):** SaaS is a cloud-based model for delivering a wide range of services and applications. Users can access and use them online, instead

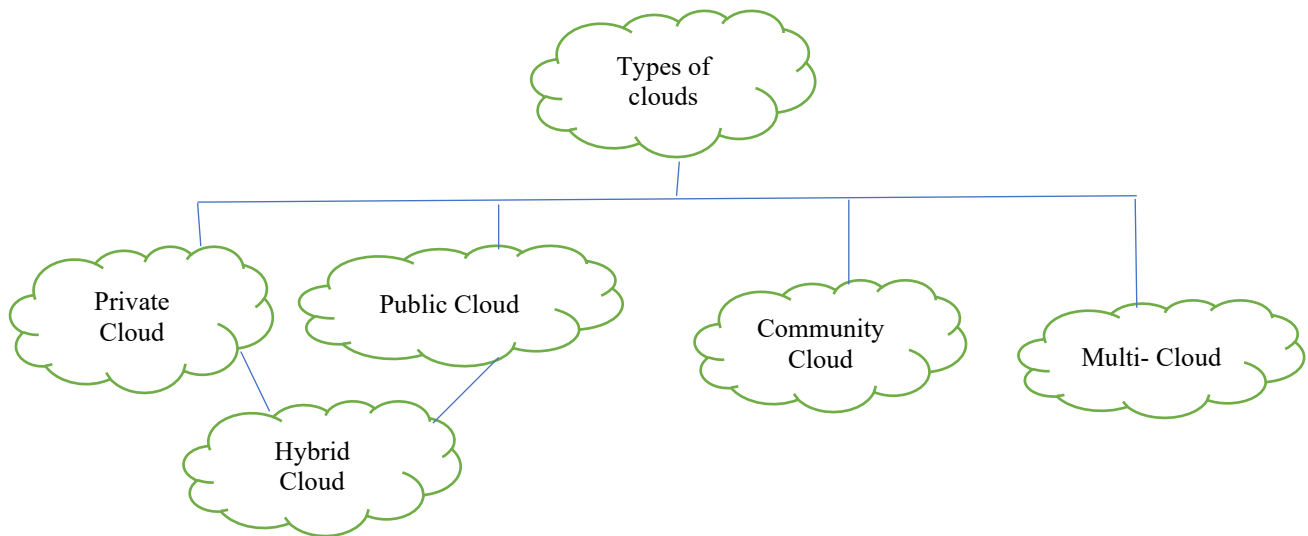
of installing and managing software on local devices and this simplifies software and hardware management. This approach eliminates the need for installing and running applications on local devices which reduces the costs related to the software. This offers a complete software solution that is available on the subscription basis from a CSP.

**2.Platform as a Service (PaaS):** PaaS helps the developers by providing with a platform and environment for building applications and services via the cloud. PaaS services are hosted in the cloud and accessed by users through their web browsers. CSPs hosts the necessary hardware and software on its infrastructure which makes the users free from the need to install and manage in-house hardware and software for development or running of new applications. This helps the application development and deployment to occur independently of the underlying hardware.

**3. Infrastructure as a Service (IaaS):** IaaS is a cloud service model which provides outsourced computer infrastructure for supporting various operations. It offers enterprises infrastructure components like networking equipment, devices, databases and web servers. Users using IaaS services mostly pay on a per-user basis and pricing often structured by the hour, week or month. Some CSPs may charge on the amount of virtual machine space utilized. It provides services related to the essential operating systems, security, networking and servers which are required for development of the applications. It also provides deploying services.

#### **1.4 Deployment Models of Cloud Computing:**

The four types of cloud deployment models include public cloud, private cloud, hybrid cloud, and community cloud [S. Carlin and K. Curran, 2013]. Another type of cloud which is multi-cloud is also discussed. Figure 1.3 describes the different deployment models of the cloud computing.



**Figure No.1.3 Deployment Models of Cloud Computing**

**1.Public Cloud:** These clouds are especially beneficial for small enterprises which allows them to launch businesses without huge investments. The key feature of public cloud is multitenancy so they can serve multiple users instead of single user. The services can easily scale resources depending upon the traffic and workload demands, optimizing performance and cost efficiency. They can help to reduce the need for significant investments in hardware and infrastructure and thereby lowering overall costs. Example of Public Cloud: Amazon EC2, IBM, Azure .

**2.Private Cloud:** Private clouds are operated on the private infrastructure which gives dynamic provisioning of the computing resources to the users. Unlike the pay-as-you-go model used for the public clouds, private clouds may use alternative ways to manage the resource usage and assign costs proportionally across different departments or sections within an enterprise. Examples of Private Cloud are VMware vCloud Suite, OpenStack, Dell Cloud Solutions, HP Helion Eucalyptus.

**3.Hybrid Cloud:** A hybrid cloud is a heterogeneous system that combines the capabilities of both public and private clouds. One major limitation of private

clouds is their lack of scalability to meet on-demand requirements and manage peak loads effectively. Public clouds address this issue by providing additional resources. Examples of hybrid cloud solutions include AWS Outposts, Azure Stack, Google Anthos, and IBM Cloud Satellite.

**4.Community Cloud:** Community clouds cater to the specific needs of a particular industry, community, or business sector. However, managing shared responsibilities among participating organizations can pose challenges. While public clouds typically offer lower security, community clouds provide a higher level of protection. They facilitate the sharing of cloud resources, infrastructure, and capabilities among various organizations. Examples of community clouds include CloudSigma, Nextcloud, Synology C2, and Stratoscale.

**5.Multi-Cloud:** Multi-cloud refers to the use of multiple cloud computing services from different providers, allowing organizations to select the most suitable services for their specific needs while avoiding vendor lock-in. Examples of multi-cloud platforms include Cloud Foundry, Kubernetes, Red Hat OpenShift, and Docker Swarm.

## **1.5 Intrusion Detection System**

Clouds can become attractive targets for intruders due to its distributed nature. Traditional security methods like firewalls are not sufficient to meet the security issues. A robust system like an Intrusion Detection System (IDS) is necessary for effectively detecting attacks in the cloud computing environment.

Security is the one of the biggest issues of the cloud computing model [A. R. Suraj et al., 2018]. Figure 1.4 describes the CIA related to security.





**Figure No.1.4 CIA related to Security**

IDS is a system which autonomously detect inappropriate events like intrusion attacks which are occurring in computer systems. An IDS with effective countermeasures is crucial for detecting attacks. The identification of different kinds of malicious network traffic along with computer utilization is the key objective of any IDS, which cannot be identifiable by traditional techniques.

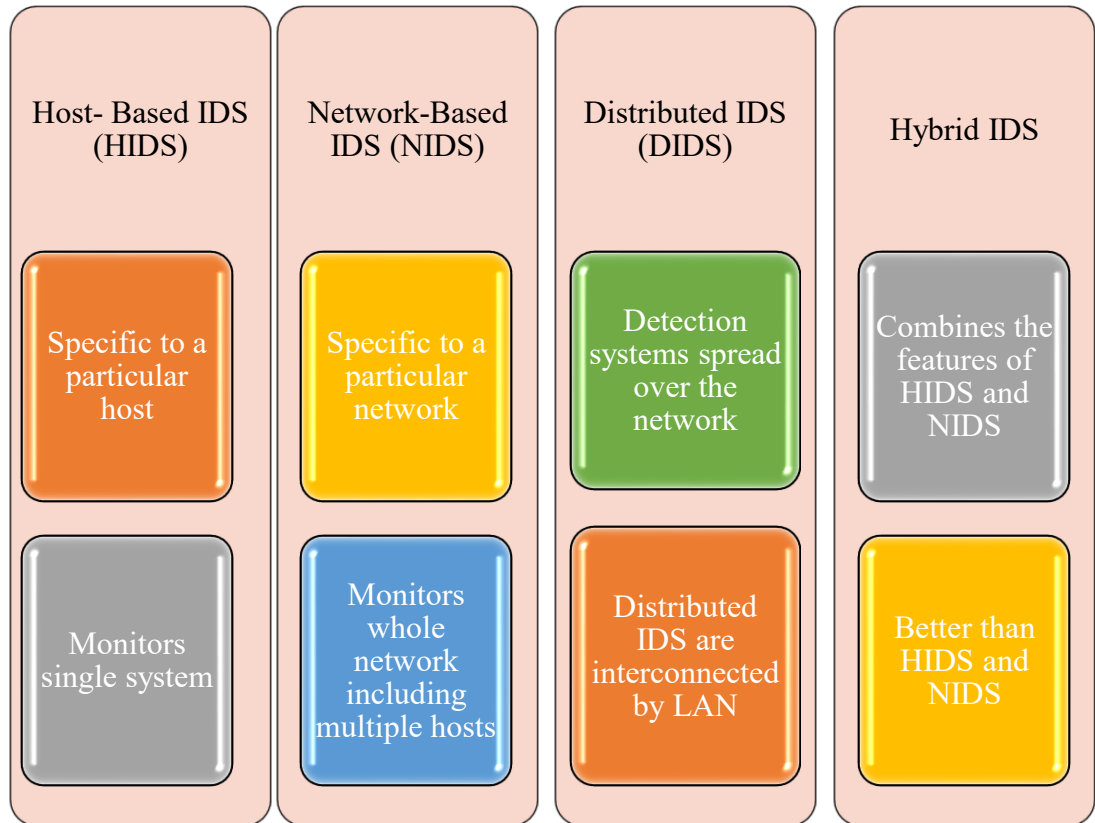
Many researchers have applied Data Mining (DM) and Machine Learning (ML) techniques to address cybersecurity challenges [P. Singh et al.,2014]. ML classifiers are commonly employed to differentiate between attack packets and normal packets [F. Kuang et al.,2014]. Additionally, rule association mining is an emerging technique in this domain [C. Nkikabahizi et al., 2017].

Neural Network is frequently used due to its capability to handle incomplete datasets [V. Balamurugan and R. Saravanan, 2019]. Various optimization algorithms like genetic algorithm [P. Ghamisi and J A. Benediktsson, 2014], particle swarm optimization [A. S. Saljoughi et al., 2017], firefly algorithm [X.S. Yang, 2008], harmony search [K. Costa et al.,2012] and artificial bee colony [S. Aljawarneh et al, 2018] have also been integrated with classifiers to categorize the network traffic.

### 1.5.1 Types of IDS Based on Monitoring Environment

There are four types of IDS, categorized based on the environment they monitor.

Figure 1.5 describes the types of IDS based on the monitoring environment.



**Figure No.1.5 Types of IDS Based on Monitoring Environment**

**1. Host-based IDS (HIDS):** When unknown malicious code is detected, host-based IDS relies on an individual device to detect critical files of the operating system for unusual or malicious activity. It only protects the host device on which it is located. Individual host machines may be programmed with a general set of rules during installation. To take into account new vulnerabilities, new rules can be loaded periodically in the host system.

**2. Network-based IDS (NIDS):** The device analyzes network links for irregular traffic and tracks them. NIDS are devices that are distributed within networks in an intelligent way and detect malicious traffic on a network.

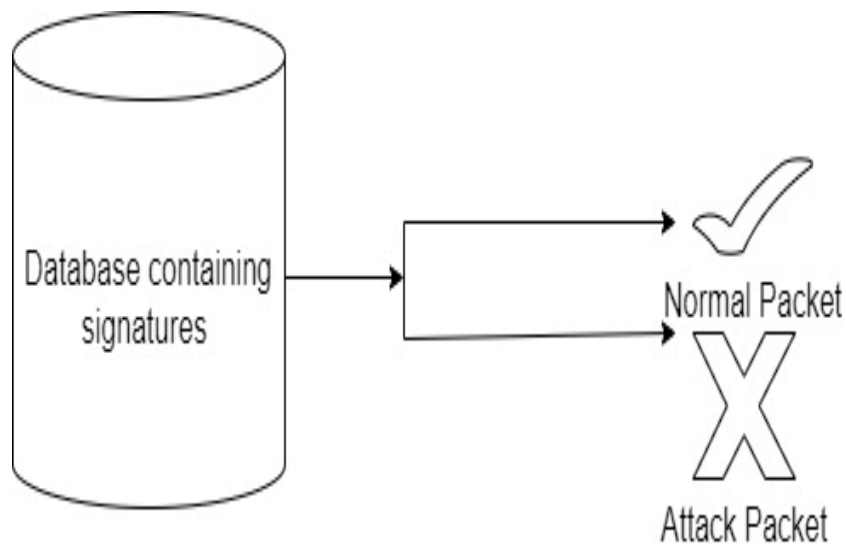
**3. Hybrid IDS:** This IDS combines both HIDS and NIDS components. The mobile agent travelling to each host conducts a device log file checker, while a central agent will check for irregularities across the entire network.

**4. Distributed IDS (DIDS):** DIDS composed of various key benefits of this IDS is that its decentralized nature and scalability along with it does not have a single point of failure. Apart from this having two important challenges; the detection algorithm for each distributed network on different locations as well as how the information is shared among all network. As depicted in figure there is one centralized server which involves various sub-networks along with IDS and this centralized passed summary statistics of their observed statistics.

### **1.5.2 Detection Techniques of IDS**

There are four types of IDS that are categorized on the basis of the detection techniques used by them. Figure 1.6 illustrates the types of IDS categorized by their detection techniques.

**1.Signature-Based IDS:** This IDS have ability to easily identify the known attacks; whereas it's challenging to identify the unknown attacks through known signature their corresponding pattern is unavailable. Figure 1.6 represents the pictorial representation of the Signature-Based IDS.



**Figure No.1.6 Signature -Based IDS**

**2.Anomaly-Based IDS:** It analyses the behaviour of networks, get patterns and then identify deviations corresponds to anomalies. The key benefit of this detection scheme over signature-based IDS is that it can identify attempts to exploit unknown cyber-threats or, vulnerabilities. Although it may generate high false alarm rate through taking acre of existing unknown system behaviours as anomalies utilized for known attacks.

The identification of different kinds of malicious network traffic along with computer utilization is the key objective of any IDS, which cannot be identifiable by traditional firewall.

**Table 1.1 Comparison of Detection Techniques of Intrusion Detection System**

<b>Feature</b>	<b>Signature-Based IDS</b>	<b>Anomaly-Based IDS</b>
<b>Detection Method</b>	Matches traffic against a database of known attack signatures.	Detects deviations from normal behaviour patterns.
<b>Effectiveness</b>	Highly effective for detecting known attacks.	Effective in identifying new or unknown attacks.
<b>False Positives</b>	Low, since it detects based on specific signatures.	Higher, as unusual but legitimate activity can trigger alerts.
<b>False Negatives</b>	High for new, unknown attacks (cannot detect them without existing signatures).	Lower for novel attacks, but can miss subtle variations of normal activity.
<b>Response Time</b>	Fast, as it only matches known signatures.	Slower due to the complexity of analyzing behavioral anomalies.
<b>Maintenance</b>	Requires frequent updates to the signature database to detect new threats.	Requires continuous training and tuning of the model to adjust to changing behavior patterns.
<b>Resource Consumption</b>	Generally consumes fewer system resources.	Requires more processing power due to constant behavior analysis.

<b>Ideal Use Case</b>	Best for environments where known threats are prevalent.	Ideal for environments where new or unknown threats are expected.
-----------------------	--	---

### 1.5.3 IDS Related to Cloud Computing

There are four types of IDS related to Cloud Computing Environment.

**1. Network-Based IDS (NIDS):** Network-based Intrusion Detection Systems (NIDS) monitor and analyze traffic across an entire network to identify potential intrusions, such as port scanning or Denial of Service (DoS) attacks. In a cloud environment, NIDS can detect attacks targeting the hypervisor or virtual machines (VMs) when positioned at the cloud server interacting with external networks. Typically, CSPs are responsible for deploying NIDS within the cloud infrastructure.

**2. Host-Based IDS (HIDS):** Host-based Intrusion Detection Systems (HIDS) collect and analyze data from a specific host to identify intrusive activities. In cloud computing environments, HIDS can be deployed on hypervisors, virtual machines (VMs), or individual hosts. It monitors system logs, user login activity, and access control policies to detect potential threats. While the cloud provider manages HIDS deployment at the hypervisor or host level, cloud users are responsible for managing HIDS on their VMs.

**3. Distributed IDS (DIDS):** A Distributed Intrusion Detection System (DIDS) comprises multiple IDS units, such as NIDS and HIDS, distributed across a large network to monitor traffic for signs of intrusion. These IDS units can communicate with each other directly or through a centralized server. In a cloud environment, DIDS can be deployed on processing servers or directly on host machines.

**4. Hypervisor-Based IDS:** The hypervisor enables communication between virtual machines (VMs) in a cloud environment. A hypervisor-based IDS is deployed at the hypervisor layer to analyze this communication and detect any anomalous activities. It monitors interactions at various levels, including VM-to-hypervisor communication, VM-to-VM communication, and within the hypervisor-managed virtual network.

#### **1.5.4 Examples of Real-time Security Breaches Related to Cloud Computing:**

**1. Tesla Cloud Cryptojacking (2018):** Cybercriminals infiltrated Tesla's AWS cloud environment by exploiting an unprotected Kubernetes console that lacked password security. They leveraged Tesla's cloud infrastructure to mine cryptocurrency, leading to increased operational costs and security threats. Failing to secure administrative tools properly can result in significant financial and cybersecurity risks.

**2. Capital One Data Breach (2019):** A misconfigured web application firewall (WAF) in Capital One's AWS cloud environment was exploited by a former AWS engineer, resulting in a major data breach. This incident exposed the personal and financial information of over 100 million customers. Improper cloud security configurations can create significant vulnerabilities, leading to widespread data leaks.

**3. Facebook Cloud Storage Leak (2019):** Due to misconfigured Amazon S3 buckets by third-party developers, **540 million Facebook user records** were exposed. The leaked data included personal details such as user IDs, passwords, and activity logs. To prevent such incidents, organizations should implement strict cloud access controls and closely monitor third-party cloud integrations.

**4. Microsoft Azure Cosmos DB "ChaosDB" (2021):** Security researchers discovered a flaw in Microsoft's Cosmos DB service that permitted unauthorized access to customer databases. This vulnerability highlighted how even weaknesses within cloud service providers can pose significant risks to businesses.

## 1.6 Attacks affecting Network

Active attacks are like a bandit storming your fortress, while passive attacks are like a spy hiding in the shadows which are silently observing. Both types of attacks are dangerous but in different ways. The active attacks disrupt and demand immediate defense whereas passive attacks can go unnoticed for a long time which potentially lead to major leaks of sensitive information.

**Table 1.2 Comparison of Active Attack and Passive Attack**

Aspect	Active Attack	Passive Attack
Definition	Involves direct interference with system operations or data.	Involves monitoring or eavesdropping on communications.
Objective	To alter, modify, or damage the target's data or system.	To gather information without detection.
Visibility	Typically noticeable by the victim.	Usually stealthy and goes unnoticed by the victim.



<b>Impact on Data</b>	Data integrity, availability, and authenticity may be compromised.	Confidentiality of data may be compromised.
<b>Interaction with System</b>	Direct interaction with the target system or communication.	No interaction with the target system, only observation.
<b>Detection Difficulty</b>	Easier to detect due to noticeable effects on the system.	Harder to detect as no immediate changes occur in the system.
<b>Response Requirement</b>	Requires immediate response and recovery actions.	May go undetected, so delayed response if discovered.
<b>Risk Level</b>	Higher risk due to the potential for immediate damage.	Lower risk but can lead to more damaging active attacks.
<b>Example Attacks</b>	Data modification, denial of service (DoS), session hijacking.	Eavesdropping, traffic analysis, password sniffing.

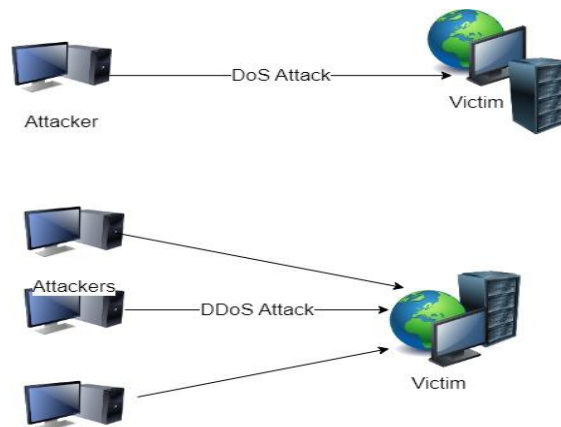
## 1. Denial-of-Service (DoS) Attack

A Denial of Service (DoS) attack refers to any event or malicious action that diminishes or disrupts a cloud's ability to deliver the services and functionalities that users expect. A DoS attack is one that targets a resource or service in the cloud with the intention of temporarily preventing it from offering its regular services. It

is a cyberattack aimed at disrupting the services of a specific computer or website, preventing legitimate users from accessing it. The goal is to interfere with an organization's network operations by overwhelming the target with excessive requests, causing system overload. This flood of traffic makes it difficult or impossible for the system to process genuine requests, effectively denying access to users.

## 2. Distributed DoS (DDoS) Attack

Circulated version of Distributed DoS (DDoS) attack is referred to as DoS attacks. It uses many network hosts to do more damage damaging consequences for its sufferer. Instead of one attacker, DDoS attacks involve multiple compromised devices (often part of a botnet) working together to flood a target with traffic, overwhelming its resources and causing the service to become unavailable. DDoS attacks are highly dangerous due to their ability to disrupt online services and infrastructure with a coordinated, large-scale assault. Figure 1.7 represents the DoS attack and DDoS attack.



**Figure No.1.7 Pictorial Representation of DoS attack and DDoS attack**

### **3. Bruteforce Attack**

This type of attack involves attempting to guess a password through repeated trial and error. The attacker tries various combinations of characters until they successfully identify the correct password, granting them access to the system. This approach relies on the premise that eventually all combinations will be tested, leading to the discovery of the correct one. Brute-force attacks can be time-consuming and resource-intensive, particularly as the complexity of the password or encryption increases. To mitigate the risk of such attacks, it is advisable to use strong, complex passwords and implement security measures such as account lockouts after multiple failed attempts, CAPTCHA challenges, or multi-factor authentication (MFA).

### **4. Infiltration Attack**

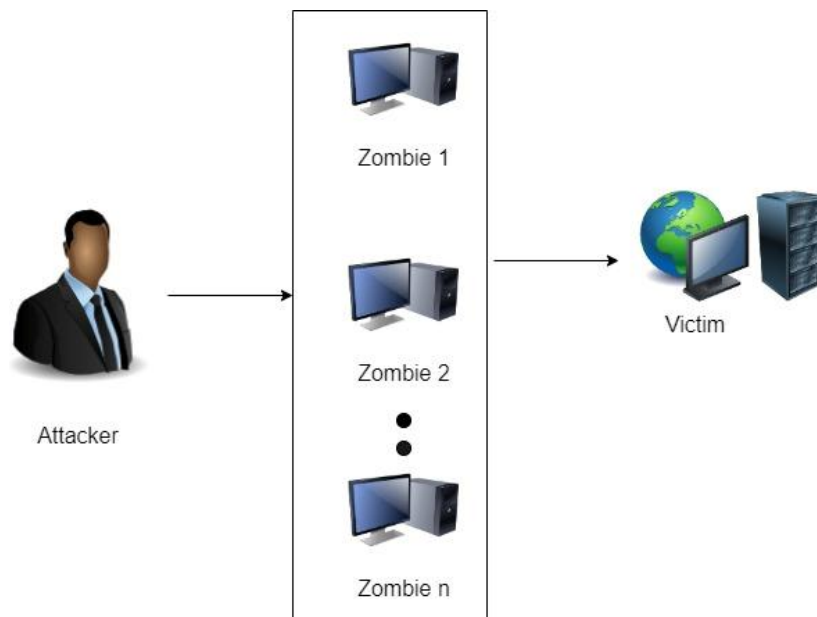
This attack tries to exploit application related vulnerabilities. They send harmful email to victim. After performing attack, a backdoor is installed in the victim. Then other vulnerabilities of the system are exploited by attacker. This type of attack typically focuses on penetrating the internal network of an organization or system rather than just attacking external-facing components. Here's an overview of how infiltration attacks work and their key characteristics:

### **5. SQL Injection Attack**

SQL injection (SQLi) is a type of attack where an attacker exploits vulnerabilities in an application's interface to execute malicious SQL queries against a database. The attackers aim to disrupt the queries made to the database by the applications. The attacker can view user data or application data. He can delete or modify the files present in the database

## 6. Botnet Attack

This attack uses zombies which are computer systems affected by malware. These systems can launch DDoS attack or send spam mails to the victim. A botnet attack involves using a network of compromised devices, known as bots or zombies, to carry out various malicious activities. The devices in the botnet are controlled by a central command-and-control (C2) server operated by the attacker. Here's a breakdown of how botnet attacks work and their impact: Figure 1.8 represents the Botnet attack.



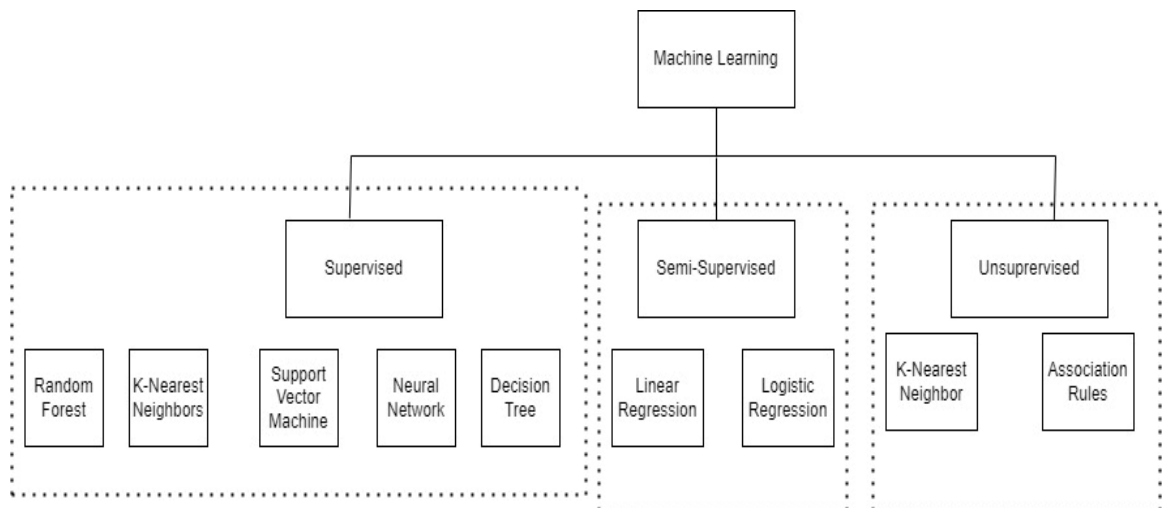
**Figure 1.8 Pictorial Representation of Botnet Attack**

## 1.7 Machine Learning

A subtype of artificial intelligence (AI) known as machine learning (ML) enables programmes to gain knowledge from data and experience without needing to be explicitly code. Training and classification are two common tasks in machine learning. A labelled dataset is given to the machine learning model during training

so that it may discover the connections between inputs (features) and outputs (labels).

Predicting a type of categorical label for an input data sample is the aim of a particular kind of machine learning problem called classification. For example, given a feature vector of a IDS system, the goal of a classification model would be to predict which category of intrusion the data belongs to. There are following type of machine learning algorithm architecture in a broad manner. Figure 1.9 shows various types of machine learning techniques.



**Figure 1.9 Types of Machine Learning**

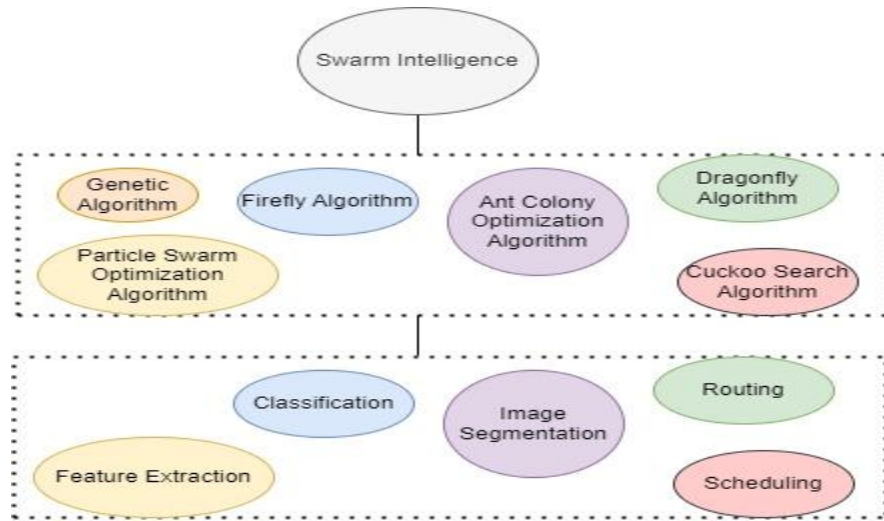
- **Supervised Learning:** A labelled dataset with known values for the target parameter is used to train the algorithm. Making forecasts for fresh, unforeseen data is the aim. Support vector machines (SVMs), decision trees, logistic regression, and linear regression are a few examples.
- **Unsupervised Learning:** The purpose of the unlabelled dataset used to train the algorithm is to find patterns or correlations in the data. A few examples include anomaly detection, dimensionality reduction (PCA), and clustering (K-means).

- **Semi-supervised Learning:** The training is done on a partially labelled dataset, where some instances have known labels and others do not. The goal is to make predictions for the unlabelled instances based on the patterns learned from the labelled instances.

## 1.8 Swarm Intelligence

SI represents a paradigm shift in how we approach problem-solving and decision-making in complex systems. It draws inspiration from the elegance and efficiency of social organisms that have evolved over millions of years to survive and thrive in their environments. By studying and emulating the collective behaviour of these organisms, researchers have unlocked a new frontier of possibilities in artificial intelligence, optimization, and distributed computing.

The power of SI lies in its decentralized and self-organizing nature. Unlike traditional approaches that rely on centralized control or complex algorithms, Swarm Intelligence harnesses the inherent wisdom of the crowd. Each individual agent, whether it's an ant, a bird, a robot, or a particle, follows simple rules based on local information and interactions with its peers. These simple rules give rise to emergent behavior at the group level, enabling swarms to tackle complex tasks with surprising efficiency. One of the captivating aspects of Swarm Intelligence is its adaptability and robustness. Swarms are capable of responding dynamically to changes in their environment without the need for global communication or top-down coordination. This adaptability makes them highly suitable for real-world problems that are subject to uncertainties and dynamic conditions, such as route optimization, resource allocation, anomaly detection, attack recognition, and data clustering. Figure 1.10 shows various types of swarm intelligence.



**Figure No. 1.10 Types of Swarm Intelligence**

SI algorithms for feature selection can be categorized into various types based on the nature of the swarm and optimization techniques they employ. Some of the prominent categories include:

**1. Particle Swarm Optimization (PSO):** PSO algorithms are inspired by the social behavior of birds or fish. In feature selection, particles represent potential feature subsets, and they adapt their positions in the feature space to find the best subset that minimizes or maximizes a given objective function.

**2. Firefly Algorithm (FA):** The Firefly Algorithm is a type of SI algorithm that models the flashing behavior of fireflies. Firefly algorithms are used for feature selection by mimicking the movement of fireflies towards brighter neighbors in the feature space, representing promising feature subsets.

**3. Dragonfly Algorithm:** The Dragonfly Algorithm is another SI-based approach that draws inspiration from the swarming behavior of dragonflies. These algorithms

optimize feature selection by mimicking the movements and interactions of dragonflies in the search for optimal feature subsets.

**4.Artificial Bee Colony (ABC) Algorithm:** ABC algorithms are inspired by the foraging behavior of honeybees. In feature selection, they employ a population of artificial bees to explore and evaluate feature subsets, converging towards an optimal solution.

**5.Cuckoo Search (CS) Algorithm:** Cuckoo Search algorithms are inspired by the breeding behavior of cuckoo birds. They optimize feature selection by simulating the random selection of host nests and the subsequent replacement of less fit feature subsets.

## **1.9 Research Motivation**

In the modern digital era, the rapid growth of internet usage and emerging technologies has led to a significant rise in cyber threats. These attacks pose serious risks, including financial damage, harm to an organization's reputation, and interruptions to critical services. Traditional security approaches often fail to keep up with the evolving complexity of these threats, highlighting the need for robust Intrusion Detection Systems (IDS).

One of the primary challenges in building an efficient IDS is accurately analyzing network traffic to distinguish between normal and malicious behavior. This involves two key tasks: selecting the most important features from complex and large-scale datasets, and choosing the right classification algorithms to ensure high detection accuracy.

IDS datasets, such as the CSE-CIC IDS 2018 dataset, contain numerous attributes that describe different aspects of network activity. However, not all features are equally useful. Some may be redundant or irrelevant, leading to increased



processing time and decreased system accuracy. Therefore, identifying the most informative features is crucial for effective threat detection.

Feature selection plays a vital role in improving the performance of IDS. By reducing the number of features, it lowers computational overhead and improves accuracy by eliminating unnecessary data. Finding the best subset of features requires advanced techniques that can balance the trade-off between speed and detection quality.

Another major consideration is the use of classifiers—machine learning models that help identify whether activity is malicious or legitimate. Each type of classifier has its strengths: Decision Trees offer simplicity and clarity, Support Vector Machines perform well in high-dimensional data, and Neural Networks are capable of capturing complex patterns. However, no single model is perfect for every scenario.

To overcome this, combining several classifiers into a hybrid model can provide better overall performance. This approach takes advantage of each algorithm's strengths, aiming to increase detection rates while minimizing errors such as false alarms or missed threats. The challenge lies in integrating these models effectively to build a unified and reliable system.

It's not enough to design a model in theory—it must be tested in realistic environments to confirm its effectiveness. A well-structured simulation platform should reflect real network conditions and allow thorough evaluation under various circumstances. This ensures the IDS can be assessed in terms of performance, scalability, and adaptability.

Ultimately, developing such a system leads to stronger defenses against cyber threats, making cloud and network environments more secure.

## **1.10 Organization of the thesis chapters**

**Chapter 1:** This chapter provides a detailed description related to the cloud computing environment and various attacks affecting it. This chapter describes intrusion detection system and its types in detail.

**Chapter 2:** This chapter outlines the related work, summarizing the research drafts published prior to this study. It includes a comparative analysis and a literature review focused on cloud computing, feature selection and intrusion detection systems. Additionally, this chapter clearly defines the problem statement and identifies research gaps pertinent to this work.

**Chapter 3:** This chapter presents the research objectives and details the methodology employed throughout the research process. A flowchart is provided to illustrate the steps taken to achieve the research objectives.

**Chapter 4:** This chapter discusses the generation of a simulated dataset using a cloud simulation tool. Both the simulated dataset and the standard CSE CIC IDS 2018 dataset are utilized to evaluate the architecture developed for attack detection. Preprocessing and feature selection are conducted on these datasets.

**Chapter 5:** This chapter details the architecture designed for detecting attacks in a cloud computing environment. It explains the various modules of the architecture in depth.

**Chapter 6:** This chapter describes the results and discussions of the research work. The results for the standard benchmark dataset CSE CIC IDS 2018 dataset and simulated dataset are described in this chapter. Results are presented in tabular and graphical form. Description of results are given in this chapter. Different sample sizes are taken for result generation.

**Chapter 7:** This chapter describes the conclusion and outlines the future scope related to the research work. Conclusion is extracting the summary of the research work. Future scope tells the possible future directions related to the research.

### **1.11 Summary**

A thorough introduction to cloud computing environment is given in this chapter, which also highlights its importance and widespread use in the contemporary computing. It describes many security concerns encountered by cloud systems, demonstrating various types of attacks that risk resources of the cloud. The importance of machine learning algorithms in improving attack detection skills is emphasized, along with the power of swarm intelligence, which together create a robust intrusion detection system.

## CHAPTER 2 LITERATURE REVIEW

---

This chapter presents an overview of the recent research focused on improving intrusion detection and security measures within cloud computing environments. Researchers have investigated a variety of approaches, including traditional machine learning techniques, to develop effective intrusion detection systems (IDS) capable of identifying attacks. The chapter also explores innovative methods, such as nature-inspired algorithms and hybrid detection systems that combine multiple detection techniques for more comprehensive cloud protection. By reviewing key studies and their findings, the chapter sheds light on the evolving landscape of intrusion detection in cloud computing environment, highlighting significant advancements in security measures within this critical and dynamic field. Additionally, problem statement with research gaps identified through the literature review.

### 2.1 Cloud Computing

[C. Gong et al. ,2010] highlighted the core features of cloud computing, describing it as an advanced evolution of existing distributed computing models, such as cluster and grid computing. Cloud computing has emerged from innovations in distributed systems, Internet technologies like service-oriented architecture (SOA), hardware virtualization, and autonomic computing. The cloud offers users the impression of unlimited, on-demand resources that can be accessed from any location. This mobility and collaborative functionality are reinforced by infrastructure abstraction, which hides complexity from users. The paper outlines key cloud computing characteristics, facilitating its development and widespread adoption. One major feature is its service-oriented architecture, which simplifies complex internal operations. Another technical feature is loose coupling, applicable

across various cloud systems. Cloud computing's robust fault tolerance also makes it compatible with commonly used network infrastructures. Its economic benefits are a primary driver for business adoption, distinguishing it from high-performance computing (HPC) and grid computing. The user-friendly design abstracts service provider complexities through simple interfaces. Other key features include reliance on TCP/IP for Internet access, virtualization capabilities, and enhanced security.

[E. Besharati et al., 2019] explored security issues in cloud environments and introduced a host-based intrusion detection system (H-IDS) to protect virtual machines. Key features for each category were selected using logistic regression and further refined through regularization techniques. Attack classification was performed using a combination of neural networks, decision trees, and linear discriminant analysis, enhanced with a bagging algorithm to improve accuracy. The proposed model was evaluated on the NSL-KDD dataset and implemented in the CloudSim simulation environment. Results showed a detection accuracy of 97.51%, surpassing other methods in identifying attacks.

[Y. S. Abdulsalam and M. Hedabo, 2021] highlighted that the main challenge in securing critical internet infrastructures, like cloud computing, is ensuring the system's ability to self-protect regarding security and privacy. It is crucial to implement secure adaptive techniques, which can be applied at different levels of the technology stack, including hardware, software, and core computing infrastructure. Secure adaptiveness enables the system to defend itself against a range of attacks or malicious users exploiting vulnerabilities. Without the practical deployment of these adaptive mechanisms, cloud computing will remain vulnerable to security and privacy threats, putting the efficiency of client and user experiences at risk.

## 2.2 Feature Selection

**Filter Method:** Filter methods assess the relevance of features based on statistical measures, independent of any machine learning algorithms. In this case, the authors utilize feature correlation to select important features, calculating the correlation coefficients between features to construct a correlation matrix. The average correlation of each feature with others is then computed to determine its significance. This approach allows for the selection of features that are most correlated with other features, aiming to enhance detection performance while reducing computational complexity

[D. Rani and N. C. Kaushal, 2020] introduced a hybrid intrusion detection system that combined the C5.0 Decision Tree with a One-Class SVM. The C5.0 model was employed for misuse detection, effectively identifying known attacks with a low false alarm rate, while the One-Class SVM focused on anomaly detection by training only on normal traffic. During training, decision boundaries were established based on normal data, and outliers were flagged as potential attacks. Testing on the NSL-KDD dataset showed that this hybrid model improved detection rates and reduced false alarms compared to previous methods.

[V. D. Ngo et al., 2024] compared feature selection and feature extraction methods in intrusion detection, using the UNSW-NB15 dataset for both binary and multiclass classification. The study found that feature selection provided higher detection accuracy and required less training and inference time when the number of features was moderately large (e.g., 8 or 16). However, feature extraction performed better with fewer features (e.g., 4 or fewer). The study also found that MLP was most effective for feature extraction, while the Decision Tree excelled in feature selection for attack detection.

**Wrapper Method:** In the realm of feature selection methodologies, wrapper methods are one of the primary categories, alongside filter and embedded methods. Wrapper methods evaluate subsets of features by training and testing a specific machine learning model, aiming to identify the combination that yields the best performance. In this study, the authors applied the wrapper method using a decision tree to select features that enhance the performance of various machine learning algorithms in intrusion detection systems

[O. Alomari and Z. A. Othman, 2012] proposed a feature selection method based on a wrapper approach, utilizing the Bees Algorithm (BA) for generating subsets and Support Vector Machine (SVM) as the classifier. The study tested the method on four randomly selected subsets from the KDD-Cup 99 dataset, each containing approximately 4,000 records. The performance was evaluated using standard intrusion detection system (IDS) metrics, achieving a detection accuracy of 99%, with the feature set reduced to only 8 features and a false alarm rate of 0.004.

[I. Ahmad et al., 2014] presented an intrusion detection technique that focuses on feature subset selection through a combination of Genetic Algorithms (GA) and Principal Component Analysis (PCA), alongside a Multilayer Perceptron (MLP) classifier. The integration of GA and PCA addressed performance challenges, improving feature selection and accuracy. Using the KDD-Cup dataset, the method reduced the feature set from 41 to 12, achieving a detection accuracy of 99%. This optimized feature selection enhanced precision and reduced computational demands, making the intrusion detection system more efficient.

[M. Otair et al., 2022] developed an intrusion detection method that optimized feature selection using Grey Wolf Optimization (GWO). They enhanced this process with Particle Swarm Optimization (PSO), which refined the GWO results and avoided local minima. Simulations using k-means and SVM on the NSL-KDD

dataset revealed that this hybrid optimization technique outperformed other methods, showing improvements in false alarm rate, detection rate, detection accuracy, and execution time due to more efficient feature selection.

[S. K. Shandilya et al., 2023] introduced an advanced firefly optimization algorithm for network monitoring, incorporating a new health function to identify suspicious nodes early. This algorithm integrates event management and optimizes the observation priority list using a genetic evolution algorithm to handle real-time network events. Simulations showed the method's effectiveness, reducing suspicious nodes by 60-80% with only a slight increase in turnaround time (1-2%). The focus was on proactive network health monitoring for enhanced protection.

[M. A. Umar et al., 2024] conducted an analysis of the impact of feature selection and normalization on various Intrusion Detection System (IDS) models, using the NSL-KDD and UNSW-NB15 datasets. The study employed five machine learning algorithms and used a decision tree wrapper-based feature selection method and min-max normalization. The random forest model performed the best, achieving 99.87% and 98.5% accuracy and 99.79% and 99.17% detection rates on the NSL-KDD and UNSW-NB15 datasets, respectively, outperforming many recent IDS studies.



[Y. K. Saheed et al., 2024] presented a hybrid feature selection technique that combined the Bat algorithm with the Residue Number System (RNS). The Bat algorithm initially partitions the data and eliminates irrelevant features. RNS was integrated to enhance processing speed and reduce training time. In the second phase, RNS was removed, and the Bat algorithm focused solely on feature selection while PCA handled feature extraction. Naive Bayes and k-Nearest Neighbors classifiers were used for classification. The method improved detection rates, accuracy, and F-scores while doubling processing speed, demonstrating competitive results compared to other intrusion detection techniques.

[M. Bakro et al., 2024] proposed a hybrid feature selection strategy that combined the grasshopper optimization algorithm (GOA) and the genetic algorithm (GA) for efficient feature selection. The random forest classifier was trained on the selected features. To address class imbalance, a hybrid approach was used: the adaptive synthetic (ADASYN) algorithm oversampled minority classes, while random under-sampling (RUS) was used for the majority class. The approach showed improved performance, increasing the true positive rate (TPR) and reducing the false positive rate (FPR). The method was tested on three datasets—UNSW-NB15, CIC-DDoS2019, and CIC Bell DNS EXF 2021—achieving accuracies of 98%, 99%, and 92%, respectively.

**Embedded Selection:** Embedded feature selection is a technique used in machine learning to automatically choose the most important input features (variables) during the process of building a model. Unlike other methods that select features before or after training the model, embedded methods do both tasks at the same time.

These techniques are built into certain algorithms that have their own way of ranking or filtering features based on how useful they are for making predictions. For example, decision trees or regularized regression models (like Lasso) can reduce the importance of less relevant features while focusing more on the key

ones. This approach is efficient because it reduces the number of features while optimizing the model's performance, saving time and improving accuracy.

[S. Ganapathy et al., 2016] introduced a feature selection method based on Conditional Random Fields (CRF) for more efficient intrusion detection. The CRF-based algorithm aimed to minimize the number of features while improving their relevance. A Layered Approach (LA)-based algorithm was then applied for classification with the reduced feature set. The resulting system demonstrated better accuracy and efficiency in detecting attacks compared to existing methods. Key benefits included faster detection times, improved classification accuracy, and fewer false alarms. Evaluations on the KDD-Cup99 dataset showed the following detection accuracies for different attack types: probe = 99.98%, DoS = 97.62%, R2L = 32.43%, and U2R = 86.91%.

### **2.3 Attacks Affecting Cloud Computing Environment**

[V. R. Kebande and H. S. Venter, 2014] introduced an innovative botnet detection system specifically designed for cloud environments, utilizing the artificial immune system. With the growing prevalence of botnet attacks that cause service disruptions and resource depletion, this approach uses a negative selection algorithm to determine whether a botnet matches self or non-self-patterns. The detectors are trained to identify malicious activity in the cloud and classify it as non-self, allowing for effective isolation of the attack.

[G. Somani et al., 2017] explored the critical issue of Distributed Denial of Service (DDoS) attacks in cloud environments. The paper offers an extensive review of recent advancements in DDoS mitigation techniques tailored for cloud computing. It examines attack characterization, prevention, detection, and mitigation strategies, proposing a detailed taxonomy to categorize existing solutions. The authors stress

the importance of solutions customized for utility computing models, highlighting the need for accurate auto-scaling, multi-layered defense systems, and efficient resource management in cloud environments. Their work provides valuable insights and guidelines for developing effective mitigation techniques, aiming to assist the cybersecurity community in building stronger defense mechanisms. Notably, it pioneers the identification of multi-level information flow and resource management strategies during DDoS attacks.

[M. K. H. Al-Dulaimi, 2024] presents a detailed overview of blockchain-based security solutions designed to address DDoS, Man-in-the-Middle (MITM), and SQL injection attacks. However, it is important to note that the proposed solution has not yet been tested in real-world scenarios, indicating the need for further research and validation to determine its effectiveness and practical feasibility. Despite this, the integration of blockchain technology into cloud security shows considerable potential to strengthen the overall security framework of cloud environments.

## **2.4 Similarity Comparison of Benchmark Dataset with Existing Dataset**

[N. Cao et al., 2013] developed a framework for performing privacy-preserving ranked search on encrypted cloud data. The study focused on aligning simulated keyword queries with real encrypted datasets, ensuring secure and efficient search operations. Simulated datasets were created from hypothetical user interactions, while actual datasets comprised encrypted academic and corporate documents. The mapping used cosine similarity for keyword alignment, complemented by hybrid measures incorporating the Jaccard index for set-based matching. Min-max scaling was applied to normalize keyword frequencies, ensuring consistent term importance across datasets. The authors emphasized that this mapping process allowed for high relevance in search results while maintaining strong privacy

protections, highlighting the importance of accurately mapping simulated data to real-world encrypted datasets for effective privacy-preserving search systems.

[B. Wang et al., 2016] addressed privacy-preserving searchable encryption by mapping simulated encrypted queries to real encrypted document datasets. The focus was on enabling efficient searches of encrypted text while maintaining security. Simulated datasets consisted of hypothetical query logs, while real datasets included encrypted academic documents from sources like PubMed. The mapping relied on cosine similarity for text alignment, combined with weighted Euclidean distance to account for term frequency and relevance. A preprocessing step involved normalizing keyword vectors through token standardization and frequency scaling to ensure that both simulated and real datasets shared the same statistical distribution. The study concluded that precise mapping significantly enhanced the relevance of search results, improving recall and precision in retrieving encrypted documents. This work demonstrated the effectiveness of hybrid similarity measures for handling large encrypted datasets while safeguarding user privacy.

[V. Popic and S. Batzoglou, 2017] introduced a hybrid cloud read aligner that mapped simulated genomic reads to actual genome assemblies. Simulated reads were generated using sequencing tools mimicking real-world error patterns, while actual datasets were sourced from repositories like Ensembl and NCBI. The alignment process involved two steps: MinHash for rapid similarity estimation, followed by cosine similarity for precise genetic sequence alignment. Normalization was applied to sequence lengths to avoid bias due to read size variations. This methodology demonstrated excellent scalability and accuracy, enabling efficient genomic data alignment in cloud-based environments.

[W. Liang et al., 2019] proposed an intrusion detection algorithm using clustering-based optimization models to map simulated attack vectors to real-world traffic

patterns. Simulated logs, generated using penetration testing tools, were aligned with industrial control system traffic. The mapping was done using cosine similarity to measure the alignment of multivariate feature vectors, and clustering techniques were used to group similar traffic patterns. Min-max normalization was applied to standardize features like bandwidth and packet rates. The study found that accurate mapping improved the detection of multi-stage attacks, enhancing the effectiveness of intrusion detection systems.

[Y. Miao et al., 2020] introduced an airborne LiDAR system for UAV-based obstacle recognition and intrusion detection. The study involved generating simulated LiDAR point cloud data under controlled conditions and mapping it to real-world LiDAR data obtained during UAV field tests. Simulated datasets were generated using models replicating environmental features like building heights and tree densities. Mapping was achieved through transfer learning, using cosine similarity to align features such as point density, elevation gradients, and object contours between the datasets. The authors showed that hybrid similarity measures combining cosine similarity and density-based clustering were effective in distinguishing obstacles from noise in LiDAR data. Normalization via z-score scaling ensured consistent feature distributions, compensating for variations in point cloud density and collection angles. This approach improved the UAV system's ability to perform accurate obstacle detection and navigation in complex environments.

[A. Alshammari and A. Aldribi, 2021] applied machine learning techniques to detect malicious network traffic in cloud environments. Their approach involved mapping synthetic traffic data generated using tools like Tcpreplay to real-world datasets such as CICIDS2017 and UNSW-NB15. The mapping process used Jaccard similarity for categorical features, such as protocol types and attack labels, and cosine similarity for numerical features, including packet arrival rates and flow durations. The authors emphasized the use of L2 normalization to ensure equal

contribution of numerical features with varying scales (e.g., bandwidth in Mbps and latency in milliseconds). The study demonstrated that this mapping approach improved detection accuracy, especially for identifying multi-vector attacks. By aligning synthetic and real-world datasets, the authors were able to develop models capable of recognizing both known and emerging threats.

[T. Li et al., 2022] provided a comprehensive review of anomaly-based network intrusion detection systems. This study analyzed methods to bridge the gap between simulated and real-world intrusion data. The authors compared datasets like NSL-KDD, which simulate a range of attacks, with real-world traffic datasets such as CICIDS2017, highlighting differences in feature richness and complexity. The mapping of features like packet sizes, flow durations, and attack signatures was achieved through advanced similarity metrics. Techniques like cosine similarity and clustering algorithms were used to measure alignment between the distributions of simulated and real data. The authors concluded that integrating real-world data during training enhances detection models' ability to generalize, particularly for identifying zero-day attacks. Normalization was performed using min-max scaling, ensuring that features with different units or scales contributed equally to similarity calculations. The review emphasized that hybrid mapping methods, combining statistical and machine learning approaches, can address the limitations of older datasets and improve intrusion detection.

## 2.5 Intrusion Detection System

**Host-Based IDS:** Host-based IDS (HIDS) in Cloud Computing is a security tool that monitors individual cloud-based virtual machines or servers for signs of suspicious activity or potential attacks.

Unlike network-based IDS, which looks at data traveling across the network, HIDS focuses on what's happening *inside* a specific cloud instance. It keeps track of things

like system logs, file changes, user activities, and running processes. If it detects unusual behavior—like unauthorized access, unexpected file modifications, or strange application behavior—it can raise an alert. In the cloud, where multiple virtual machines may be running different tasks for different users, HIDS provides a deeper level of protection at the machine level. It's especially useful for detecting insider threats or attacks that have already bypassed network defenses.

HIDS helps cloud providers and users maintain visibility and control over each system's security, making it an important layer in a multi-level defense strategy.

In [R. Patil et al., 2019] proposed a Hypervisor-Level Distributed Network Security (HLDNS) framework to secure cloud computing environments. Deployed on each processing server, the framework monitored network traffic for virtual machines using a Random Forest classifier, with features extracted via an extended binary bat algorithm (BBA). The system generated intrusion alerts and correlated them across servers to identify distributed attacks. Testing on a cloud network testbed and evaluation with recent intrusion datasets (UNSW-NB15 and CICIDS-2017) showed the framework's effectiveness in cloud network security.

[L. Chen et al. ,2020] developed a network intrusion detection system specifically for cloud computing environments, addressing challenges like variability and unpredictability of network intrusions. They employed the C4.5 decision tree algorithm along with a random forest algorithm to build their intrusion detection model. Real-time network traffic data was collected from various network levels of cloud servers using the tcpdump tool and data mining techniques. Experimental results showed that the system achieved a 99.71% detection accuracy while reducing training and testing times, proving effective for real-time monitoring and intrusion detection in cloud environments.

**Network Based IDS:** Network IDS (NIDS) in Cloud Computing is a security system designed to monitor and analyze network traffic within a cloud environment to detect suspicious activities or potential cyberattacks. In a cloud setup, multiple virtual machines, services, and users share resources over a virtual network. A Network IDS keeps an eye on this traffic flow, looking for patterns that match known attacks (like malware or hacking attempts) or abnormal behavior that could indicate a new threat. Because cloud networks are often large, dynamic, and spread across different regions or data centers, placing NIDS at strategic points—such as between virtual machines or at the cloud gateway—helps detect threats early without impacting system performance. By continuously inspecting data packets moving through the cloud network, a NIDS helps identify unauthorized access attempts, data breaches, or other malicious actions, playing a critical role in securing cloud infrastructures.

In [Y. Liu and R. Ma, 2013] introduced a novel intrusion detection model, BQPSO-BN, to address the growing need for effective network intrusion detection due to the rise of network attacks in cloud computing environments. They adapted the classical QPSO algorithm for use in a binary search space, aligning it with the discrete nature of Bayesian network learning. Experiments with the KDD'99 dataset demonstrated the superiority of BQPSO-BN, showcasing faster convergence compared to models like BPSO-BN and GA-BN.

In [K. Wang et al., 2014] presented a behavior-based botnet detection model, BB DP, which employed fuzzy pattern recognition to detect botnets in real-time through analysis of DNS queries and TCP requests. Unlike signature-based methods, BB DP utilized a five-stage process: traffic reduction, feature extraction, data partitioning, DNS-based detection, and TCP-based detection. By parallelizing these stages across multiple servers, BB DP achieved high accuracy, with a true



positive rate exceeding 95% and a low false positive rate of 3%. Their experiments on Windows Azure demonstrated the scalability of BBDP.

In [J. Hussain et al.,2016] proposed a two-stage hybrid classification method for Network Intrusion Detection Systems (NIDS), combining Support Vector Machine (SVM) for anomaly detection in the first stage and Artificial Neural Network (ANN) for misuse detection in the second stage. This hybrid approach improved classification accuracy and minimized false positives. Testing with the NSL-KDD dataset resulted in a detection rate of 99.97% and a false positive rate of 0.19%, outperforming traditional models.

In [H.H. Pajouh et al.,2017] introduced a two-tier classification model for network anomaly detection that combined Naïve Bayes, a variant of KNN, and Linear Discriminant Analysis. Evaluated on the NSL-KDD dataset, the model achieved enhanced detection rates and reduced false alarms, specifically targeting rare and dangerous attack types. The model utilized SMOTE for balancing datasets and incorporated efficient dimension reduction and feature selection, leading to minimized computational time and the ability to detect complex attacks that closely resembled normal behaviour.

In [R. Kesavamoorthy and K.R. Soundar, 2019] addressed DDoS attacks in cloud computing with an autonomous multi-agent system. This system used particle swarm optimization to enable effective agent communication and decision-making. Multiple agents detected DDoS attacks by communicating with a coordinator agent, which analyzed scenarios using entropy and covariance methods. The system's performance was optimized for faster attack detection and recovery, with simulations showing a 53% improvement in efficiency compared to HMM-CRL and 45% better than HCF.

[P. Ghosh et al., 2019] discussed how Cloud Computing, offering various services over the Internet, attracted a large user base due to its cost-effectiveness and

efficiency. However, this popularity also made it susceptible to security threats. To address these issues, Intrusion Detection Systems (IDS) were deployed in the cloud environment. Training these IDS systems effectively was crucial for accurate and timely intrusion detection. However, the presence of redundant features in the training data led to increased memory usage and longer training times. The authors proposed a novel CS-PSO-based IDS to classify attacks quickly and efficiently. They utilized the NSL-KDD dataset to demonstrate the effectiveness of their IDS approach.

In [R. Rajendran et al., 2019] addressed security challenges in cloud networks, focusing on Denial of Service (DoS) attacks. They proposed a novel rule-based method for detecting DoS attacks by leveraging domain expert knowledge. The authors also introduced two new algorithms: one for feature selection, called the Feature Selection Algorithm using Scoring and Ranking, and another for classification, the Rule-based Classification Algorithm. Their approach demonstrated improved DoS detection accuracy over existing methods, validated through experiments in a cloud-based test environment with real-time DoS attack tools.

[D. J. Prathyusha and G. Kannayaram, 2020] introduced an innovative IDS based on artificial immune systems (AIS) to mitigate Distributed Denial of Service (DDoS) attacks in cloud computing. By mimicking biological immune systems, their approach identified critical features of DDoS attacks. Experiments with the KDD Cup 99 dataset showed that the AIS-based IDS effectively detected and neutralized DDoS threats, achieving high detection accuracy and a low false alarm rate. This research highlighted the potential of AIS for enhancing cloud security against DDoS attacks.

In [G. S. Kushwah and V. Ranga, 2020] proposed a method for detecting Distributed Denial of Service (DDoS) attacks using voting extreme learning

machines (V-ELM). Their approach demonstrated strong accuracy in detecting attacks, as evidenced by experiments with the NSL-KDD and ISCX datasets. Comparative evaluations revealed that their V-ELM-based system outperformed other detection methods, including backpropagation neural networks, black hole optimization-trained neural networks, extreme learning machines, random forests, and AdaBoost. The system's effectiveness in mitigating DDoS threats in cloud computing environments was reaffirmed through experiments under various parameter settings.

In [K.B. Virupakshar et al., 2020] explored the widespread adoption of cloud computing, driven by the rapid expansion of internet-based applications that reduce IT infrastructure management costs. However, the distributed nature of cloud resources, centrally controlled over the internet, makes them vulnerable to potential intrusions. The study focused on detecting Distributed Denial of Service (DDoS) attacks, which are particularly common in private clouds and can lead to service degradation or denial. The authors proposed a solution that integrated an OpenStack firewall with a DDoS detection system, utilizing raw socket programming to monitor network traffic. Several algorithms, including Decision Tree, K Nearest Neighbor (KNN), Naive Bayes, and Deep Neural Networks (DNN), were compared using a dataset created from a controlled DDoS attack environment. The system successfully detected DDoS attacks and notified the private cloud administrator.

In [S. Rajagopal et al.,2021] tackled the challenges posed by sophisticated hacktivist attacks in network intrusion detection. They introduced a meta-classification approach using decision jungle for both binary and multiclass classification, optimizing hyperparameters and feature subsets through Azure machine learning. Their model was validated using several datasets, including UNSW NB-15, CICIDS 2017, and CICDDOS 2019, achieving accuracy rates of 99.8%, 98%, and 97%, respectively. This approach effectively identified thirty-three modern attack types, using a 40:60 train-test split for legitimacy assessment,

which differed from traditional stacking ensembles. Statistical tests compared the performance of various classifiers, and the automated model showed promise for real-time intrusion detection.

In [E. Arul and A. Punidha, 2021] examined vulnerabilities in cloud devices with weak defenses, often leaving users unaware of security compromises. The study focused on DDoS attacks targeting MemCached, a caching mechanism used to accelerate websites and networks. Hackers exploited insecure UDP MemCached servers by submitting spoofed applications that concealed their real IP addresses. The proposed method utilized Supervised SD-LVQ (Self-Organizing Map-based Learning Vector Quantization) to detect MemCached attacks across various cloud systems. The approach achieved a true positive rate of 97.23% and a false negative rate of just 0.03%, demonstrating the system's effectiveness in detecting DDoS attacks.

In [G. Sreelatha et al., 2022] explored the vulnerability of on-demand services to various network threats, which pose significant security and privacy challenges. Their solution included feature selection through sandpiper and the implementation of deep transfer learning. They used datasets like NSL KDD and UNSW NB15 for evaluation. Their simulations, conducted with a pre-trained AlexNet, resulted in a high detection rate and low false alarm rate, outperforming other methods in terms of detection efficiency.

[H. Ghani et al., 2023] investigated the use of a deep learning-based Feedforward Neural Network (FFNN) classifier to assess classification performance on the UNSW-NB15 and NSL-KDD datasets. Their study showed that while large feature sets could lead to unnecessary features, using a smaller feature vector improved classification accuracy. The approach achieved 91.29% accuracy on the UNSW-NB15 dataset and 89.03% accuracy on the NSL-KDD dataset, demonstrating its effectiveness in identifying network anomalies.

[Z. Long, 2024] proposed a novel Network Intrusion Detection System (NIDS) based on the Transformer model tailored for cloud environments. By integrating the Transformer's attention mechanism, the system improves the accuracy of intrusion detection by better analyzing the relationships between input features and attack types. Experimental results indicated that the Transformer-based model achieved over 93% accuracy, comparable to the CNN-LSTM model, demonstrating its potential to enhance cloud security.

[A.V. Songa and G.R. Karri, 2024] focused on early detection of attacks in Software-Defined Networking (SDN) switches, emphasizing the need for traffic clustering and anomaly prediction at each switch to identify potential DDoS attacks. They proposed event correlation to analyze network behavior and detect coordinated attack patterns. This approach addresses the limitations of existing methods, which often fail to provide early detection and lack integration for comprehensive threat analysis.

**Hybrid IDS:** Hybrid IDS in Cloud Computing refers to a security system that combines multiple techniques to detect malicious activity in cloud environments. Typically, it blends two main types of intrusion detection approaches: signature-based (which looks for known attack patterns) and anomaly-based (which identifies unusual or suspicious behavior).

By combining both methods, a hybrid IDS provides more accurate and comprehensive protection. The signature-based part quickly spots known threats, while the anomaly-based component can catch new, unknown attacks that haven't been seen before.

In the context of cloud computing, where resources are dynamic and distributed across different locations, hybrid IDS solutions are especially useful. They can monitor both network traffic and virtual machines, adapting to the flexible and scalable nature of cloud environments.

This combination helps reduce false positives and ensures better detection of complex threats, making cloud systems more secure.

### **Latest Trends Related to IDS in Cloud Computing:**

[R. R. Dewangan et al., 2025] presented that the cloud computing faces several prevalent security challenges. One major concern is data breaches, which may grant unauthorized individuals access to confidential information stored in the cloud. Such incidents can compromise privacy and confidentiality, potentially resulting in legal and financial consequences for both individuals and organizations. Additionally, the risk of data loss or corruption is significant, stemming from causes like hardware malfunctions, human mistakes, or malicious activity. Cloud systems are also vulnerable to distributed denial of service (DDoS) attacks, which can interrupt services and cause extended downtime for users. To address these issues, it is essential for organizations to adopt robust security strategies aimed at safeguarding cloud-based data and maintaining system integrity.

In [M. Younus et al., 2025] presented a systematic literature review aimed at exploring trends in cloud computing within the context of e-government. Employing a descriptive qualitative methodology combined with a bibliometric analysis, the research leverages the latest version of CiteSpace software to map the knowledge landscape in this field. The results highlight a rapidly evolving domain shaped by technological advancements and shifting governmental priorities. Worldwide, public administrations are increasingly recognizing the potential of cloud computing to enhance the efficiency, accessibility, and scalability of digital government services. Despite ongoing challenges—particularly around data security and privacy—the findings indicate a clear strategic movement toward adopting digital innovations to deliver more responsive and citizen-centric services.

[H. Park, 2025] presented an approach utilizes a Resilient Backpropagation Neural Network (RBN) to strengthen security and improve resilience by enabling real-time detection and mitigation of DDoS attacks across both network and application layers. To facilitate secure communication between containers, an Inter-Container Communication Bridge (ICCB) is integrated into the system. Additionally, it incorporates high-performance, low-latency technologies such as eXpress Data Path (XDP) and Extended Berkeley Packet Filter (eBPF) for efficient security enforcement, addressing the performance constraints of prior methods. This solution offers comprehensive protection against emerging cyber threats while preserving the flexible and scalable nature of cloud-native infrastructures. It also supports continuous operations through proactive threat monitoring and dynamic system adjustments, ensuring robust defense mechanisms without compromising the agility of modern cloud environments.

## 2.6 Comparative Analysis

Some of the research papers discussed above are presented in the following table 2.1 and table 2.2 which are summarizing the papers in the tabular form.

**Table 2.1 Comparison of related work to similarity measures**

<b>[Author, Year]</b>	<b>Similarity Measure Used</b>	<b>Purpose</b>
[Cao et al. ,2013]	Cosine Similarity, Jaccard Index	Facilitate multi-keyword ranked search in encrypted cloud data with privacy guarantees.

[Wang et al. ,2016]	Cosine Similarity, Weighted Euclidean Distance	Enable efficient and privacy-preserving searchable encryption over feature-rich data.
[Popic and Batzoglou, 2017]	MinHash, Cosine Similarity	Align simulated genomic reads with actual genome sequences for efficient cloud-based genetic analysis.
[Liang et al. ,2019]	Cosine Similarity, Clustering	Enhance network intrusion detection using multifeature data clustering optimization models.
[Miao et al. ,2020]	Cosine Similarity, Density-Based Clustering	Map simulated LiDAR data to real-world UAV point cloud data for obstacle detection and intrusion monitoring.
[Alshammari and Aldribi, 2021]	Jaccard Similarity, Cosine Similarity	Detect malicious network traffic by aligning synthetic traffic patterns with real-world network datasets.
[Li et al. ,2022]	Cosine Similarity, Clustering Algorithms	Align simulated and real-world datasets for anomaly-based intrusion detection, focusing on modern attack patterns.



**Table 2.2 Comparative Analysis of Literature Review Related to IDS**

<b>[Authors, Year]</b>	<b>Attacks Detected</b>	<b>Dataset Used</b>	<b>Methodology</b>	<b>Outcomes</b>	<b>Limitations/Future Scope</b>
[Y. Liu and R. Ma, 2013]	Various network attacks	KDD'99	Modified QPSO algorithm aligned with Bayesian network learning	Better convergence speed compared to existing models	Further evaluation on different datasets and real-world scenarios
[N.Pitropakis et al., 2014]	Co-residency and network stressing attacks	Controlled environment	Smith-Waterman genetic algorithm	Effective detection in kernel layer of KVM-based cloud environment; Method for identifying malicious insider attacks	Creating system call patterns as 'attack signatures' for IDS
[K.Wang et al., 2014]	Botnets	Windows Azure cloud service	Behavior-based botnet detection	High accuracy (true positive rate > 95%) with low false positive rate; Scalability demonstrated	Further scalability testing in larger cloud environments

				in cloud environment	
[N.Pandeeswari and Kumar ,2016]	Insider and outsider attacks in cloud computing	DARPA's KDD cup dataset (1999)	Hybrid algorithm combining Fuzzy C-Means clustering with Artificial Neural Network (FCM-ANN)	High detection accuracy and low false alarm rate for both insider and outsider attacks	Evaluation on more recent datasets and consideration of evolving attack vectors
[J. Hussain et al. ,2016]	Network Intrusion	NSL-KDD datasets	Two-stage hybrid classification method combining SVM and ANN	High detection rate (99.97%) with low false positive rate (0.19%)	Further validation on diverse network environments and attack scenarios
[H. H. Pajouh, 2017]	Network Anomalies	NSL-KDD dataset	Two-tier classification model incorporating Naïve Bayes, KNN, and Linear Discriminant Analysis	Enhanced detection rates and decreased false alarms; Efficient identification of complex attack types	Further evaluation on diverse datasets and real-world scenarios

[R. Kesavamoorthy and K. R. Soundar, 2019]	DDoS attacks in cloud computing	Simulated data	Autonomous multi-agent system with particle swarm optimization	Faster attack detection compared to other methods; Improved security	Evaluation under real-world DDoS attack scenarios; Scalability testing
[Z. Chiba et al., 2019]	Network Intrusions in Cloud Environments	Benchmark IDS datasets, CloudSim 4.0	Hybrid optimization framework (IGASAA) for building efficient DNN-based IDS	High detection precision and low false alarms; Outperformed existing methods	Further validation on diverse cloud environments and real-world datasets
[R. Patil et al., 2019]	Intrusions in Cloud Computing	UNSW-NB15, CICIDS-2017	Hypervisor Level Distributed Network Security (HLDNS) framework	Capability to meet cloud network security requirements; Intrusion detection and correlation across servers	Scalability testing on larger cloud networks and evaluation under diverse attack scenarios
[E.Besharati et al., 2019]	Virtual Machine Intrusions in Cloud Computing	NSL-KDD dataset, Cloudsim software	Host-based Intrusion Detection System (H-IDS)	Acceptable accuracy (approximately 97.51%) for detecting	Exploration of additional feature selection and classification techniques

				attacks against normal states	
[M. Jelidi et al., 2019]	Cloud Protection	Signature based and Anomaly based methods	Hybrid model combining distributed and centralized intrusion detectors	Effectiveness in protecting various cloud layers; Comprehensive framework for monitoring and managing cloud security	Integration with real-time threat intelligence and adaptive security measures
[P. Ghosh et al., 2019]	Various Network Intrusions in Cloud Environments	NSL-KDD dataset	CS-PSO-based IDS for quick and efficient attack classification	Effective IDS approach demonstrated on NSL-KDD dataset	Evaluation on diverse datasets and real-world cloud environments
[R. Rajendran et al., 2019]	Denial of Service (DoS) attacks in Cloud Networks	Cloud experimental setup with real-time DoS tools	Rule-based approach for DoS attack detection	Improved DoS attack detection accuracy compared to existing methods	Validation under various DoS attack scenarios and network configurations
[Cui et al., 2019]	Distributed Denial of Service (DDoS) attacks in	DDoS Attack 2007	Cognitive-inspired computing approach with dual	Rapid detection, high accuracy, low false positive rates, and	Investigation of scalability and performance under varying network conditions

	Software-defined networking (SDN)		address entropy	effective implementation of defense and recovery measures	
[A. N. Jaber and S. U. Rehman, 2020]	Various network intrusions in Cloud Computing	NSLKDD dataset	Hybrid IDS combining fuzzy c-means clustering (FCM) with support vector machine (SVM)	High accuracy and low false alarm rates compared to existing techniques	Evaluation on larger datasets and consideration of real-world cloud network environments
[D. J. Prathyusha and G.Kannayaram ,2020]	Distributed Denial of Service (DDoS) attacks in Cloud Computing	KDD cup 99 dataset	Artificial Immune System (AIS) based IDS for DDoS attack mitigation	High detection accuracy and low false alarm rate in cloud environments	Exploration of AIS performance under evolving DDoS attack techniques
[G. S. Kushwah and V. Ranga, 2020]	Distributed Denial of Service (DDoS) attacks in Cloud Computing	NSL-KDD, ISCX datasets	Voting Extreme Learning Machines (V-ELM) for DDoS attack detection	Strong accuracy in detecting attacks; Outperformed alternative methods	Evaluation on diverse cloud environments and consideration of evolving DDoS attack techniques

[L. Chen et al., 2020]	Network Intrusions in Cloud Computing Environment	Real-time network traffic data	C4.5 decision tree and random forest algorithms	Reduced training and testing times; High detection accuracy	Investigation of system scalability and performance under varying network conditions
[K. B. Virupakshar et al., 2020]	Distributed Denial of Service (DDoS) attacks in Private Clouds	Controlled DDoS attack environment dataset	Integration of OpenStack firewall with DDoS detection system	Successful detection of DDoS attacks and alerting administrator	Validation under real-world DDoS attack scenarios and consideration of additional attack vectors
[Rajagopal et al., 2021]	Network Intrusions in Computer Security	UNSW NB-15, CICIDS 2017, CICDDOS 2019 datasets	Meta-classification approach using decision jungle	High accuracies for detecting modern attack types; Promise for real-time intrusion detection	Further validation on larger and more diverse datasets; Real-world deployment and performance evaluation
[G. S. Kushwah et al., 2021]	Distributed Denial of Service (DDoS) attacks in Cloud Computing	NSL-KDD, ISCX IDS 2012, UNSW-NB15, CICIDS 2017 datasets	Enhanced Self-adaptive evolutionary extreme learning machine (SaE-ELM)	Superior detection accuracies compared to other techniques	Investigation of system scalability and performance under real-world DDoS attack scenarios

[E. Arul and A. Punidha ,2021]	DDoS attacks on MemCached in Cloud Computing	Prototype Malware Pool	Supervised SD-LVQ for MemCached attack detection	High true positive rate and low false negative rate for DDoS attack detection	Evaluation under diverse MemCached attack scenarios and consideration of additional attack vectors
[S. Velliangiri et al. ,2021]	Distributed Denial of Service (DDoS) attacks in Cloud Computing	KDD Cup Database Synthetic User-Generated Database and Cloud Server Log Database	Taylor-Elephant Herd Optimisation-based Deep Belief Network (TEHO-DBN)	Improved DDoS attack detection performance with TEHO-based DBN classifier	Investigation of TEHO-DBN performance under evolving DDoS attack techniques
[GSreelatha et al. ,2022]	Various network attacks in On-Demand Services	NSL KDD, UNSW NB15 datasets	Feature selection using sandpiper and deep transfer learning	High detection rate and low false alarm rate compared to alternative approaches	Evaluation under real-world network attack scenarios and consideration of additional datasets
[M. Otair et al.,2022]	Network Intrusions in Cloud Computing	NSL KDD dataset	Grey Wolf Optimization (GWO) and Particle Swarm Optimization	Superior outcomes in terms of false alarm rate, detection rate, detection	Investigation of GWO-PSO performance under diverse network attack scenarios

			(PSO) for feature selection optimization	accuracy, and execution time	
[Imran et al., 2022]	Network Intrusions in Cloud Computing	NSL-KDD dataset	Cuckoo search algorithm (CSA) integrated with k-means clustering for feature selection	Outperformed existing approaches in intrusion detection precision using CS	Evaluation on diverse cloud computing environments and consideration of real-time intrusion detection scenarios
[Eren et al., 2023]	Network Intrusions in UNSW-NB15 and NSL-KDD datasets	UNSW-NB15, NSL-KDD datasets	Combined machine learning and deep learning methods for attack detection	High accuracy rates for two-class and multi-class classification	Exploration of real-world deployment and scalability considerations
[H. Ghani et al., 2023]	Network Traffic Anomalies in UNSW-NB15 and NSL-KDD datasets	UNSW-NB15, NSL-KDD datasets	Feedforward Neural Network (FFNN) classifier with a small feature vector	Improved classification accuracy with reduced computational resources	Investigation of the approach's performance under diverse network conditions and additional datasets



[Z. Long et al., 2024]	Network Intrusions in CSE CIC IDS 2018 dataset	CSE- CIC IDS 2018 dataset	Seq2Seq (sequence-to-sequence) is used	Outperforms CNN with LSTM technique	Simulated dataset can be used for the evaluation.
[A. V. Songa and G. R. Karri,2024]	DDoS attack in CICD DoS 2019	CICD DoS 2019	RDAER model developed based on Recursive Feature Elimination (RFE), Density Based Spatial Clustering (DBSCAN) and time series techniques	Improved detection in less time	RDAER training and testing data can be tuned for improving the accuracy.
[W. A. H. Aljuaid and S. S. Alshamrani,2024]	Network intrusions in CSE-CIC IDS 2018	CSE-CIC IDS 2018	CNN -based model in which Multi-Blocks of CNN	Performs better than existing techniques	Simulated datasets can be used to enhance the performance of the model.

## 2.7 Problem Statement

In today's digital landscape, the rapid expansion of internet connectivity and digital technologies has led to a surge in cyber threats. These threats can result in

significant financial losses, tarnish organizational reputations, and disrupt essential services. Conventional security methods frequently fall short in addressing the increasingly advanced and dynamic nature of cyber threats, underscoring the importance of implementing strong Intrusion Detection Systems (IDS).

A key challenge in creating an effective IDS is accurately classifying network traffic to differentiate between legitimate and malicious activities. This task comprises two essential components: selecting the most relevant attributes from large, complex datasets and integrating appropriate classifiers to optimize detection performance.

Datasets employed for IDS, such as the CSE-CIC IDS 2018 dataset, contain a wide range of features that represent various attributes of network traffic. However, not all features equally contribute to the detection of intrusions. Irrelevant or redundant features can increase computational complexity and diminish detection accuracy. Therefore, the challenge is to identify and select the most pertinent features that provide valuable insights for detecting malicious activities.

Feature selection plays a critical role, as it directly affects the performance of the IDS. Effective feature selection reduces the dimensionality of the dataset, thereby enhancing both the efficiency and accuracy of the IDS by eliminating noise and irrelevant data. Selecting the optimal subset of features from a large pool requires sophisticated techniques to balance the trade-off between computational cost and detection performance.

Another significant challenge lies in integrating classifiers that can work together effectively to enhance the detection capabilities of the IDS. Different machine learning algorithms have unique strengths and weaknesses. For example, Decision Trees (DT) are valued for their interpretability and ability to manage categorical

data, Support Vector Machines (SVM) excel in high-dimensional spaces, and Neural Networks (NN) are adept at learning complex patterns. However, no single classifier is universally optimal for all types of network traffic and attack patterns.

By combining multiple classifiers into a hybrid model, it is possible to leverage the strengths of each to create a more robust and accurate detection system. The challenge here is to select and integrate these classifiers in a way that maximizes detection accuracy while minimizing false positives and false negatives.

Developing a theoretical model is not enough; it must be validated in a realistic environment to ensure practical applicability. The simulation architecture should replicate real-world network conditions and allow for extensive testing under various scenarios to evaluate the robustness and scalability of the IDS. A comprehensive simulation environment is essential for assessing the IDS's performance across multiple metrics. Ultimately, the architecture developed will yield a robust system for detecting attacks, thereby enhancing the security of cloud environments.

## **2.8 Research Gaps**

1.Many intrusion detection systems are trained and evaluated on outdated datasets that may not reflect real-world traffic [B. L. Farhan and A. D. Jasim, 2022].

2.Hybrid concept in feature selection module with hybrid classification module needs to be more explored.

3.Optimization algorithm like Firefly Algorithm (FA) can be explored by hybridizing with any classifier. FA is very efficient algorithm and, in some cases, represents Particle Swarm Optimization, Genetic Algorithm and Differential Evolution [I. Fister et al., 2013].

## **2.9 Summary**

This chapter offers a summary of recent research efforts focused on improving intrusion detection and security measures in cloud computing environments. Various methodologies have been explored by researchers, including traditional machine learning approaches, to develop effective intrusion detection systems (IDS) capable of identifying emerging threats. Additionally, the chapter discusses innovative approaches such as nature-inspired algorithms and hybrid detection systems that integrate multiple detection methods for comprehensive cloud protection. Through an examination of key studies and their findings, this chapter elucidates the evolving landscape of intrusion detection in cloud computing, highlighting the strides made towards bolstering security measures in this dynamic and critical domain. Research gaps are also observed from literature review.

## CHAPTER 3 RESEARCH METHODOLOGY

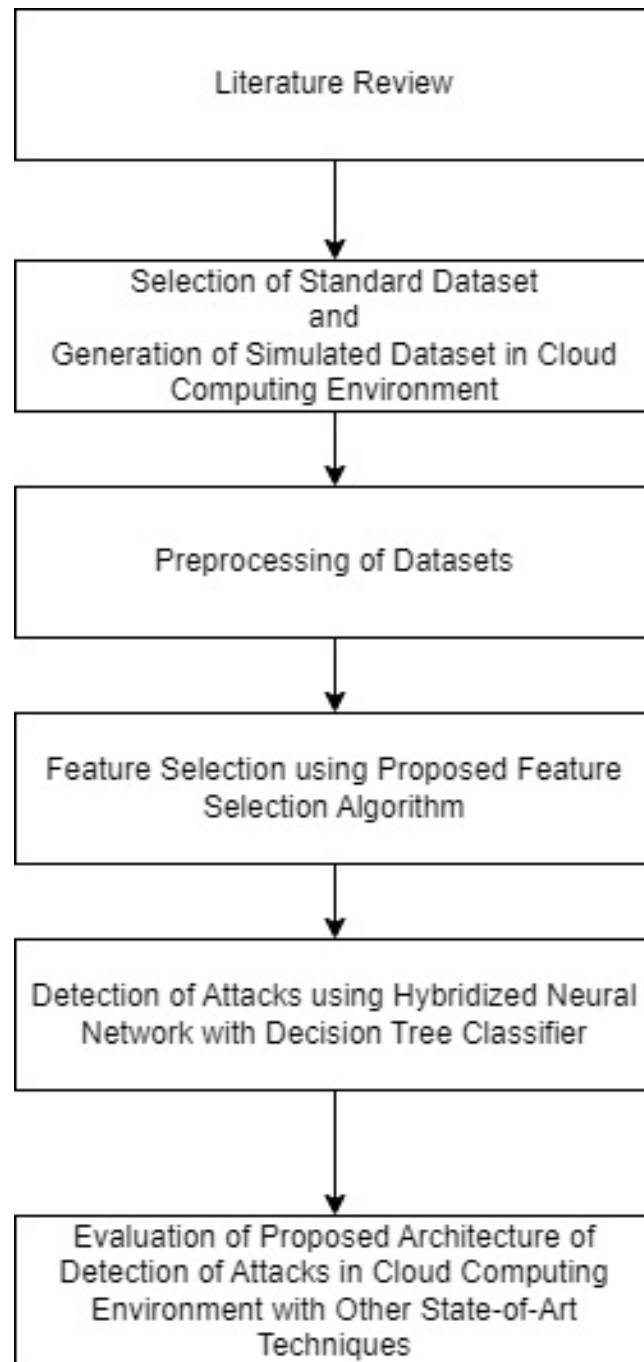
---

This chapter presents the research methodology used to accomplish the study's objectives. Research methodology refers to the organized approach adopted to address a research problem. It encompasses the systematic process a researcher follows, detailing the steps involved in investigating and solving the research problem.

**3.1 Research Objectives:** The research objectives are well-defined, clear, and measurable targets that a study seeks to accomplish. They establish the purpose of the research and specify what the researcher aims to investigate or uncover. The objectives of this research are as follows:

- 1.To study and analyze various attacks in the cloud computing environment.
- 2.To preprocess data obtained from various sources for attack detection.
- 3.To extract the relevant features for classification.
- 4.To design and implement a proposed architecture for the detection of various attacks in the cloud computing environment by using the optimized classifier.
- 5.To compare the proposed architecture with the existing ones.

**3.2 Research Methodology of Research Work:** The methodology used to achieve the research objectives is illustrated in Figure 3.1. The research work was achieved in various modules. These modules are described in the following section of this chapter.



**Figure 3.1 Research Methodology of Research Work**

**1.Literature Review:** When beginning the research process, of journals, as well as goo indexed published research papers and review papers were studied. Relevant academic journals, conference proceedings and books are explored. One source helped to get additional relevant sources. Prior studies that are similar to the current research are thoroughly reviewed and analyzed. A well-equipped library was accessed during this stage.

**2.Selection of Standard Dataset and Generation of Simulated Dataset in Cloud Computing Environment:** There are various methods for collecting relevant data, each differing significantly in terms of cost, time and available resources for the researcher. Dataset can be gathered through experimental or survey-based approaches. CSE CIC IDS 2018 dataset is selected after reviewing research papers and review papers. Cloudsim toolkit is used for generating simulated dataset which is collected by cloud computing environment simulation.

This rapid expansion of data demands more sophisticated security measures. Vulnerabilities in computer systems, poor security policies, and limited awareness of potential threats have made networks more prone to breaches. IDS can detect intrusions either through signature matching within network packets or by analyzing behavioural patterns.

**Table 3.1 Comparison of IDS Datasets**

<b>Dataset</b>	<b>Source</b>	<b>Attack Types</b>	<b>Labelling Details</b>	<b>Description</b>
<b>DARPA 1998</b>	Simulated network traffic	DoS, Probe, R2L, U2R	Detailed attack labels; some outdated scenarios	Early dataset; historical relevance

<b>KDD Cup 1999</b>	Simulated network traffic	DoS, R2L, U2R, Probe	Detailed attack labels; some redundant records	Widely used; criticized for outdated relevance
<b>NSL-KDD</b>	Refined KDD Cup 1999 dataset	DoS, R2L, U2R, Probe	Improved labelling, reduced redundancy	Enhanced version of KDD Cup 1999
<b>ISCX 2012</b>	Real-world network traffic (Canadian institution)	DoS, Brute Force, Web Attacks	Detailed attack and normal behavior labels	Focuses on specific attack types and normal traffic
<b>AFDaKyoto</b>	Real-world network traffic (Kyoto University)	DoS, DDoS, Probe, Malware, Information Gathering	Detailed attack labels and normal traffic	Comprehensive dataset with a wide range of attacks
<b>UNSW-NB15</b>	Real-world network traffic	Exploits, DoS, Generic, Fuzzers, Analysis	Detailed attack and normal behavior labels	Includes modern attack types and normal traffic
<b>CICIDS 2017</b>	Real-world network traffic (Canadian institution)	DDoS, Brute Force, Data Exfiltration, Botnet	Detailed labels for various attack types and normal traffic	Reflects modern attack scenarios



<b>CICIDS 2018</b>	Real-world network traffic (Canadian institution)	DoS, DDoS, Botnet, Malware, Web Attacks	Detailed attack labels and normal traffic	Up-to-date with current attack trends
------------------------	---	--	--	---

**3. Preprocessing of Datasets:** Dataset pre-processing involves transforming raw data into a more organized and efficient format for further analysis. During this phase, data is scaled using the min-max normalization technique, which adjusts all values to a consistent range, typically between 0 and 1. This normalization improves training efficiency by ensuring uniformity across the data. This step is crucial for various knowledge discovery tasks, such as network-based intrusion detection systems (NIDS), which classify network traffic as either normal or anomalous. Additionally, dataset sampling is applied to address data imbalance. An imbalanced dataset can result in a higher rate of false negatives, which negatively impacts the performance of the IDS.

**4. Feature Selection on Datasets using Proposed Feature Selection Algorithm:** Feature Selection (FS) tackles the issue of high-dimensional datasets by pinpointing the minimal subset of optimal features. FS is a critical preprocessing step in machine learning, commonly used for attack detection. Developing an efficient IDS relies on selecting relevant features that enhance attack detection capabilities. However, this process is complex due to the potential relevance, redundancy or excessiveness of features, which can increase the computational complexity of detecting attacks.

For years, researchers have been exploring optimal methods to improve accuracy and efficiency in this area. A key focus of recent research is enhancing the feature selection process by integrating additional algorithms into the learning model. One promising approach involves employing metaheuristic techniques as optimizers in conjunction with the learning model. New feature selection algorithm was proposed which is the hybridization of Firefly Algorithm (FA) with the Decision Tree. Decision Tree is used for finding the classification accuracy of the FA. This hybridization results in improving the performance of the FA.

FA was introduced by Xin-She Yang in 2008 and draws inspiration from the natural behaviour of fireflies, where their allure is influenced by the brightness of their flashes. Fireflies use their flashes to attract mates and communicate, which has inspired this optimization technique. In optimization problems, including feature selection, FA mimics the movement of fireflies as they search for optimal solutions. A firefly's brightness corresponds to its fitness value, with brighter fireflies representing better solutions.

Feature selection (FS) is a vital step in improving the accuracy and efficiency of classification tasks. A modified version of the Firefly Algorithm (FA) has been introduced for this purpose. In this enhanced method, a decision tree (DT) classifier is utilized to evaluate the performance (fitness) of each solution generated by the FA. The Firefly Algorithm was selected because it is not as widely used in FS applications, making it a novel choice in this context. According to research by H. S. Gebremedhin et al. (2020), FA has shown superior results compared to other popular optimization techniques like Particle Swarm Optimization (PSO) and Genetic Algorithms (GA). Interestingly, certain PSO variants can be viewed as specific forms of FA. Moreover, fine-tuning the FA's parameters can significantly improve its convergence speed and overall performance.

**5. Detection of Attack using Hybrid Classifier:** In cybersecurity, attack detection involves identifying various attacks that affect the integrity, confidentiality or availability of systems and data. Classification is performed on the datasets to detect the attacks. In many instances, enhancing overall classification accuracy can be accomplished by designing hybrid classification models that capitalize on the strengths of multiple algorithms.

A common approach is to combine the strengths of multiple classifiers to create a more accurate and robust model. In the proposed method, a hybrid model merges a Neural Network (NN) with a Decision Tree (DT) classifier. NN is a computational model inspired by the human brain, consisting of interconnected layers of artificial neurons that process and learn from data. These networks are trained to recognize patterns and extract features by adjusting the connections between neurons to improve accuracy. They are particularly effective at capturing complex, non-linear relationships between input and output variables, making them ideal for tasks like image and speech recognition, natural language processing, and both classification and regression. A typical NN architecture includes an input layer, one or more hidden layers, and an output layer. On the other hand, DT is a straightforward yet powerful model used for classification and regression tasks. It splits data into subsets based on feature values, forming a tree-like structure of decisions. Each internal node represents a decision based on a particular feature, while branches indicate the outcome of that decision. This process continues until the final prediction is made at the leaf nodes, which correspond to class labels or predicted values. Decision trees are intuitive and easy to interpret, mimicking human decision-making processes. They can handle both numerical and categorical data and model non-linear relationships between features.

In many situations, boosting classification accuracy can be achieved by developing hybrid models that integrate the advantages of multiple algorithms. A widely used

strategy involves merging different classifiers to build a more reliable and precise prediction system. In the current study, a hybrid classification approach is introduced by combining a Neural Network with a Decision Tree. This fusion aims to enhance overall performance by leveraging the Neural Network's ability to capture complex patterns and the Decision Tree's interpretability and efficiency. The combined model addresses the shortcomings of individual classifiers and delivers more accurate and understandable results for the classification task at hand.

## **6. Evaluation of Proposed Architecture for the Detection of Various Attacks:**

Evaluation is essential for checking the performance of the research work with existing state-of-the-art techniques. Evaluating an architecture using state-of-the-art techniques involves assessing its performance and effectiveness for the detection of attacks in comparison to the other techniques in the field. Popular techniques like Decision Tree, Support Vector Machine, Naïve Bayes are used for the comparison purpose.

Some Popular Classifiers which are widely used for dataset categorization:

**1. Support Vector Machine (SVM):** For data that is linearly separable, SVM identifies a hyperplane that efficiently separates the two classes. When the data is not linearly separable, SVM applies kernel functions to map the data into a higher-dimensional space, allowing the identification of an appropriate hyperplane.

**2. Naive Bayes (NB):** Naive Bayes (NB) is a group of probabilistic classifiers based on Bayes' theorem, typically used for classification tasks. It assumes that the features affecting the outcome are independent, which simplifies the computational process. Despite this "naive" assumption, it frequently delivers accurate results, particularly when working with large datasets. Naive Bayes is particularly well-suited for text classification tasks like spam detection and sentiment analysis, due to its efficiency and scalability.

**3. K-Means Nearest Neighbor (KNN):** KNN is a hybrid method that combines K-Means clustering with the K-Nearest Neighbors (KNN) algorithm to enhance classification and prediction results. The K-Means algorithm identifies clusters in the dataset by grouping data points into  $k$  clusters, based on their similarity to the centroids of these clusters. After the clustering process, the centroids are computed to represent the center of each cluster.

The performance of the proposed architecture is evaluated using the following metrics:

**1. Precision:** This metric indicates the proportion of predicted positive cases that are actually positive.

**2. Recall:** It is also known as Sensitivity or True Positive Rate; recall assesses the number of actual positive cases correctly identified by the model. A high recall indicates that the majority of true positives are accurately detected.

**3. Accuracy:** Accuracy reflects the overall correctness of the model by calculating the percentage of cases that were correctly predicted, encompassing both positive and negative outcomes.

**4. F-Measure:** The F-Measure, also known as the F1-score, is the harmonic mean of precision and recall. It serves as a balanced metric that accounts for both false positives and false negatives. This measure is especially valuable in scenarios where it's important to strike a balance between precision and recall, particularly when dealing with imbalanced class distributions.

These metrics collectively offer a thorough assessment of a model's performance, aiding in the evaluation of its effectiveness in classification tasks.

### 3.3 Key Challenges while implementing the IDS

The key challenges faced during IDS implementation:

#### 1. Resource Overhead

**Challenge:** IDS (especially host-based IDS) consume CPU, memory, and storage resources.

IDS monitors activities and changes within individual devices to detect suspicious behavior or potential threats. While effective in identifying internal security breaches, these systems can place a significant load on system resources. They often require continuous scanning, real-time analysis, and log generation, which can lead to increased CPU usage, memory consumption, and storage demands. The constant monitoring of files, processes, and system configurations may slow down system performance, especially on machines with limited hardware capabilities. Therefore, it's important to balance security needs with system efficiency when deploying HIDS.

#### 2. Tuning and Configuration

**Challenge:** IDS requires careful tuning to avoid excessive false positives or false negatives.

Intrusion Detection Systems (IDS) must be carefully configured to ensure accurate threat detection without overwhelming users with incorrect alerts. If not properly tuned, an IDS can generate excessive false positives, flagging normal activity as malicious, which can lead to alert fatigue and reduced trust in the system. On the other hand, insufficient sensitivity may result in false negatives, allowing real threats to go unnoticed. Achieving the right balance requires ongoing adjustment based on the network environment, typical user behavior, and emerging threat patterns. Regular updates and fine-tuning help maintain the IDS's effectiveness while minimizing unnecessary disruptions.

### **3. False Positives & Negatives**

**Challenge:** IDS can generate too many false alarms or miss real threats. Intrusion Detection Systems (IDS) can sometimes struggle with accurately identifying threats, leading to two major issues: false alarms and missed detections. When an IDS produces too many false positives, legitimate activities are mistakenly flagged as malicious, which can overwhelm security personnel and cause important alerts to be ignored. Conversely, if the system fails to recognize actual threats—resulting in false negatives—harmful activities may go undetected, putting the network at risk. These challenges highlight the importance of fine-tuning the IDS to ensure it effectively distinguishes between normal and suspicious behavior while maintaining a manageable alert volume.

### **4. Real-Time Processing**

**Challenge:** Detecting intrusions in real-time requires fast data ingestion and processing. High-volume traffic can overwhelm the IDS if it's not properly scaled. Real-time intrusion detection demands rapid collection and analysis of data to identify potential threats as they occur. To keep up with high-speed networks, an IDS must be capable of efficiently handling large volumes of traffic without delay. If the system lacks sufficient scalability or performance optimization, the surge in data flow can exceed its processing capacity, leading to delayed responses or missed threats. Ensuring the IDS is properly scaled and equipped with adequate computational resources is essential for maintaining its effectiveness in fast-paced, high-traffic environments.

### **5. Data Storage & Retention**

**Challenge:** IDS logs and alerts can generate massive amounts of data. Intrusion Detection Systems (IDS) continuously monitor network and system activities, producing a significant volume of logs and alerts in the process. Each event,

whether benign or suspicious, is recorded to provide a detailed account of activity for analysis and auditing. Managing, storing, and analyzing this information requires substantial resources and effective data handling strategies to ensure that important alerts are not lost in the noise and that the system remains efficient.

**3.4 Summary:** This chapter details the research methodology used to accomplish the research objectives, which is structured into several modules. The first module is the Literature Review, Selection of Standard Dataset and Generation of Simulated Dataset in Cloud Computing Environment, Preprocessing of Datasets, Feature Selection on Datasets, Detection of Attacks, Evaluation of Proposed Architecture for the Detection of Various Attacks. Every module has its own function and results in developing an efficient Intrusion Detection System.



## **CHAPTER 4 PREPROCESSING AND FEATURE SELECTION ON SIMULATED DATASET AND CSE CIC IDS 2018 DATASET**

---

The goal of data pre-processing is to transform raw data into a format that is more appropriate for analysis and processing. The missing values known as Not a Number (NaN) values are converted to 0 to prevent value errors in the systems. The preprocessing phase involves several essential operations to prepare datasets for training. Outliers and unwanted traffic are removed, while additional features are introduced to enhance the feature set, ensuring that classifiers achieve optimal detection performance. Noise and missing values are eliminated. Normalization is a scaling and mapping technique used in the preprocessing stage [A. L. Shalabi et al., 2006]. It is particularly beneficial for prediction and forecasting purposes. Normalization of the data using the Min-Max method helps to scale the values to a defined range, typically between 0 and 1. This step helps to enhance the training efficiency by ensuring that the data is on a consistent scale.

### **4.1.1 Generation and Preprocessing of Simulated Dataset**

#### **4.1.1.1 Generation of Simulated Dataset by Simulating Cloud Computing Environment**

Cloud environment simulators are utilized to explore, evaluate, and test these approaches under stress before implementing them in real-world settings. Simulations are helpful for testing the performance of the applications and it is not expensive [T. Goyal et al., 2012]. This model incorporates a virtualized cloud setup, where cloud resources are simulated using CloudSim environment which allows for the deployment of nodes that exchange data seamlessly through virtual machines. The CloudSim toolkit enables the modeling and creation of one or more

virtual machines (VMs) on a simulated data center node, along with jobs and their assignment to appropriate VMs [R. Buyya, 2010].

This toolkit follows a model which consists of cloud brokers and data centers [R. N. Calheiros, 2011]. CloudSim's management interfaces allow for the administration of data centers, virtual machines (VMs), and other components [D. M. Reddy, 2023]. By simulating real-world cloud conditions, the model ensures that the classifier is tested in scenarios that resemble actual cloud deployments.

Each VM in the cloud simulation communicates with other nodes to share data and workloads which simulates the cloud environment. The hybrid classifier processes the aggregated data, ensuring that tasks such as resource allocation, network throughput are optimized. By integrating the hybrid classification method into the simulation model, the study ensures a realistic and robust test environment that mirrors the complexities and challenges of cloud data analysis and service management.

#### **•Deployment model for generating simulated dataset**

The deployment of the cloud simulation has been done by using CloudSim which is a robust cloud simulation framework designed for modelling and simulating cloud computing environments. The objective of this deployment is generating a simulated dataset which represents the real-world cloud environments.

The configuration of the CloudSim Environment used in the simulation is described below:

#### **1.Data Center**

- The simulation model is equipped with one data center, which acts as the core of the cloud infrastructure. The data center manages resources, handles requests and oversees the operations of the virtual machines (VMs).

- The data center is designed to mimic the functioning of an actual cloud data center, including resource provisioning, allocation of computational tasks and monitoring of system performance.
- The Datacenter in this simulation represents the main cloud infrastructure, where all resources are managed and provisioned. It includes hosts and (VMs) to simulate real-world cloud environments.
- The datacenter is built with certain characteristics, such as architecture (x86), operating system (Linux), and virtual machine monitor (XEN), making it a multi-purpose infrastructure capable of handling various workloads.

## **2.Hosts (Physical Machines)**

Within the data center, ten hosts known as Physical Machines (PMs) have been deployed. These hosts represent physical servers that provide computational resources such as CPU, memory and storage. Each host is configured with adequate processing power and memory to support the virtual machines that operate on top of them. The hosts serve as the foundation for the virtualized resources which offers processing capabilities to handle diverse workloads that are generated by nodes during the simulation.

- Hosts in this setup are essentially physical servers that provide the computational power and memory for VMs. Each host is configured with a certain amount of RAM, bandwidth and storage.
- The amount of RAM, bandwidth, and storage for each host is generated randomly to simulate real-world variability in resource availability across different machines.
- The host contains processing elements (PEs), which represent the CPU cores. In this setup, each host has two CPU cores (PEs), and the computational power of each PE is defined using the `PeProvisionerSimple` class.

### **3.Virtual Machines (VMs)**

- The simulation incorporates 100 virtual machines (VMs) where each running on top of the ten hosts. VMs are used to simulate cloud resources that can dynamically handle workloads. Each VM has its allocated share of the host's resources which includes CPU and memory and is responsible for executing tasks that are submitted by the nodes.
- The VMs have been configured to simulate various workload types which provides a diverse testing ground for the hybrid classifier. The simulation allows the VMs to handle resource requests dynamically by adjusting their performance based on the workloads generated.
- Virtual Machines are the entities running on top of the hosts. Each VM runs independently and handles its allocated tasks. The hosts provide the computational power, memory, and bandwidth for the VMs.
- The VMs are assigned a certain amount of RAM, bandwidth, and storage from the host on which they reside.

### **4.Workload Generation**

- Nodes have been strategically generated within the simulation model to create and distribute workloads from various geographical locations. Each node represents a unique source of data or a client system that submits tasks to the cloud environment.
- The diversity in node locations helps to simulate the real-world cloud environment where data and requests originate from multiple locations which requires the cloud infrastructure to manage and allocate resources dynamically.
- The workloads generated by the nodes are distributed across the VMs, and the hybrid classifier is used to classify, process and optimize the handling of these workloads. The classifier helps to allocate resources efficiently and ensure that the cloud infrastructure performs optimally under different conditions.

## 5. Host List Generation

The hostlist is dynamically generated in this setup. The number of hosts (host\_count) is determined at runtime and for each host, RAM, bandwidth and storage are assigned using random values. This variability helps in simulating real-world environments where not all physical machines have the same resource capacity.

## 6. Costing

Each host is associated with certain costs:

**Compute cost per second:** The cost incurred for utilizing computational resources per second, randomly generated.

**Cost per memory:** The cost for consuming RAM resources, slightly randomized to simulate variable pricing.

**Cost per storage:** A fixed cost per unit of storage.

**Cost per bandwidth:** A fixed cost for bandwidth usage.

## 7. Provisioning

- PeProvisionerSimple provisions the processing power to the CPU cores (PEs). It ensures that the allocated CPU resources are properly managed during the simulation.
- RamProvisionerSimple class ensure the proper allocation and management of the memory and BwProvisionerSimple class ensure the proper allocation and management of bandwidth.

This architecture simulates a cloud environment using the CloudSim framework, where a datacenter is created with multiple hosts, and each host manages one or more Virtual Machines (VMs). The datacenter supports resource provisioning, VM scheduling, and task execution. Here are the key details:

### Attributes of Hosts

Each host has the following attributes:

- **RAM:** Randomly assigned between 0 to 8000 MB.
- **Bandwidth (BW):** Randomly assigned between 0 to 8000 Mbps.
- **Storage:** Randomly assigned between 0 to 100,000 MB.
- **Processing Elements (PEs):** Two PEs (cores), each with 1000 MIPS processing capacity.

**Table 4.1 Attributes of the VMs**

Attribute	Description	Value/Range
<b>VM ID</b>	Unique identifier for each VM	0 to 99
<b>MIPS</b>	Processing capacity	1000
<b>RAM</b>	Memory allocated to the VM	512 MB
<b>Bandwidth (BW)</b>	Network bandwidth	1000 Mbps
<b>Image Size</b>	VM image size	10,000 MB
<b>PEs</b>	Number of processing elements (cores)	1
<b>VMM</b>	Virtual Machine Monitor	XEN

**Table 4.2 Cloud Sim Configurations Used for Cloud Simulation**

Component	Description	Value/Details
<b>Architecture</b>	Defines the CPU architecture used in the cloud environment.	x86
<b>Operating System</b>	OS used by the hosts.	Linux
<b>VMM (Hypervisor)</b>	Virtual Machine Monitor that manages VMs.	XEN

<b>Timezone</b>	Time zone for the datacenter operations.	5.0
<b>Compute Cost/Sec</b>	Cost of using CPU resources per second, randomized for variability.	$3.0 * \text{random}()$
<b>Cost per Memory</b>	Cost of using memory resources, slightly randomized.	$1.0 + \text{random}()$
<b>Cost per Storage</b>	Fixed cost of storage utilization per unit.	0.05
<b>Cost per Bandwidth</b>	Fixed cost of bandwidth usage per unit.	0.10
<b>Hosts</b>	Physical servers providing resources (RAM, bandwidth, storage, CPU cores).	10 Hosts
<b>RAM</b>	Randomized value between 2000 and 8000 MB for each host.	2000-8000
<b>Bandwidth (bw)</b>	Randomized value between 0 and 8000 units for each host.	0-8000
<b>Storage</b>	Randomized storage capacity per host.	$100000 * \text{random}()$
<b>Processing Elements</b>	Number of CPU cores (PEs) assigned to each host.	2 cores per host (PE0 and PE1)

<b>VM Scheduler</b>	Scheduler used to allocate CPU resources to VMs running on the hosts.	VmSchedulerSpaceShared
<b>Virtual Machines</b>	Independent VMs running on hosts, each with allocated RAM, bw, and storage.	100 VMs
<b>RamProvisioner</b>	Allocates and manages RAM resources for each host.	RamProvisionerSimple(ram)
<b>BwProvisioner</b>	Allocates and manages bandwidth for each host.	BwProvisionerSimple(bw)

**Table 4.3 CloudSim Node Communication Simulation Algorithm**

---

**Algorithm 4.1: Node Communication Simulation Algorithm**

---

Initialize Parameters: Number\_of\_Nodes = n, Node\_Locations,  
Node\_Properties, Injection\_Rate=0, Number\_Of\_Simulations=m  
for i=1 to m  
    for j=1 to n  
        Set Route [0]= j  
        Compute Injection\_Rate= Injection\_Rate+200\* Random ()  
        Generate Random Probability  
        if Probability >0.3 then  
            Starttime=Systemtime  
            Set Destination = Random\_Destination\_Node  
            Computer distance between Node<sub>j</sub> and Destination  
            if Distance<300 then  
                Add Destination to Route

---



---

```

else
    Find Intermediate Nodes to Reach Destination
    while (Destination_Flag=0) do
        for Each Intermediate Node k do
            if k is not intermediate node and distance<300 then
                Add k to intermediate
                if Distance from k to Destination <300 then
                    Set Destination_Flag =1
                    Add Destination to Route
                end if
            end if
        end for
    end while
end if
Calculate Total Losses and Power Consumption for the Route
Compute Throughput and PDR
Store Results In Output Tabler
end if
end for
end for

```

---

In this simulation model, the nodes represent workloads rather than physical machines (PMs). Each node simulates a distinct workload that is processed within the cloud infrastructure which consists of PMs and VMs that manage these workloads. The workloads (nodes) are randomly generated with specific characteristics which includes the data volume, processing time and required resources.

The algorithm begins by generating random locations for each workload (node), representing different geographical or logical origins. Each workload is assigned specific properties such as the injection rate which is the rate at which the workload injects data into the system and processing requirements. The PMs in the data center serve as the physical infrastructure while the VMs hosted on these PMs handle the actual processing of the workloads. As workloads are introduced into the system, the algorithm dynamically routes them through the cloud infrastructure.

For each node, the algorithm calculates the most efficient route to a destination, which could be a service endpoint or another processing unit. In this context, the route represents how the workload travels through the cloud infrastructure, possibly being processed by multiple VMs across different PMs. If the workload is within a certain distance threshold of the destination, the VM handling the workload processes it directly. Otherwise, intermediate VMs on different PMs may be used to forward the workload, simulating a multi-hop transmission scenario where workloads are moved between different parts of the cloud infrastructure.

As workloads move through the cloud, VMs process the incoming data which handle computational tasks and forward the workload to the next VM or destination. The throughput of each workload is tracked to measure how quickly data is processed and delivered through the cloud system. The algorithm ensures that VMs efficiently handle workloads based on available resources like CPU, memory and bandwidth. The Packet Delivery Ratio (PDR) is another key metric used to measure the efficiency of the network in delivering workloads. PDR indicates the proportion of the workload that is successfully processed and transmitted to its destination.

To detect potential attacks in the network or to find the compromised route, deviations in service parameters such as throughput, PDR are monitored.

Anomalies like sudden drops in throughput or unexpected spikes in resource usage, particularly bandwidth, can indicate malicious activity. For example, if a VM exhibits a sharp decline in PDR while consuming significantly more bandwidth, this might signal a Denial of Service (DoS) attack or data interception attempt. Additionally, if certain VMs process workloads much slower than expected or use excessive CPU resources without corresponding workload increases, it could indicate that the VM has been compromised.

The algorithm also calculates the power consumption for each VM based on the workload being processed. This information is crucial for optimizing cloud resource utilization and ensuring that the infrastructure remains energy-efficient while handling diverse workloads. Moreover, abnormal spikes in power consumption may also signal malicious behavior, where compromised VMs engage in unauthorized computational tasks. By integrating these parameters, the system can flag potential attacks. Ultimately, the simulation provides a detailed analysis of how efficiently workloads are processed and delivered in a cloud environment and also, tracking key metrics such as throughput, PDR and power consumption.

The security and robustness of the cloud infrastructure in managing workloads in the simulation includes a step where the aggregated data from workload processing is analyzed using k-means clustering. This method enables the system to categorize data into three clusters based on key performance metrics, such as throughput, Packet Delivery Ratio (PDR) and Power Consumption. The data collected from workload nodes represents different processing demands and the locations are aggregated and fed into the k-means algorithm to group the workloads based on their similarities and differences.

The k-means clustering algorithm divides the aggregated data into three distinct categories where each representing a different level of system performance. Once

the data is categorized, each cluster is further evaluated based on the standard deviation (STD) of its performance metrics. The standard deviation is a critical indicator of how much variability there is within each cluster.

Clusters with a high standard deviation represent workloads where performance metrics such as throughput and PDR, vary significantly, suggesting that the system is less consistent and less robust in managing these workloads. In practice, this could mean that some workloads are processed quickly and securely, while others experience delays, packet loss and higher energy consumption. A high standard deviation signals potential security risks or inefficiencies, as the system may struggle to maintain consistent performance under these conditions. These clusters with high variability are labelled as least robust, indicating that the cloud infrastructure is less reliable for certain workloads.

On the other hand, clusters with a low standard deviation are indicative of a more robust system. Here, the performance metrics are more tightly clustered which means that the system processes workloads consistently, with the minimal variations in throughput, power consumption and PDR. A low standard deviation demonstrates that the cloud infrastructure is capable of delivering stable and secure processing across different workloads, efficiently managing resources and ensuring that all workloads are handled securely and effectively. These clusters are labelled as highly robust, highlighting the system's ability to maintain secure, efficient operations, even under variable workload conditions.

After categorizing the aggregated data into three clusters using k-means clustering—representing varying levels of robustness based on standard deviation (STD), the next step involves further classification of this categorized data using a hybrid classifier. The hybrid classifier integrates multiple classification techniques, combining the strengths of different algorithms to achieve higher accuracy and

reliability in classifying the data. This classifier is designed to ensure that the system can accurately classify workloads into one of the three categories: highly robust, moderately robust, or least robust, based on the system's ability to process them efficiently and securely.

The hybrid classifier processes the clustered data by taking into account the performance metrics from each category, such as throughput, power consumption and Packet Delivery Ratio (PDR). These metrics help the classifier in determining the overall robustness of the system for each workload. The goal of the hybrid classifier is to maintain high overall classification accuracy, ensuring that the system can reliably identify workloads that fall into the least, moderate, or highly robust categories.

- **Potential Biases for simulated dataset generation**

Potential Biases while generating simulated datasets can be:

- 1. Simulation Configuration Bias**

We used specific simulation parameters (e.g., VM specification, datacenter models) that don't reflect real-world cloud environments.

- 2. Workload Bias**

Datasets may be generated using synthetic workloads that do not match real user behavior or applications.

- 3. Algorithmic Bias**

Research might test a specific scheduling or load-balancing algorithm and report performance improvements that don't generalize.

- 4. Time Scale & Duration Bias**

Short simulation durations may not reveal long-term performance trends or anomalies.

## 5. Overfitting to Simulation

Some datasets might be tuned or generated to demonstrate high performance of a specific approach, overfitting to the simulated environment.

### 4.1.1.2 Mapping of simulated data to actual dataset

In the domain of network security and anomaly detection, the classification of data into attack and non-attack categories forms the cornerstone of model development. While simulated data provides a controlled environment for experimentation, its effectiveness hinges on its fidelity to real-world scenarios. Simulated data often lacks the complexity and variability of actual data, making it prone to mislabelling when relied upon without cross-referencing. This necessitates mapping simulated data to actual data to refine the labelling process. Below, we explore the importance, challenges, and benefits of this mapping process in depth, alongside its implications for improving precision and reliability.

Labels define the underlying structure of datasets used for training and evaluation of machine learning and deep learning models. A minor error in labelling can propagate through the entire pipeline, leading to systemic issues such as:

- **Model Degradation:** Incorrect labels degrade the performance of models, as they learn patterns based on false information.
- **Overfitting or Underfitting:** Mislabelling can create noise that forces models to overfit irrelevant patterns or underfit essential ones.
- **Operational Failures:** Systems trained on poorly labelled data risk failing in real-world scenarios, leading to security breaches or missed detections.

Accurate labelling is particularly critical in domains like cybersecurity, where real-world attack patterns are nuanced, dynamic, and often multi-faceted.

Statistical methods, while valuable, provide a limited view of data characteristics. Techniques such as Mean Squared Error (MSE), variance analysis, and deviation

thresholds are commonly applied for initial labelling of simulated data. However, these methods face several limitations:

**1. Lack of Contextual Awareness:**

- Statistical methods evaluate data in isolation, focusing solely on numerical patterns.
- Attack signatures often depend on complex, multi-dimensional patterns that exceed simple statistical descriptions.

**2. High Sensitivity to Thresholds:**

- Choosing the correct threshold for labelling can be arbitrary and may vary depending on the dataset. A misaligned threshold can either over-label benign data as attacks (false positives) or under-label actual attacks (false negatives).

**3. Ambiguity in Complex Scenarios:**

- Certain attack types, such as Advanced Persistent Threats (APTs) or blended attacks, exhibit subtle patterns that evade detection by statistical methods alone.

**4. Static Nature of Statistical Approaches:**

- Real-world environments are dynamic, with attack patterns evolving continuously. Statistical methods, unless frequently updated, fail to adapt to these changes.

Statistical error-based methods rely on thresholds to classify data points. For instance, a high MSE value might indicate an anomaly, while a low MSE suggests normal behaviour. These methods are straightforward and computationally efficient, making them suitable for initial labelling in large datasets. However, they come with inherent limitations. Statistical methods lack contextual awareness, which is crucial for distinguishing between attacks and benign anomalies that might appear similar in numerical terms but differ significantly in their underlying causes. For example, a legitimate surge in network traffic during an online sale could be

mistaken for a Distributed Denial of Service (DDoS) attack if labelled purely based on packet volume thresholds. Such mislabelling can lead to high false-positive rates, eroding the reliability of the resulting system.

Mapping simulated data to actual data addresses these shortcomings by introducing real-world context into the labelling process. Actual datasets, typically derived from real-world scenarios, come with ground truth labels validated by experts or trusted frameworks. These labels capture the nuances of real attack patterns, including subtle behaviours that might not manifest in a simulation. By aligning simulated data with these benchmarks, researchers can refine their labels, ensuring greater accuracy and consistency. For instance, a simulated DDoS attack might be labelled based on its high packet volume, but mapping it to actual data might reveal additional characteristics such as synchronized timing or geographic distribution, enriching the dataset's fidelity.

This mapping process also enhances the generalizability of models trained on the labelled data. Models trained solely on simulated data risk overfitting to the specific patterns of the simulation environment. These patterns might not translate well to real-world scenarios, where attacks are more diverse and unpredictable. Mapping simulated data to actual data bridges this gap, ensuring that the resulting models perform reliably across a broader range of conditions. For example, a model trained on purely simulated phishing email data might struggle to identify sophisticated real-world phishing attempts that use social engineering tactics. Mapping to actual datasets helps incorporate these subtleties, making the model more robust and adaptable.

Another advantage of mapping simulated data to actual data lies in reducing false positives and false negatives. Statistical methods, while efficient, often lead to overgeneralizations. They might classify all high-MSE data points as attacks, even



if some are benign anomalies. Conversely, they might miss attacks with subtle signatures that don't trigger the predefined thresholds. Mapping provides a validation layer, cross-referencing these classifications against real-world examples to correct such errors. For instance, simulated data with high MSE might be flagged as a potential attack, but mapping to actual data could confirm whether it aligns with known attack patterns or is simply noise.

While statistical labelling methods provide a good starting point, combining them with mapping creates a hybrid approach that leverages the strengths of both techniques. Statistical methods quickly identify potential anomalies, which can then be validated and refined through mapping. This iterative process ensures that the final labels are both accurate and contextually relevant. Moreover, insights gained from mapping can inform adjustments to statistical thresholds, making them more effective in future labelling efforts. For example, if mapping reveals that certain high-MSE cases consistently align with benign behaviours, the thresholds can be adjusted to avoid similar false positives in the future.

Mapping simulated data to actual data is particularly critical in dynamic environments where attack patterns evolve over time. Simulations are often static, reflecting a snapshot of potential attack behaviours based on known scenarios. However, real-world data captures the evolving nature of cyber threats, such as new forms of malware or blended attack strategies that combine multiple vectors. By mapping simulated data to actual datasets, researchers can ensure that their simulations remain relevant and aligned with current threat landscapes. For example, a simulation might model a brute force attack based on fixed parameters, but mapping to real-world data could reveal new variations, such as adaptive algorithms that change attack patterns based on system responses.

Despite its advantages, mapping simulated data to actual data is not without challenges. One major hurdle is the availability of high-quality, labelled datasets.

Real-world datasets are often proprietary or subject to privacy restrictions, limiting their accessibility. Even when such datasets are available, ensuring compatibility with simulated data can require extensive preprocessing and transformation. For instance, simulated data might use simplified features or metrics that don't directly align with the richer, more complex attributes of actual datasets. Addressing these challenges requires careful planning and resource allocation, but the benefits of improved labelling accuracy and model reliability far outweigh the costs.

The computational overhead of mapping can also be significant, especially for large datasets. Simulated data often contains millions of records, and cross-referencing each record with actual data can be time-consuming and resource-intensive. However, advances in automated mapping techniques, such as the use of machine learning to align and compare datasets, have made this process more efficient. For example, clustering algorithms can group similar data points from simulated and actual datasets, streamlining the mapping process and reducing manual effort.

In conclusion, mapping simulated data with actual data is an essential step for achieving precise and reliable labelling in cybersecurity research. While statistical error-based methods provide a useful starting point, they fall short in capturing the complexity and variability of real-world attack patterns. Mapping addresses these gaps by introducing real-world context, improving generalizability, and reducing false classifications. The hybrid approach of combining statistical and mapping methods ensures that labelled datasets are both accurate and adaptable, enabling the development of robust models capable of meeting the challenges of evolving cyber threats. Despite the challenges involved, the benefits of mapping far outweigh the drawbacks, making it an indispensable practice in the pursuit of high-quality, actionable datasets.

Mapping simulated data to actual data is a multi-step process involving the application of various similarity measures to align and compare the features of the two datasets. The goal is to refine the simulated data's labelling by benchmarking it against actual data, ensuring that it captures real-world patterns effectively. Several methods for mapping exist, each with its strengths and limitations. This illustration explores similarity measures, their justification, and the importance of normalization in enhancing their effectiveness.

Mapping involves identifying relationships between the simulated and actual data points. This is typically achieved by quantifying how similar the data points are across one or more dimensions. The key methods for mapping include feature matching, similarity measures, dimensional reduction and projection, and cluster alignment. Similarity measures play a central role in these approaches, offering a mathematical basis for comparing datasets.

- **Key Similarity Measures for Mapping**

Among the many similarity measures available, the most common ones include Euclidean distance, cosine similarity, and hybrid similarity measures. These measures provide a robust foundation for aligning simulated and actual data, ensuring precision and reliability in the mapping process.

### **1. Euclidean Distance**

Euclidean distance is calculated as the straight-line distance between two points in multi-dimensional space:

$$d(p, q) = \text{sqrt}(\text{sum}((p_i - q_i)^2)) \quad (1)$$

It is suitable for continuous numerical spaces but is sensitive to scale differences, making it less effective for high-dimensional data.

## 2. Cosine Similarity

Cosine similarity measures the cosine of the angle between two vectors in a multi-dimensional space:

$$\text{Cosine Similarity}(A, B) = \frac{(A \cdot B)}{(\|A\| \|B\|)} \quad (2)$$

It is robust to magnitude differences, ideal for high-dimensional data, and computationally efficient. This makes it particularly useful for sparse datasets like text data represented as vectors.

## 3. Hybrid Similarity Measures

Hybrid measures combine the strengths of different similarity techniques, such as a weighted combination of cosine similarity and Euclidean distance. An example formula :

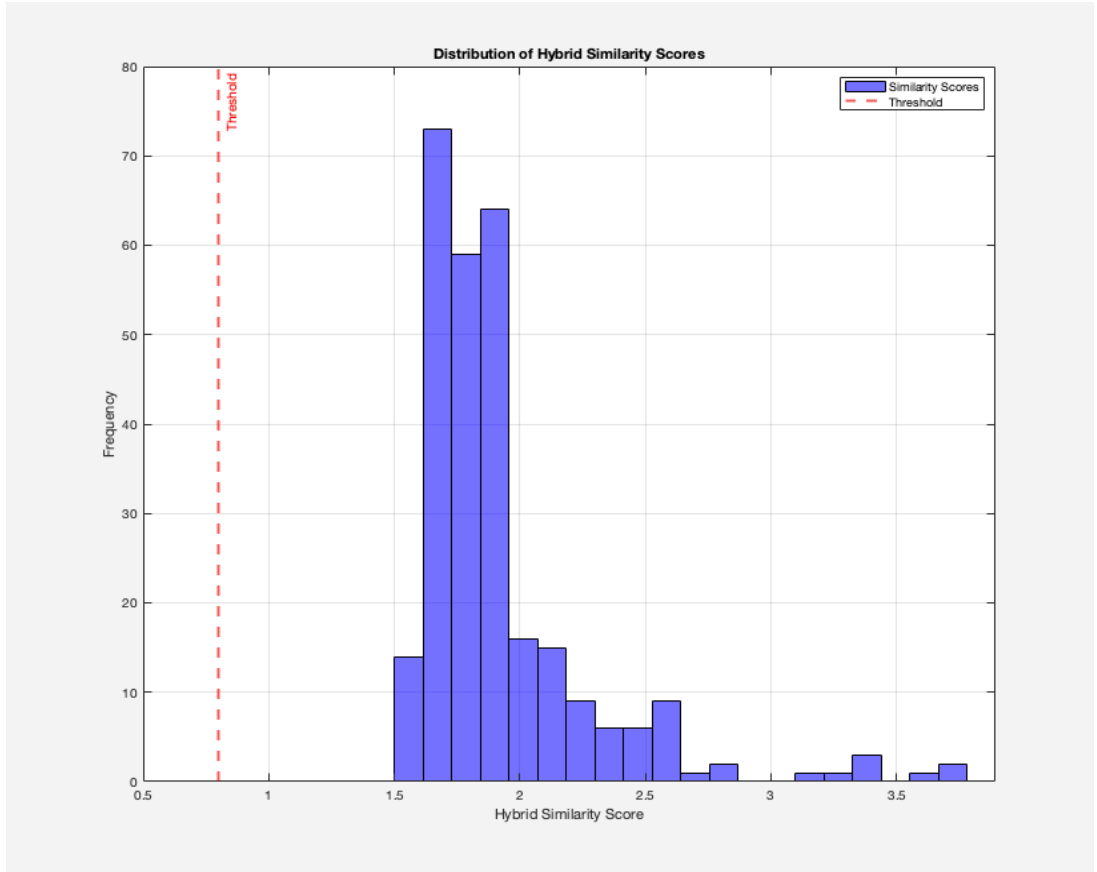
$$\text{Hybrid Similarity}(A, B) = w1 * \text{Cosine Similarity}(A, B) + w2 * (\text{Eucl}) \quad (3)$$

These methods offer enhanced flexibility and precision by leveraging multiple perspectives on data similarity. Cosine similarity is particularly effective for mapping simulated data to actual data due to its robustness to magnitude differences, suitability for high-dimensional data, and ability to align with real-world characteristics. It ensures computational efficiency while capturing directional alignment, which is critical for tasks involving complex patterns like cybersecurity.

Normalization is crucial for eliminating scale bias, improving robustness, and enabling comparability. Common techniques include Min-Max scaling, Z-score normalization, and L2 normalization. Normalizing data ensures that similarity measures like cosine similarity operate effectively, capturing the true relationships between data points.

The loading of the simulated dataset and the CSE-CICIDS dataset is performed initially. The CSE-CICIDS dataset is a well-known benchmark dataset that contains labelled records, with the last column specifying the labels (e.g., "Normal," "DDoS"). The simulated dataset, while feature-compatible, lacks these labels. To enable meaningful comparisons, feature alignment is performed, retaining only common features between the two datasets. This ensures consistency in the feature space used for similarity computation. Following alignment, Min-Max normalization is applied to scale all features to a range between 0 and 1. This step is crucial because features like packet size or flow duration may differ significantly in scale, leading to biased similarity calculations. By normalizing the data, the algorithm ensures that all features contribute equally, avoiding domination by larger-scaled features.

The core of the algorithm lies in the computation of hybrid similarity for mapping the simulated data points to the CSE-CICIDS dataset. For each simulated data point, cosine similarity is computed to measure the directional alignment of feature vectors, which captures the pattern similarity between the data points. Additionally, Euclidean distance is calculated to assess the magnitude difference, ensuring that proximity in feature space is considered. Since Euclidean distance is inherently a measure of dissimilarity, it is normalized to a similarity score ranging from 0 to 1 by subtracting it from the maximum possible distance in the dataset. These two measures are combined into a hybrid similarity score, calculated as a weighted average where the parameter  $\alpha$  balances the importance of cosine similarity and normalized Euclidean similarity. This hybrid similarity approach provides a comprehensive way to compare data points, leveraging both direction and magnitude for precise mapping. The label of the closest CSE-CICIDS data point, as determined by the highest hybrid similarity score exceeding a threshold ( $\theta$ ), is assigned to the simulated data point.



**Figure 4.1 Calculation of similarity measures and threshold**

After mapping, the algorithm appends each simulated data point with its assigned label and hybrid similarity score, producing a labelled dataset. This enriched dataset is saved to an Excel file, creating a permanent record of the mapping process. To evaluate the effectiveness of the mapping, the algorithm generates a histogram of the hybrid similarity scores.

This visualization helps in understanding how well the simulated data aligns with the CSE-CICIDS dataset, with the threshold line distinguishing between points that were successfully mapped and those that fell below the similarity requirement. The histogram provides insights into the overall distribution of similarity scores,

identifying potential areas where parameter tuning (e.g., adjusting  $\alpha$  or  $\theta$ ) might improve the mapping accuracy. By combining precise computation, effective labelling, and detailed visualization, the algorithm ensures that the simulated data can be reliably used for downstream tasks like model training or anomaly detection in a realistic context.

**Table 4.4 Result of Similarity Calculation**

Route	Throughput (MBPS)	PDR	ConsumedPower (mJ)	Label	Hybrid_Similarity_Score
26	7.69E-12	0.0904	1.30E+04	Bruteforce	3.7137
10	3.28E-11	0.3482	1.42E+04	Bruteforce	3.1438
25	6.44E-12	0.0659	1.47E+04	Bruteforce	3.7673
14	2.49E-11	0.2283	1.68E+04	Bruteforce	3.3832
3	2.51E-11	0.2146	1.75E+04	Bruteforce	3.4105
40	3.51E-11	0.2737	1.96E+04	Bruteforce	3.2733
35	3.09E-11	0.2258	2.08E+04	Bruteforce	3.3726
39	9.27E-11	0.6629	2.08E+04	Bruteforce	2.5866
27	7.38E-11	0.5045	2.18E+04	Bruteforce	2.8154
44	1.73E-11	0.1139	2.26E+04	Bruteforce	3.6254
20	8.42E-11	0.5304	2.36E+04	Bruteforce	2.7614
22	1.26E-10	0.7659	2.45E+04	Bruteforce	2.4544
28	1.12E-10	0.633	2.62E+04	Bruteforce	2.5876
8	1.54E-10	0.845	2.72E+04	Bruteforce	2.3939
22	1.31E-10	0.6877	2.86E+04	Bruteforce	2.5001
36	1.30E-10	0.6642	2.89E+04	Bruteforce	2.5262
43	1.22E-10	0.5904	3.05E+04	Bruteforce	2.6219
15	1.38E-10	0.6499	3.14E+04	Bruteforce	2.5283
16	1.21E-10	0.5414	3.35E+04	Bruteforce	2.6847

The proposed work focused on validating the effectiveness of the hybrid similarity-based mapping process by training a Pattern Recognition Network (PatternNet) to

classify simulated data labelled using the CSE-CICIDS dataset. The PatternNet, a specialized neural network for classification tasks, was designed with three hidden layers containing 128, 64, and 32 neurons, respectively. These layers were chosen to allow the network to learn complex patterns and subtle relationships within the labelled dataset. The dataset, prepared from simulated data mapped to CSE-CICIDS labels, was divided into 70% training and 30% testing subsets to ensure robust evaluation and to prevent data leakage between the training and testing phases. To enable the PatternNet to handle multi-class classification tasks, the class labels were transformed into one-hot encoded vectors, which represent each class as a binary vector.

During the training process, the network was optimized using the Adam optimizer with a learning rate of 0.01, targeting a mean squared error goal for 200 epochs. Training progress was monitored through the generation of performance and training state plots, which provided insights into the network's learning curve, including the reduction of loss over epochs and validation accuracy trends. These plots confirmed that the network converged effectively, demonstrating stability in its learning process. By leveraging a diverse set of labelled features generated during the mapping process, the PatternNet was expected to validate the accuracy and generalizability of the proposed mapping approach. Once trained, the network was evaluated using the unseen test subset to assess its classification accuracy, generating predictions that were compared to the true labels in the test set.

To provide a detailed evaluation, a confusion matrix was created to analyze the network's classification performance for each class. The confusion matrix highlighted true positive, false positive, and false negative rates for every class, offering granular insights into how well the network distinguished between different categories of traffic patterns, including normal and malicious activities. A heatmap was plotted to visualize the confusion matrix, making it easier to interpret the classification results. However, during this step, it was observed that some



classes from the CSE-CICIDS dataset were absent in the test subset, resulting in a mismatch between the dimensions of the confusion matrix and the total number of expected categories. This issue was addressed by expanding the confusion matrix to include all categories from the dataset, ensuring that even missing classes were represented in the final matrix. The confusion matrix was then normalized to show percentages, enabling a clearer assessment of classification performance across all classes. These measures ensured that the evaluation remained comprehensive, validating the mapping process and the capability of the PatternNet to generalize effectively to unseen data. The results of this work highlight the robustness of the proposed methodology, emphasizing its potential for applications in real-world network intrusion detection systems.

#### 4.1.1.3 Parameters analysis for analyzing the impact of attacks on the system

There are many parameters which are affected when any attack occurs on the cloud. Some key parameters are analyzed when DDoS attack and User to Root attack were deployed on the cloud environment.

**1. CPU Utilization:** CPU utilization represents the proportion of time the central processing unit (CPU) spends actively executing tasks or processing instructions. This metric is vital for assessing the efficiency and overall performance of a computer system. By tracking CPU utilization, organizations can uncover performance bottlenecks, resource limitations and potential security vulnerabilities. In cybersecurity, monitoring CPU usage takes on added importance as it can aid in detecting malicious activities.

$$\text{CPU Utilization} = \frac{\text{Busy Time of CPU}}{\text{Total Active Time of CPU}} * 100 \quad (4)$$

**2. Bandwidth Utilization:** Bandwidth utilization refers to the ratio of the amount of data being transmitted over a network to the maximum bandwidth capacity of

that network. It is an important metric for assessing network performance and efficiency.

$$\text{Bandwidth Utilization} = \frac{\text{Amount of Data Tranferred}}{\text{Total Bandwidth of Network}} * 100 \quad (5)$$

**3.CPU Load:** CPU load refers to the amount of computational work that a CPU is currently handling. It is a measure of how busy the CPU. This includes the count of tasks that are currently running and the number of tasks waiting to be processed.

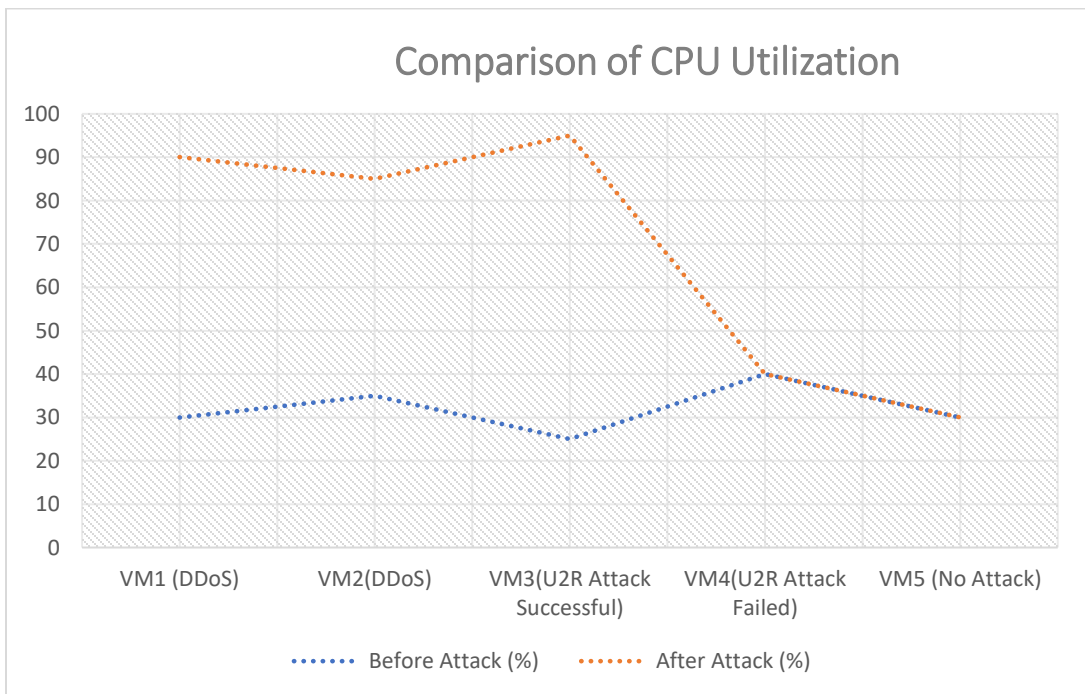
$$\text{CPU} = \text{Number of Tasks Running} + \text{Number of Tasks Waiting} \quad (6)$$

**Table 4.5 Parameters Analyzed Before Attack**

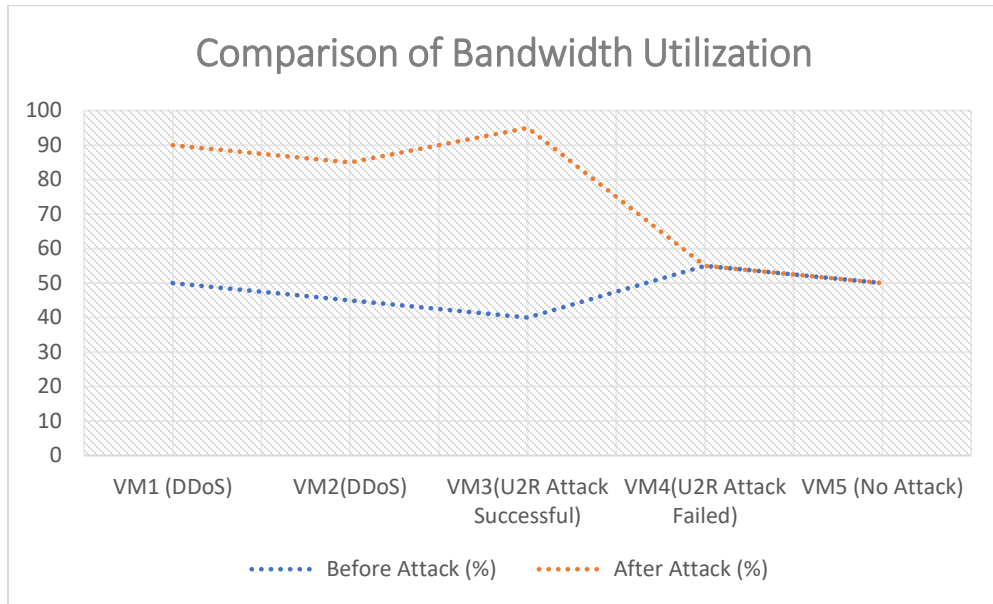
Parameter	VM 1	VM 2	VM 3	VM 4	VM 5
<b>CPU Utilization (%)</b>	30%	35%	25%	40%	30%
<b>Bandwidth Utilization (%)</b>	50%	45%	40%	55%	50%
<b>Load Distribution ((Number of Tasks)</b>	2	2	1	2	2

**Table 4.6 Parameters Analyzed After Attack**

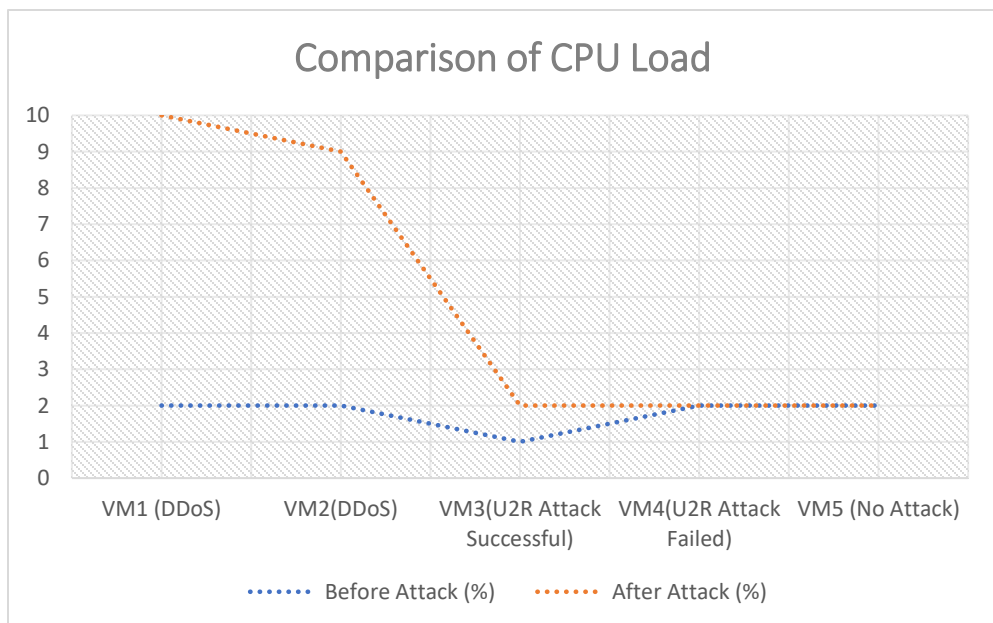
Parameter	VM 1 (DDoS)	VM 2 (DDoS)	VM 3 (U2R Successful)	VM 4 (U2R Failed)	VM 5 (No Attack)
<b>CPU Utilization (During Attack)</b>	90%	85%	95%	40%	30%
<b>Bandwidth Utilization (During Attack)</b>	90%	85%	95%	55%	50%
<b>Load Distribution (Number of Tasks)</b>	10	9	2	2	2



**Figure No. 4.2 Comparison of CPU Utilization**



**Figure No. 4.3 Comparison of Bandwidth Utilization**



**Figure No. 4.4 Comparison of CPU Load**

The tables provide a detailed overview of the behavior of CPU utilization, bandwidth utilization, and load distribution across VMs in a cloud environment before, during, and after the simulation of DDoS and U2R attacks. Before the attacks, the cloud infrastructure operates in a balanced state. Each virtual machine (VM) shows moderate CPU utilization: VM 1 at 30%, VM 2 at 35%, VM 3 at 25%, VM 4 at 40%, and VM 5 at 30%. Bandwidth utilization also reflects a stable environment, with VMs operating between 40% and 55% bandwidth usage, where VM 1 and VM 5 show 50% utilization, VM 2 at 45%, VM 3 at 40%, and VM 4 experiencing slightly higher bandwidth at 55%. The load on the VMs is distributed with 1 or 2 tasks allocated to each, indicating a well-managed and efficiently running system.

**4.Security Efficiency (SE):** Security Efficiency measures the system's overall performance in detecting attacks while minimizing false alarms. It is defined as:

$$SE = \frac{True\ Positives\ (TP) - False\ Positives\ (FP)}{Total\ Instances} \quad (7)$$

A higher SE indicates a more reliable detection system that can accurately differentiate between malicious and benign traffic

**5.Throughput (T):** Throughput reflects the system's efficiency in processing data (packets or instances) per unit time, critical for real-time applications. It is expressed as:

$$T = \frac{Total\ Packets}{ProcessedTime(s)} \quad (8)$$

Improved throughput signifies enhanced scalability and responsiveness of the system.

The results before and after implementing the proposed methodology are shown in the table below:

**Table 4.7 Parameter Analysis Before and After Implementation of IDS**

<b>Parameter</b>	<b>Before Implementation</b>	<b>After Implementation</b>	<b>Improvement (%)</b>
<b>Security Efficiency (SE)</b>	72.5%	90.2%	+24.14%
<b>Throughput (T) (pkt/s)</b>	1100	1580	+43.64%

#### **4.1.1.4 Preprocessing of Simulated Dataset**

The first step is the removal of unused columns, elimination of incomplete or incorrect records (i.e., those with missing values) and discarding repeated features and columns as these can reduce the efficiency of the ML model. The second step involves converting categorical and string values into numerical values. This includes encoding non-numerical string values into integers and making them compatible with ML methods. In simple terms, this process cleans the simulated data. The raw dataset contains both categorical and numerical data, which can sometimes be inconsistent or skewed, leading to inaccurate results. Machine learning algorithms only work with cleaned numerical data. This step removes any missing values from the dataset and converts strings into numerical form so that the computer can easily process them.

Normalization is then applied to scale all feature values within a specific range, typically between  $[-1, 1]$ . We use the popular Min-Max Normalization which is represented by the following equation:

The formula of Min-Max Normalization is referred to Eq. (9).

$$A' = \frac{(A - \min \text{value of } A)}{(\max \text{value of } A - \min \text{value of } A)} * (D - C) + C \quad (9)$$

where  $A'$  is the having the Min-Max Normalized Data  $A$  if predefined boundary belongs to  $[C, D]$ .

#### **4.1.2 Preprocessing on CSE CIC IDS 2018 Dataset**

Several critical characteristics have been identified as essential to establish and evaluate an effective Intrusion Detection System (IDS) dataset framework [I. Sharafaldin, 2018]. These attributes encompass a variety of attack types, protocol information, extensive network traffic capture, full network interactions, detailed network configurations, a strong set of features, labelled data samples, diversity and pertinent metadata. [I. Sharafaldin, 2018]. The CSE-CIC-IDS-2018 dataset was developed with these characteristics in mind, employing a structured approach that utilizes profiles for systematic dataset generation. The dataset offers a comprehensive insight into various types of attacks and provides detailed information regarding different application models, network devices, and protocols. CICFlowMeter was utilized to capture network traffic, labelling data flows and documenting specifics such as source and destination addresses, port numbers, timestamps, and types of attacks. The testing environment simulated network traffic from protocols including HTTP, HTTPS, SSH, as well as email protocols like SMTP and POP3. To enhance the efficiency of the machine learning (ML) model, any incomplete or incorrect records (missing values) are eliminated, along with duplicate features and columns. Categorical and string values are converted into numerical formats to facilitate analysis. After encoding the network traffic datasets, normalization becomes necessary. Without normalization, one feature might overshadow others, regardless of the advantageous characteristics of the dataset. Both min-max normalization and Z-score normalization can be utilized for data preprocessing, with the choice depending on the specific dataset and algorithm employed. In the case of the CSE-CIC-IDS-2018 datasets, min-max normalization is preferred due to the significantly varying feature ranges. This

technique scales numeric columns from a range of 10,000 to 100,000 down to a range of 0 to 1, preserving the relative differences in values and ensuring that no information is lost.

The Min-Max scaling formula is applied, as normalized data generally enhances ML training efficiency. While min-max normalization may eliminate some outliers, this does not adversely affect system performance, as the detection task primarily focuses on long-term attacks. The 'Label' column, which contains the names of attacks identified in the dataset, is converted into numeric values. To balance the dataset, sampling is performed, resulting in 20 samples from which the best one is chosen for experimentation.

To evaluate the stability, generalization capability, and performance consistency of the classification model across different subsets of the input data, a repeated sampling-based experimentation framework was employed. This method operates by drawing multiple randomized subsets from the original training dataset, each subset containing a fixed percentage of the total available data.

The sampling is performed independently across multiple iterations to ensure statistical independence and to simulate varying data conditions that a real-world model might encounter. In each iteration, a specified proportion of data records is selected through random index generation, ensuring uniform distribution across the dataset space. The selected records are used to construct a new training matrix and a corresponding target label vector. These subsets, along with their associated labels, are preserved across iterations for traceability, reproducibility, and further pattern analysis. A supervised learning model, configured to operate with a fixed architecture, is then trained independently on each of these subsets.

The model's training process follows a standardized regimen, including internal validation and testing, which allows it to fine-tune its parameters based on both the



immediate subset and a small portion reserved for evaluation. After training, the classifier is assessed on its ability to correctly map the input features to the expected labels. The predictions generated by the model are processed to determine class membership through a probabilistic interpretation, typically based on the most activated output node. These predicted outcomes are then statistically compared with the ground truth labels using a confusion matrix to identify correct classifications and misclassifications across all categories.

To assess the quality of predictions beyond binary accuracy, a matching score is computed, reflecting the average confidence with which the model assigns its predictions during each sampling run. This score provides a more nuanced metric for evaluating the model's certainty and class separability. Over multiple iterations, this experimental design captures the variability in model performance caused by changes in input distributions, offering a practical understanding of how robust the model is to partial data exposure. By analyzing performance metrics across these iterations, researchers can gain insights into the resilience and adaptability of the classifier, verify its capacity to avoid overfitting to a specific subset, and estimate its expected performance under real-world conditions where training data may be incomplete, imbalanced, or non-uniformly distributed.

#### **4.1.3. Limitation of CSE CIC IDS 2018 Dataset**

There are various categories of malicious activities, including BOT attacks, Denial of Service (DoS), brute-force attempts, infiltration, and SQL injection. The Heartbleed attack is not represented in Figure 1 due to its limited sample size. Typically, datasets show an exponential distribution in the number of samples per class, where benign instances significantly outnumber malicious ones. For machine learning applications, a dataset is most effective when its class distribution is balanced or approximates a normal distribution.

## 4.2 Feature Selection Technique

To enhance the efficiency of classifiers used for attack detection, there is a pressing need for advanced techniques. One effective method is through Feature Selection (FS), which reduces the dimensionality of large datasets. Not all features are essential for detecting attacks; therefore, identifying the most relevant ones can significantly improve detection efficiency. FS addresses the challenge of high-dimensional datasets by pinpointing the smallest subset of optimal features, making it a crucial step in machine learning (ML) applications for attack detection. Building an effective Intrusion Detection System (IDS) necessitates the identification of relevant features that facilitate the detection of attacks.

However, this task is complex due to the potential relevance, redundancy, or excessiveness of features, which can increase the computational complexity of attack detection. Several methods can be applied to FS, including Greedy Search techniques such as Sequential Backward Selection (SBS) and Sequential Forward Selection (SFS). However, these methods often encounter challenges, including high computational costs and the risk of stagnation in local optima. To overcome these limitations, efficient search techniques capable of performing global searches are required. Evolutionary algorithms are particularly well-suited for this purpose due to their ability to explore broader search spaces. These algorithms have been successfully applied to various real-world problems, including wireless sensor networks. Their stochastic, population-based nature makes them ideal for feature selection. Two commonly used evolutionary algorithms in FS are Particle Swarm Optimization (PSO) and Genetic Algorithm (GA).

Feature selection is a complex issue that often necessitates the application of artificial intelligence methods for effective resolution. Researchers have long sought optimal approaches to enhance accuracy and efficiency in this domain. A

current focus in research is on improving the feature selection process by integrating additional algorithms with learning models. The following are various types of feature selection techniques:

**1.Filter Method:** This approach assesses the inherent characteristics of the data without relying on any specific classification algorithm. It uses predefined criteria to identify and select the most relevant features for the problem at hand. Features that meet or exceed the set threshold are retained, while those that do not are discarded. Common metrics for setting these thresholds include distance, information gain, correlation, and consistency. Examples of filter techniques include Correlation-based Feature Selection (FCBF) and Minimum Redundancy Maximum Relevance (MRMR). While this method is known for its speed and scalability, its independence from the classifier may lead to a compromise in accuracy.

**2.Wrapper Method:** Unlike the filter method, the wrapper method incorporates the classification algorithm into the feature evaluation process. This method establishes a connection between the selection of feature subsets and the classifier, typically resulting in more accurate outcomes compared to the filter approach. However, this increased interaction can lengthen the processing time and elevate the risk of overfitting, where the classifier becomes too tailored to the training data and struggles to generalize to new instances

**3.Embedded Method:** Similar to the wrapper approach, the embedded method utilizes the same classifier for feature selection during the evaluation stage. However, it is generally more efficient in terms of computational resources and requires less processing time compared to the wrapper method.

**4.Hybrid Method:** This technique combines elements of both the filter and wrapper methods in a two-phase sequential process. Initially, the filter method is employed to create a preliminary subset of features, which is then further refined using the wrapper method.

#### **4.2.1 Techniques for Feature Selection**

**1.Information Gain (IG):** This approach ranks attribute subsets by calculating the Information Gain (IG) entropy for each attribute in descending order. Each attribute is assigned a score ranging from 1 (most relevant) to 0 (least relevant). Attributes with the highest scores are selected for the next dimensionality reduction step.

**2.Principal Component Analysis (PCA):** Although the attributes chosen using the Information Gain (IG) method can be directly applied for classification purposes, IG often favors attributes with a wide range of potential values, potentially leading to inflated gain values. To address this issue, selected features undergo further reduction using PCA, which identifies an optimal subset of attributes. PCA transforms a set of features into linearly uncorrelated variables through orthogonal transformations, retaining most of the original information. The transformed variables, known as principal components, are ranked by decreasing variance, with the first component capturing the maximum variance.

**3.Metaheuristic Techniques:** These techniques can be employed as optimizers alongside learning models. Recent research has demonstrated their ability to deliver highly accurate results and enhance the feature selection process. Various metaheuristic algorithms, such as Simulated Annealing and Ant Colony Optimization, have been applied to the feature selection problem. Specific Algorithms for Feature Selection are:

- **Particle Swarm Optimization (PSO):** PSO is a metaheuristic optimization algorithm inspired by the social behavior of birds or fish. It maintains a population of candidate solutions, referred to as particles, within a multi-dimensional search space. Each particle represents a potential solution to the optimization problem, and its position corresponds to a specific set of feature values. PSO operates on the principle of collaboration among particles, with each adjusting its position based on its own historical best-known position (pbest) and the best-known position within the entire swarm (gbest).
- **Genetic Algorithm (GA):** The Genetic Algorithm (GA) is a heuristic optimization method that draws inspiration from the principles of natural selection and genetics. It evolves a population of potential solutions across multiple generations. Each candidate solution, or chromosome, is represented as a string of symbols, usually in binary form. The GA operates using three primary processes: selection, crossover, and mutation. During selection, individuals are chosen as parents based on their fitness levels. This is followed by the crossover operation, which produces offspring, and mutation, which introduces random alterations to preserve genetic diversity. This iterative process continues until a stopping criterion is met, such as reaching a predetermined number of iterations or obtaining a satisfactory level of solution quality.

#### 4.2.2 Proposed Feature Selection Algorithm

- **Basic Firefly Algorithm**

Introduced by Xin-She Yang in 2008, Firefly Algorithm (FA) is based on the attractiveness of fireflies, which is determined by their brightness or flash intensity. Fireflies use their flashes to attract mates and communicate with each other, and their behavior has inspired the development of this optimization technique.

In the context of optimization problems, including feature selection, FA simulates the movement of fireflies in search of optimal solutions. The attractiveness of a firefly is determined by its fitness value, with brighter fireflies representing better solutions. Fireflies move towards brighter ones while gradually exploring the search space to find optimal or near-optimal solutions.

The Firefly Algorithm (FA) is effective because it does not rely on prior values, unlike Particle Swarm Optimization (PSO), which helps prevent premature convergence that can occur with PSO. Additionally, FA does not incorporate the concept of velocity found in PSO. It can manage its modality and easily adjust to the problem landscape by varying its scaling parameter, such as  $\gamma$ . Moreover, FA serves as a generalization of several optimization methods, including PSO, Simulated Annealing (SA), and Differential Evolution (DE). The relevance of FA in cybersecurity lies in its ability to efficiently sift through the vast feature space, discerning between relevant and irrelevant features to enhance the effectiveness of intrusion detection systems, malware detection, and other security mechanisms. By focusing on selecting the most impactful features for classification or detection tasks, the Firefly Algorithm (FA) minimizes computational demands while enhancing the accuracy and reliability of cybersecurity systems.

#### • **Hybridized Firefly Algorithm with Decision Tree Algorithm**

Feature selection (FS) plays a crucial role in classification. A modified Firefly Algorithm (FA) for FS is proposed. In this modified approach, a decision tree (DT) classifier is employed as the fitness function for the FA. FA was chosen due to its rare application in FS by researchers. A literature review has demonstrated that FA outperforms Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) [H. S. Gebremedhin et al., 2020]. Additionally, some PSO variants are considered

special cases of FA. Adjusting the parameters of FA can further enhance its convergence performance

**1. Initialization:** Fireflies are randomly distributed within the search space, with their positions corresponding to different feature subsets.

**2.Objective Function:** In the context of feature selection, the objective function measures the performance of a classifier (e.g., accuracy) using the selected features.

**3.Attractiveness:** The attractiveness of a firefly is determined by its brightness, which is calculated based on its fitness value. Brighter fireflies have higher fitness values, indicating better solutions. The attractiveness decreases with distance between fireflies and increases with their brightness.

**4.Movement:** Fireflies move towards brighter ones in the search space, guided by their attractiveness and randomization. The movement of fireflies is characterized by a random walk process, where they adjust their positions iteratively to approach better solutions.

**5.Update Positions:** The fireflies update their positions based on the attraction index generated.

**6.Termination:** The algorithm iterates until a termination criterion is met, such as a maximum number of iterations or achieving a satisfactory solution quality.

**7.Return Solution:** Finally, the algorithm returns the best solution found by the fireflies, which corresponds to the optimal or near-optimal feature subset identified by the Firefly

**Table 4.8 Pseudocode for Proposed Feature Selection Algorithm**

---

**Algorithm 4.2 : Hybridized Firefly Algorithm with Decision Tree Algorithm**

---

Initialize the parameters  $n$ ,  $\alpha$ ,  $\beta$ ,  $\gamma$ ,  $\maxIterations$

Number of fireflies =  $n$

Initialize fireflies randomly

for  $i=1$  to  $\maxIterations$  do

    Evaluate fitness for each firefly by using decision tree

    fitness = zeros ( $n$ , 2)

    for  $i= 1$  to  $n$

        SelectedFeatures = find (fireflies(:, $i$ ) > 0.5

        [accuracy, loss] = calculate\_accuracy(selectedFeatures, alldata, data\_label)

        fitness ( $i$ , :) = [accuracy, loss]

    end for

Sort fireflies based on fitness

[:, sortedIndices] = sortrows (fitness, [-1,2])

fireflies = fireflies(:, sortedIndices)

for  $i=1$  to  $n$

    for  $j= 1$  to  $n$

        attractiveness =  $\beta_0 * \exp(-\alpha * ||\text{fireflies}(:, i) - \text{fireflies}(:, j)||^2)$

        fireflies(:,  $i$ ) = fireflies(:,  $i$ ) +  $\gamma * (\text{fireflies}(:, j) - \text{fireflies}(:, i)) * \text{attractiveness}$

        Perform a random walk if necessary to prevent out-of-bounds

        fireflies(:,  $i$ ) = min (max(fireflies(:,  $i$ ), 0, 1))

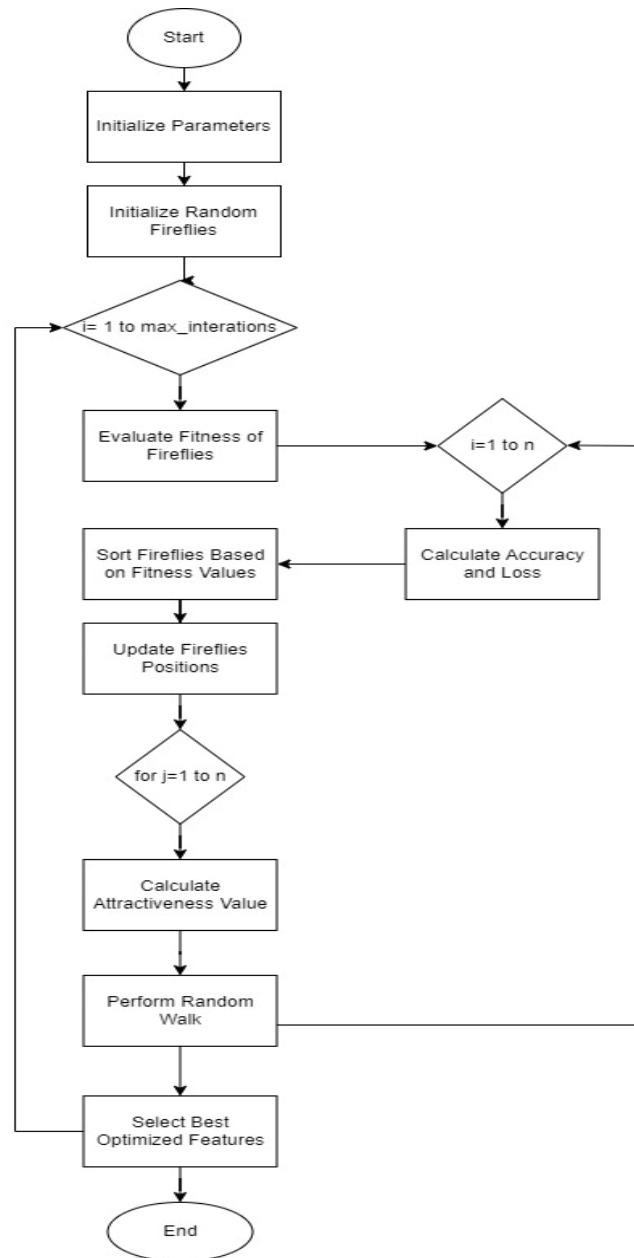
        attractionvalue( $i, j$ ) = attractiveness

    end for

end for

---





**Figure No. 4.5 Flowchart of Proposed Feature Selection Algorithm**

1. **Initialization:** The algorithm initializes parameters such as the number of fireflies, maximum fitness threshold, maximum iterations, and algorithmic parameters. It also initializes the attraction value matrix.
2. **Random Initialization of Fireflies:** Fireflies are randomly initialized, where each firefly represents a potential solution (i.e., a feature subset). Fireflies' positions correspond to binary selections of features, with each feature being either selected or not selected.
3. **Main Loop (Iterations):** The algorithm iterates for a maximum number of iterations. Within each iteration, the following steps are performed:
  - a) **Fitness Evaluation:** The fitness of each firefly is evaluated by calculating its accuracy and loss using a decision tree classifier. The accuracy represents how well the selected features perform in classification, and the loss is used to guide the search towards better solutions.
  - b) **Sorting Fireflies:** Fireflies are sorted based on their fitness values, with higher fitness values indicating better solutions.
  - c) **Updating Fireflies' Positions:** Each firefly updates its position based on the attractiveness of other fireflies. Attractiveness is determined by the brightness (fitness) of other fireflies and their proximity. The firefly moves towards brighter fireflies while considering the absorption coefficient, step size, and randomization.
  - d) **Selection of Best Fireflies:** After updating positions, the best fireflies' positions are selected based on their fitness values.

**e) Post-processing:** The algorithm post-processes the selected features to ensure that the maximum number of selected features does not exceed the specified threshold. If the number of selected features exceeds the threshold, a random subset of features is retained to meet the threshold.

**f) Evaluation with Decision Tree:** To assess the accuracy of the chosen features, a decision tree classifier is trained and then tested against the dataset. The classifier's accuracy acts as a performance indicator for the selected features.

**g) Return Selected Features:** The algorithm returns the selected features, along with their corresponding accuracy evaluated by the decision tree classifier.

The entire implementation provided is written in MATLAB, a widely used programming language for numerical computing and algorithm development.

• **Features Selected:**

Number of simulations on the CSE CIC IDS 2018 dataset are performed by using Proposed Feature Selection Algorithm which is the hybridized firefly algorithm and it is observed that 43 features are selected as the optimized feature subset.

Number of simulations on the CSE CIC IDS 2018 dataset are performed by using PSO and it is observed that 51 features are selected as the optimized feature subset.

Number of simulations on the CSE CIC IDS 2018 dataset are performed by using GA and it is observed that 63 features are selected as the optimized feature subset.

The 43 features optimized by proposed algorithm gives better results than the techniques such as Particle Swarm Optimization (PSO) and Genetic Algorithms (GA).

**4.3 Summary:** This study presents a novel approach to developing an intrusion detection system by combining a hybrid firefly algorithm with a hybrid classifier. The proposed architecture's effectiveness was assessed using both a simulated dataset and the recent CSE CIC IDS 2018 dataset. A new feature selection

algorithm, which integrates the firefly algorithm with a decision tree, is introduced. The proposed feature selection method outperforms both the PSO algorithm and approaches without feature selection. The study emphasizes the significance of feature selection before classification and shows that the proposed algorithm outperforms other leading methods in the field.

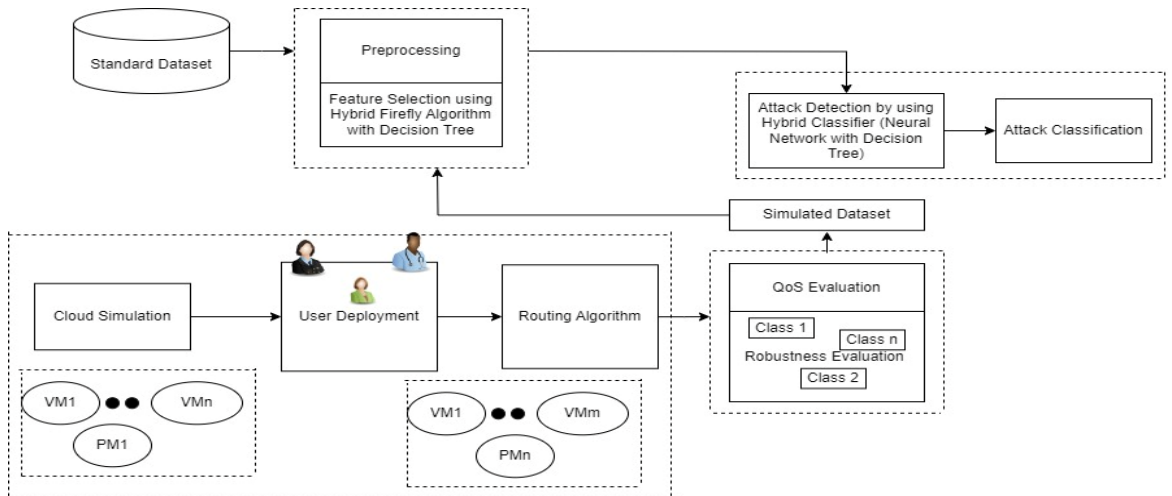
## CHAPTER 5 PROPOSED ARCHITECTURE FOR DETECTION OF ATTACKS IN CLOUD COMPUTING ENVIRONMENT

---

The architecture is designed to detect attacks in the cloud computing environment by employing Swarm Intelligence techniques and Machine Learning techniques. Swarm Intelligence methods are well-regarded for their capability to optimize difficult problems by facilitating collaboration and knowledge sharing among agents, which makes them particularly effective for feature selection.

### 5.1 Proposed Architecture for Detection of Attacks in Cloud Computing Environment Using Optimized Classifier

Figure 5.1 is showing the proposed architecture for the detection of the attacks in cloud computing environment. The description of the architecture is described as follow:



**Figure 5.1 Proposed Architecture for Detection of Attacks in Cloud Computing Environment**

The architecture incorporates a hybrid classifier to improve the effectiveness of attack detection. This algorithm combines two classifiers to use their individual strengths, thereby improving threat identification. This combining use of Swarm Intelligence for feature selection and a hybrid classification algorithm for detection promises not only to refine the analysis but also to significantly improve the performance of identifying attacks within the network.

**1.Standard Dataset:** Standard benchmark datasets for research purposes can be accessed on platforms like Kaggle and the UNB website. The CSE CIC IDS 2018 dataset can be downloaded from <https://www.unb.ca/cic/datasets/ids-2018.html>. Profiles are created within this dataset, which was developed through the simulation of various attack scenarios. The CSE CIC IDS 2018 dataset includes seven categories for evaluation: Benign, Bot Attack, Bruteforce Attack, DoS Attack, DDoS Attack, Infiltration Attack and Web Attack. This dataset was produced as a collaborative project by the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC). The attack simulations were conducted using 50 machines, while the victim system comprised five departments with 420 machines and 30 servers. The dataset captures network traffic alongside log files for each machine, resulting in 80 features extracted from the network traffic using CICFlowMeter-V3.

**2. Cloud Simulation:** Simulated dataset created for the evaluation of the architecture is done by using the well-known cloud simulating environment, i.e., CloudSim. Deployment of real-world cloud is not easy for research as it is very costly and requires a lot of infrastructure. By using the simulating environments, researchers can perform their test multiple times. The environment in simulation is also under the control of the researchers. In the proposed approach, there are three features generated, i.e., Throughput, PDR, Consumed Power. Three categories are generated, i.e., Highly Robust, Moderately Robust and Least Robust. Approximately one lakh records are generated in the dataset. In least robust category, two attacks are generated-DDoS Attack and User-to-Root Attack.

For the cloud simulation, users are deployed in the environment. In this simulation model, nodes represent workloads rather than physical machines (PMs). Each node simulates a unique workload processed within a cloud infrastructure composed of PMs and virtual machines (VMs) that manage these tasks. The workloads (nodes) are randomly generated with specific characteristics, such as data volume, processing time, and resource requirements. The purpose of the simulation is to model how these workloads are distributed and processed efficiently across the cloud infrastructure by the virtualized resources, specifically VMs hosted on PMs.

**3. Routing Algorithm:** Routing algorithm is used which is using the that which offers the least cost. The algorithm starts by generating random locations for each workload (node), representing different geographical or logical sources. Each workload is assigned specific attributes. Physical machines (PMs) in the datacenter provide the infrastructure, while virtual machines (VMs) hosted on these PMs handle the actual processing of workloads. As workloads enter the system, the algorithm dynamically routes them through the cloud infrastructure, deciding whether a workload can be processed directly by a VM on a PM or needs to be forwarded through multiple VMs due to resource constraints or network conditions.

For each workload, the algorithm calculates the most efficient path to its destination, such as a service endpoint or another processing unit. This route represents the movement of the workload through the cloud, potentially involving multiple VMs on different PMs. If the workload is within a certain distance threshold from the destination, the handling VM processes it directly. Otherwise, intermediate VMs on other PMs may be used to forward the workload, simulating a multi-hop scenario where it moves between different parts of the cloud infrastructure.

As workloads progress through the cloud, VMs process the incoming data, perform computational tasks, and forward the workload to the next VM or its final destination. The throughput of each workload is monitored to measure how efficiently data is processed and delivered within the system. The algorithm ensures that VMs handle workloads effectively, based on available resources like CPU, memory, and bandwidth. Another key metric, the Packet Delivery Ratio (PDR), is used to assess network efficiency by indicating the percentage of the workload successfully processed and delivered to its destination without loss.

**4.QoS Evaluation and Robustness Evaluation:** Clusters with a low standard deviation indicate a more robust system, where performance metrics are tightly grouped, reflecting consistent processing of workloads with minimal variation in Throughput, Power Consumption and Packet Delivery Ratio (PDR). A low standard deviation demonstrates the cloud infrastructure's ability to deliver stable and secure processing, efficiently managing resources and ensuring workloads are handled effectively. These clusters are identified as highly robust, emphasizing the system's capability to maintain secure and efficient operations, even under fluctuating workload conditions. By clustering aggregated data and categorizing it based on standard deviation, this approach provides a detailed analysis of the cloud



infrastructure's performance in terms of security and robustness. It helps cloud administrators identify areas of lower security or efficiency, allowing them to optimize resource allocation, reduce security risks, and improve the overall stability and reliability of the cloud environment. This method enhances the system's ability to handle current workloads while preparing it for future scaling, ensuring long-term robustness, efficiency, and security.

After categorizing the data into three clusters using k-means clustering, varying levels of robustness based on standard deviation are represented and the next step is to classify the data further using a hybrid classifier. This hybrid classifier integrates machine learning methods, including decision trees and neural networks to enhance both accuracy and reliability. It ensures that workloads are classified into one of three categories: highly robust, moderately robust or least robust, based on the system's capacity to process them efficiently and securely.

The hybrid classifier analyzes performance metrics from each category, such as throughput, power consumption, and PDR, to determine the system's robustness for each workload. By using a combination of machine learning methods, the classifier can detect subtle variations in performance metrics, enabling more accurate classification. Its goal is to achieve high classification accuracy, reliably identifying workloads as least, moderately, or highly robust based on the system's performance.

Once the classifier processes the data, a confusion matrix is generated to evaluate its performance across the three robustness categories. The confusion matrix breaks down predictions versus actual categories, showing how well the classifier performed in categorizing workloads. The matrix consists of four key components for each category:

1. True Positives (TP) Workloads correctly classified into their actual category (e.g., highly robust workloads correctly identified as highly robust).
2. True Negatives (TN): Workloads correctly excluded from a category (e.g., moderately robust workloads not misclassified as highly robust).
3. False Positives (FP) Workloads incorrectly classified into a category (e.g., least robust workloads mistakenly classified as highly robust).
4. False Negatives (FN): Workloads that should have been classified into a category but were not (e.g., highly robust workloads misclassified as moderately robust).

The hybrid classifier aims to maximize True Positives and True Negatives while reducing False Positives and False Negatives. This ensures precise differentiation among the three robustness levels while keeping the classification error rate low. Achieving high classification accuracy is crucial, as it represents the ratio of accurately classified workloads to the total number of workloads.

The confusion matrix also helps identify areas where the hybrid classifier may need improvement. For example, a high number of False Positives or False Negatives in the least robust category could indicate that the classifier is incorrectly interpreting performance variability. This insight allows for fine-tuning of the classifier, adjusting parameters and thresholds to better detect security vulnerabilities or inefficiencies in workload handling.

By leveraging the hybrid classifier, the system enhances its ability to classify workloads into the appropriate robustness category, improving the overall security and efficiency of the cloud infrastructure. The integration of k-means clustering, hybrid classification, and confusion matrix evaluation ensures the system can handle a diverse range of workloads while maintaining high standards in performance, security, and resource management. This process enables the cloud infrastructure to adapt to changing conditions, dynamically identifying and

addressing potential weaknesses in real-time, ultimately leading to a more resilient and robust cloud computing environment.

**5. Preprocessing Module:** In the preprocessing module, the raw dataset is turned into fine form which makes the classification task easy and generates the results with better accuracy. Strings are converted into the numeral form. Normalization is performed for making the values in the dataset to lie in the range of  $[-1,1]$ .

Data sampling is performed for balancing the dataset. CSE CIC IDS 2018 dataset is unbalanced as some classes are in large number as compared to other classes. Machine learning techniques do not give biased results with these types of datasets. Data sampling makes the dataset balanced. The pre-processed dataset is given as input to the next module, i.e., Feature Selection Module.

**6. Feature Selection Module:** Feature selection (FS) plays a crucial role in classification. A modified Firefly Algorithm (FA) for feature selection is proposed. The innovation of the proposed feature selection algorithm is its incorporation of a decision tree (DT) classifier as the fitness function for firefly algorithm (FA). The choice of FA is notable because it has not been widely utilized for feature selection in previous research. Evidence suggests that FA delivers better performance compared to Particle Swarm Optimization (PSO) and Genetic Algorithm (GA). Additionally, some variants of PSO are considered special cases of FA. By adjusting FA parameters, its convergence can be further improved. Since FA is a swarm-based algorithm, it shares the same advantages as other swarm-based techniques. Some variants of Particle Swarm Optimization (PSO) are actually forms of FA, functioning as accelerated PSO. FA's adaptability allows it to control randomness and adjust as iterations progress. These benefits make FA well-suited for handling clustering, classification, and continuous optimization problems.

**7. Classification Module:** In many cases, improving the overall classification accuracy can be achieved by designing hybrid classification models that leverage the strengths of multiple algorithms. One common approach is to combine the capabilities of different classifiers to create a more robust and accurate model. In the context of the proposed work, the hybrid model consists of a combination of Neural Network and decision tree classifiers. It is noted that various machine learning algorithms have been used to address these issues, including decision tree algorithms and support vector machine models, k-means, k-nearest neighbor, artificial intelligence approaches and several other.

- **Neural Network (NN):** Neural Networks consist of interconnected layers of artificial neurons that process input data and learn to extract patterns and features from the data through training. Neural networks are capable of learning complex relationships and non-linear mappings between input and output variables, making them well-suited for tasks such as pattern recognition, classification, and regression.
- **Decision Tree:** Decision trees are simple yet effective classification models that partition the feature space into a set of hierarchical decision rules based on the values of input features. Each internal node of the tree represents a decision based on a feature, and each leaf node corresponds to a class label or prediction. Decision trees are interpretable, easy to understand, and capable of handling both numerical and categorical data. They are particularly useful for capturing interactions and non-linear relationships in the data.

**Table 5.1 Pseudocode for Classification Algorithm**

---

**Algorithm: Hybridized Neural Network with Decision Tree Classification**

---

**Algorithm**

---

Initialization: Initialize Data X, Labels Y, Decision Tree Classifier DT, Neural Network Classifier NN, Number of Predictions =P, Number of Iterations =N

$P_{DT} = DT.predict(X)$

$P_{NN} = NN.predict(X)$

$P=zeros(N,1)$

for i=1 to N

    if  $P_{DT}[i] = P_{NN}[i]$  then

$P[i] = P_{DT}[i]$

    else

$P[i] = Unsure$

    end if

end for

Return Predictions P

---

1. **Input:** The algorithm takes as input the dataset X containing the instances to be classified and their corresponding labels Y. Additionally, it requires two classifiers: a decision tree classifier (DT) and a neural network classifier (NN).

2. **Prediction Phase:**

- The algorithm first makes predictions on the dataset X using both the decision tree classifier (DT) and the neural network classifier (NN).
- It stores the predictions made by the decision tree classifier in  $P_{DT}$  and the predictions made by the neural network classifier in  $P_{NN}$ .

### 3. **Comparison and Final Prediction:**

- The algorithm then iterates through each instance in the dataset.
- For each instance, it checks if the predictions made by both classifiers are the same.
- If both classifiers agree on the prediction for the instance, the algorithm assigns the agreed prediction to the final predictions (P).
- If the classifiers disagree on the prediction for the instance, the algorithm labels the prediction as "Unsure".

4.**Output:** The algorithm returns the final predictions P, where a prediction is considered true only if both the decision tree and neural network classifiers agree on it. If they disagree, the prediction is labelled as "Unsure".

This algorithm ensures that a classification is considered true only when both the decision tree and neural network classifiers produce the same prediction for an instance, thereby adding an additional level of confidence to the classification results. If the classifiers disagree, the algorithm flags the prediction as "Unsure", indicating uncertainty in the classification decision. By combining the strengths of Neural Network and decision tree classifiers into a hybrid model, several benefits can be realized:

a) **Complementary Capabilities:** Neural networks are particularly effective at identifying intricate patterns and relationships within data, whereas decision trees are skilled at managing categorical variables and generating easily interpretable rules. By combining these models, the hybrid model can leverage their complementary capabilities to improve overall classification accuracy.

**b) Ensemble Effect:** Ensemble methods, such as combining different classifiers, often lead to better generalization performance than individual models. The hybrid model can harness the ensemble effect by integrating predictions from both Neural Network and decision tree classifiers, resulting in a more robust and accurate classification model.

**c)Model Interpretability:** Decision trees offer clear and understandable models, making them accessible for domain experts to interpret. By incorporating decision tree components into the hybrid model, it enhances interpretability while maintaining the predictive power of Neural Network.

In the proposed work, the hybrid classification model comprising Neural Network and decision tree components offers a promising approach to enhance the overall classification accuracy. By effectively combining the strengths of both models, it can overcome the limitations of individual algorithms and provide more accurate and interpretable predictions for the given classification task.

**1.Improved Robustness to Overfitting:** Neural networks, especially deep architectures, have a tendency to overfit the training data, particularly when dealing with small datasets or noisy input. Decision trees, on the other hand, are less prone to overfitting due to their inherent simplicity and ability to capture local patterns. By integrating decision trees into the hybrid model, it can help mitigate the risk of overfitting, leading to more generalized and reliable predictions.

**2.Enhanced Feature Representation Learning:** Neural networks excel at automatically learning feature representations from raw data, extracting hierarchical and abstract features through successive layers of neurons. By pre-processing the data using Neural Network layers before feeding it into the decision tree classifier, the hybrid model can provide better feature representations,

potentially leading to improved discrimination between classes and higher classification accuracy.

**3.Flexible Model Architecture:** The hybrid model offers flexibility in designing the architecture and configuration of both Neural Network and decision tree components. Researchers and practitioners have the freedom to experiment with different network architectures, activation functions, learning rates, and tree parameters to optimize the model's performance for the specific classification task at hand. This flexibility enables fine-tuning and customization to adapt the model to varying datasets and application requirements.

**4.Adaptive Learning and Adaptability:** Neural networks are inherently adaptive and can continuously update their internal parameters through backpropagation and gradient descent, allowing them to adapt to changes in the data distribution over time. Decision trees, although static once trained, can be easily updated or retrained with new data to accommodate concept drift or changes in the underlying data characteristics. The hybrid model can leverage this adaptability to maintain high performance in dynamic and evolving environments, making it suitable for real-world applications where the data distribution may change over time.

**5.Enriched Model Interpretability and Explain ability:** While Neural Network models are known for their black-box nature and lack of interpretability, decision trees offer transparent and interpretable models that can provide insights into the decision-making process. By combining both models in the hybrid architecture, it not only improves classification accuracy but also enhances model interpretability and explain ability, allowing users to understand the rationale behind predictions and gain actionable insights from the model's output.



## 5.2 Computation Complexity Analysis

### Neural Network Complexity (Pattern Recognition Model)

The implemented architecture is a multi-layer feedforward neural network consisting of three hidden layers with 128, 64 and 32 neurons.

If  $D$ =Number of input features,  $C$ = Number of output classes,  $N$ = Number of training samples then,

$$P = (D * 128 + 128) + (128 * 64 + 64) + (64 * 32 + 32) + (32 * C + C) \quad (5)$$

where the total number of trainable parameters  $P$ (including weights and biases).

The overall computational complexity for training the network over  $E$  epochs becomes  $O(E*N*P)$ .

This accounts for forward propagation, backpropagation and parameter updates using gradient descent. In the implementation,  $E=200$  and the training dataset is split using the stratified holdout sampling (70% training and 30% testing).

**5.3 Summary:** This chapter presents a framework designed for detecting attacks within cloud computing environments. The architecture's effectiveness is assessed using two distinct datasets. Hybridized firefly algorithm with decision tree is used to extract relevant features from the datasets. A hybrid classification method that considers a classification true only when both the decision tree and neural network classifiers agree on the prediction is used for the detection of attacks.

## CHAPTER 6 RESULTS AND DISCUSSIONS

---

The chapter describes the results of the research work. Results are represented in tabular and graphical form. Detailed discussion on the results is described in this chapter. Two datasets are used for the evaluation of the proposed architecture for the detection of the attacks. Comparison of proposed feature selection with the popular PSO and GA is described in this chapter.

### 6.1 Implementation Details

The implementation details describe the hardware and software requirements used during the research. Table 6.1 describes the implementation details related to the research work.

**Table 6.1 Implementation Details**

<b>Operating System</b>	Windows 10
<b>Hard Disk</b>	8 GB
<b>Implementation Software</b>	MATLAB 2022b
<b>Simulation IDE</b>	Eclipse IDE 2024
<b>Datasets</b>	Simulated Dataset, CSE CIC IDS 2018 Dataset

### 6.2.1 Performance Metrics

The essential terms associated with performance metrics include True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). TP refers to the number of normal packets correctly identified, while TN indicates the number of attack packets accurately recognized. FP denotes the number of packets incorrectly classified as attack when they are actually normal packets and known as Type I error. FN refers to the packets that are misclassified as normal packets when they are attack packets and known as a Type II error.

Precision is measuring the accuracy of all positive predictions. Accuracy is measuring the overall correctness of the classification. Recall is measuring the ability to identify all actual positives. F-Measure is balancing the precision and recall.

Performance metrics are described in equation forms as Eq. (1)-(4).

$$\text{Precision} = \frac{\text{True Positive}}{(\text{True Positive} + \text{False Positive})} \quad (1)$$

$$\text{Accuracy} = \frac{(\text{True Positive} + \text{True Negative})}{(\text{True Positive} + \text{True Negative} + \text{False Positive} + \text{False Negative})} \quad (2)$$

$$\text{Recall} = \frac{\text{True Positive}}{(\text{True Positive} + \text{False Negative})} \quad (3)$$

$$\text{F - Measure} = \frac{(2 * \text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})} \quad (4)$$

### 6.2.2 Hypothesis Testing

Hypothesis testing was performed on the classification output to validate whether the model performance is statically significant compared to random chance.

1. Null Hypothesis ( $H_0$ ): The classifier performs at the level of random guessing.
2. Alternate Hypothesis ( $H_1$ ): The classifier performs significantly better than random guessing.

Given that the confusion matrix encapsulated observed vs. expected frequencies across multiple classes, a Chi-Square Test for Independence was applied. The Chi-Square is defined as

$$\chi^2 = \sum (O_i - E_i)^2 / E_i \quad (6)$$

where  $O_i$  is observed values and  $E_i$  is expected values assuming uniform distribution.

### 6.3 Feature Selection Algorithm Analysis

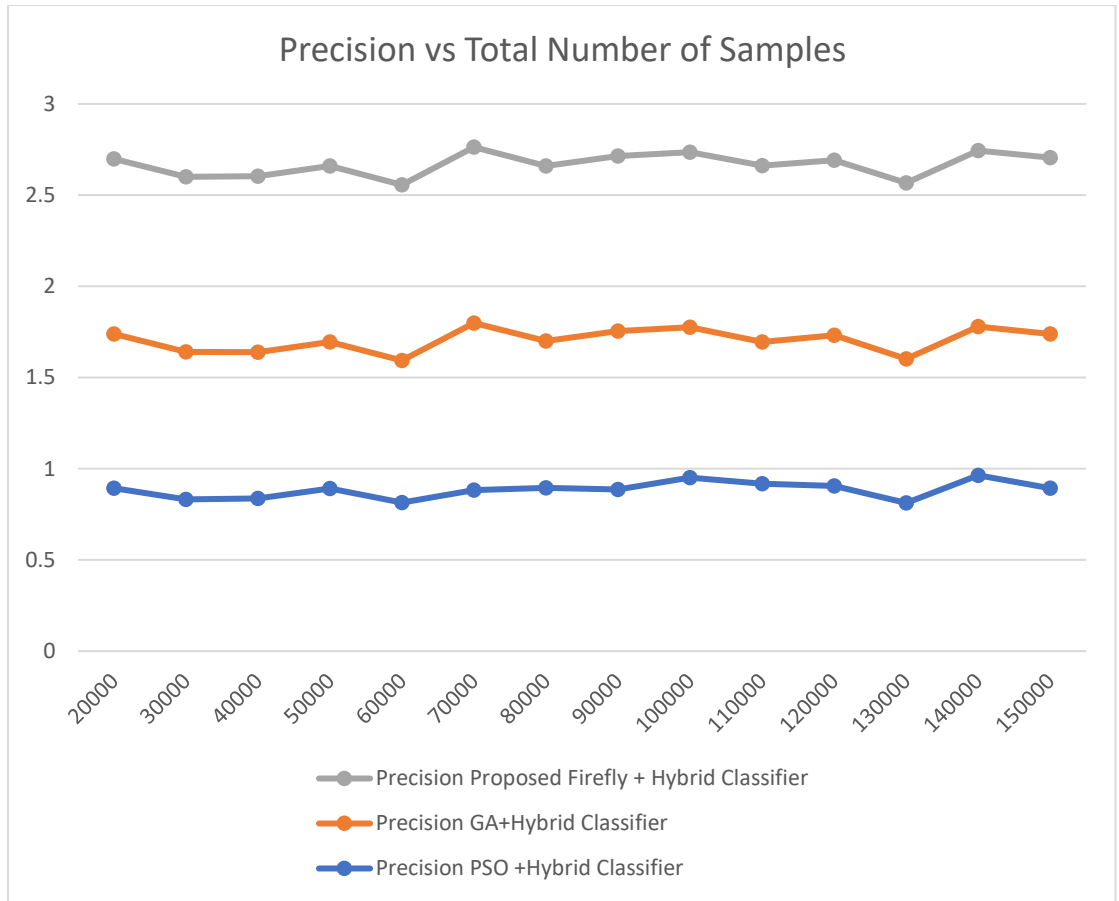
The proposed architecture presented in this study represents a significant step forward in the field of intrusion detection. Leveraging a substantial dataset comprising 1.5 lakh (1,50,000) records of CSE CIC IDS 2018 dataset, this research focuses on the identification and classification of the attacks.

This is achieved through the comprehensive evaluation of detection performance metrics, including accuracy, precision, recall and f-measure. These metrics collectively offer a holistic view of the system's efficacy, capturing both its ability to accurately identify attacks and its capacity to minimize the false alarms.

**Table 6.2 Precision Comparison of Feature Selection Algorithms**

<b>Total number of Samples</b>	<b>Precision PSO+Hybrid Classifier</b>	<b>Precision GA+Hybrid Classifier</b>	<b>Precision Proposed Firefly + Hybrid Classifier</b>
<b>20000</b>	0.89325121	0.84606881	0.95885899
<b>30000</b>	0.83120870	0.81011737	0.95917421

<b>40000</b>	0.83774387	0.80041096	0.96476389
<b>50000</b>	0.89183689	0.80315651	0.96498339
<b>60000</b>	0.81438771	0.77887040	0.96272396
<b>70000</b>	0.88233581	0.91639030	0.96415675
<b>80000</b>	0.89576420	0.80485535	0.95938611
<b>90000</b>	0.88532679	0.86966180	0.95862370
<b>100000</b>	0.95187920	0.82428211	0.95963389
<b>110000</b>	0.91838369	0.77661431	0.96626869
<b>120000</b>	0.90554341	0.82591551	0.95983295
<b>130000</b>	0.81215359	0.78950563	0.96603604
<b>140000</b>	0.96386093	0.81487536	0.96588206
<b>150000</b>	0.89279464	0.84612539	0.96579691



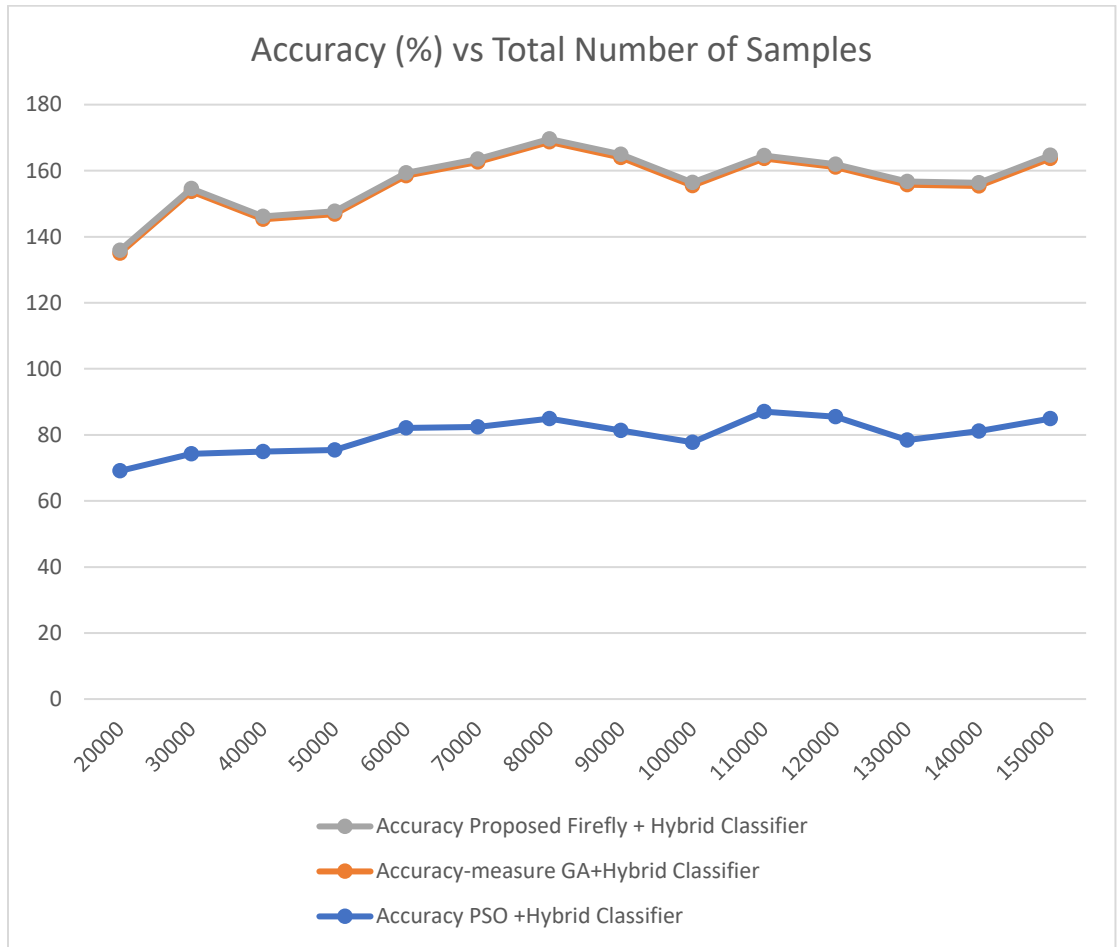
**Figure 6.1 Comparison of Precision Feature Selection Algorithms**

The precision of the architecture is represented as Precision Proposed Firefly + Hybrid Classifier is highest with different subsets of the dataset like 20,000 samples. The last sample subset is containing 1,50,000 records. The precision of Proposed Firefly + Hybrid Classifier outperforms precision of PSO and GA algorithms as it is 0.9589 for 20,000 samples. For 1,50,000 records precision of the proposed architecture is 0.965.

**Table 6.3 Accuracy Comparison of Feature Selection Algorithms**

<b>Total number of Samples</b>	<b>Accuracy PSO +Hybrid Classifier</b>	<b>Accuracy-measure GA+Hybrid Classifier</b>	<b>Accuracy Proposed Firefly Hybrid Classifier</b> +
<b>20000</b>	69.0787370	65.9316031	87.9843
<b>30000</b>	74.292018	79.4134382	88.88347
<b>40000</b>	74.9688163	70.2975348	89.9885
<b>50000</b>	75.4650278	71.3911008	86.0560
<b>60000</b>	82.1031559	76.395777	0.870500
<b>70000</b>	82.3963686	80.2633572	87.760
<b>80000</b>	84.9556829	83.7484475	88.2863
<b>90000</b>	81.2901379	82.7116438	90.7894
<b>100000</b>	77.7693584	77.7328603	91.19845
<b>110000</b>	87.0312721	76.6258699	92.8784
<b>120000</b>	85.4534590	75.5824956	93.7846
<b>130000</b>	78.4072976	77.3573391	94.2330

<b>140000</b>	81.1369337	74.2547736	94.9835
<b>150000</b>	84.9296383	78.8100726	94.9880



**Figure 6.2 Comparison of Accuracy for Feature Selection Algorithms**

The accuracy of the architecture is represented as Accuracy Proposed Firefly + Hybrid Classifier is highest with different subsets of the dataset like 20,000 samples. The last sample subset is containing 1,50,000 records. The accuracy of Proposed Firefly + Hybrid Classifier outperforms precision of PSO and GA

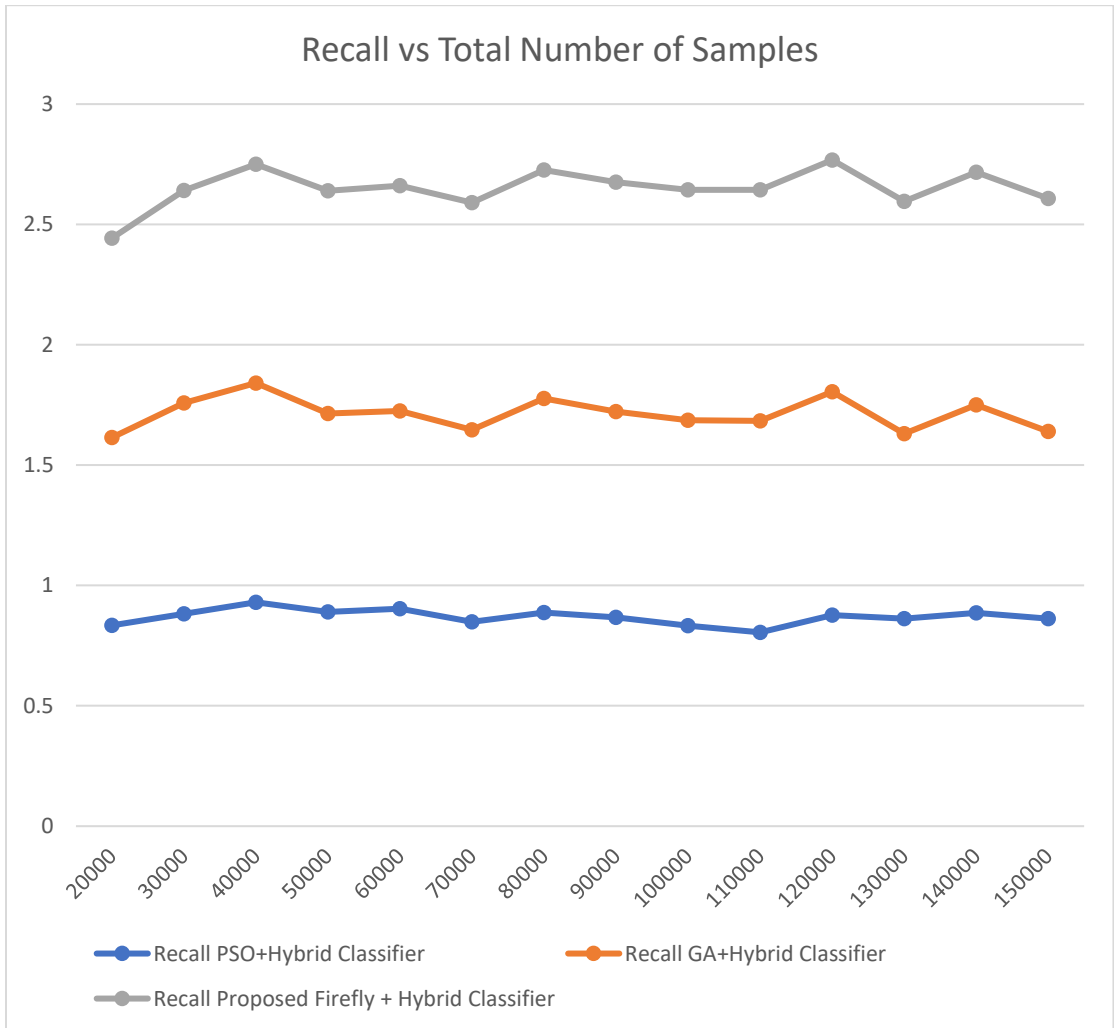


algorithms as it is 87.9843 for 20,000 samples. For 1,50,000 records accuracy of the proposed architecture is 94.9880.

**Table 6.4 Recall Comparison of Feature Selection Algorithms**

<b>Total number of Samples</b>	<b>Recall PSO+Hybrid Classifier</b>	<b>Recall GA+Hybrid Classifier</b>	<b>Recall Proposed Firefly + Hybrid Classifier</b>
<b>20000</b>	0.83396589	0.78073702	0.82851306
<b>30000</b>	0.88278632	0.87612737	0.88205147
<b>40000</b>	0.93018483	0.91105840	0.90895161
<b>50000</b>	0.88995665	0.82414106	0.92531343
<b>60000</b>	0.90343611	0.82126443	0.93597118
<b>70000</b>	0.84921977	0.79800564	0.94365591
<b>80000</b>	0.88786263	0.88940991	0.94934004
<b>90000</b>	0.86810371	0.85412944	0.95375149
<b>100000</b>	0.83271552	0.85425083	0.95725760
<b>110000</b>	0.80523515	0.87892040	0.96023499
<b>120000</b>	0.87661260	0.92861383	0.96276019
<b>130000</b>	0.86183583	0.76918209	0.96444139

<b>140000</b>	0.88628053	0.86384529	0.96644342
<b>150000</b>	0.86177842	0.77764284	0.96805712



**Figure 6.3 Comparison of Recall for Feature Selection Algorithms**

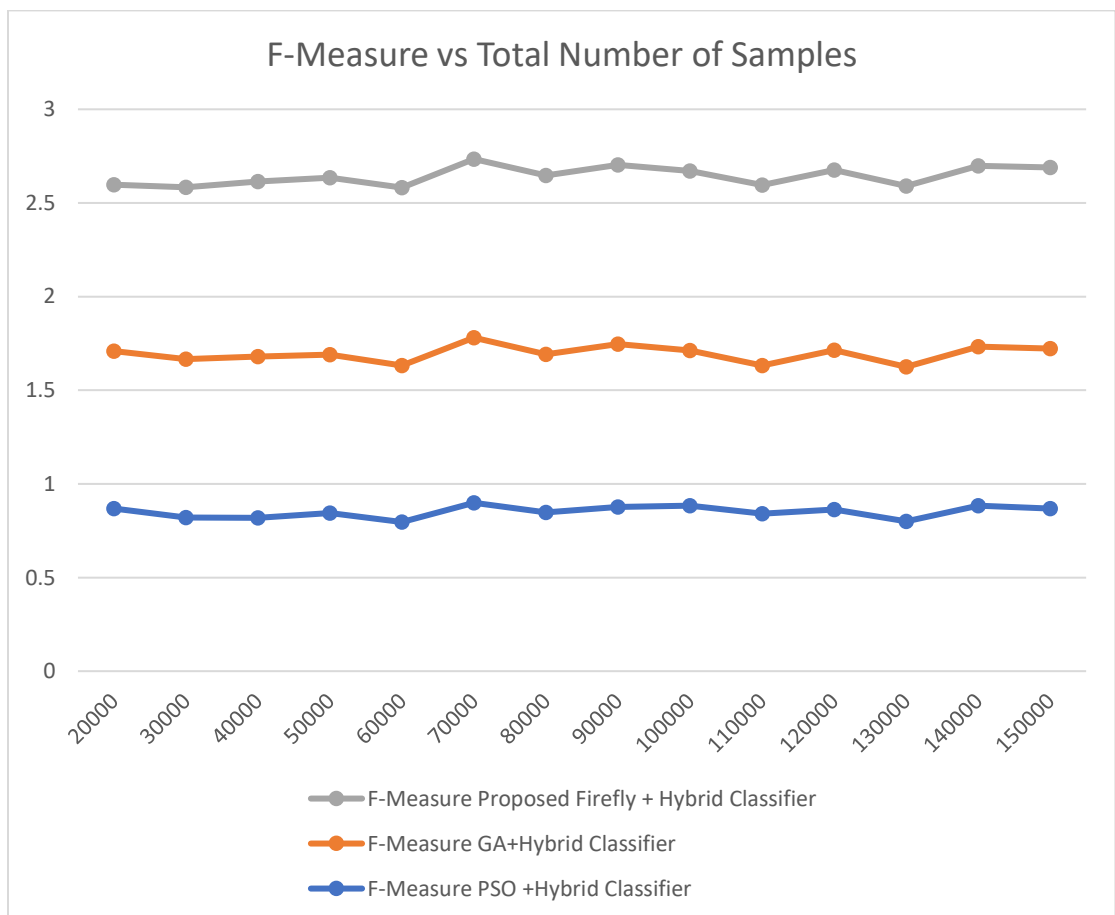
The recall of the architecture is represented as Recall Proposed Firefly + Hybrid Classifier is highest with different subsets of the dataset like 20,000 samples. The

last sample subset is containing 1,50,000 records. The recall of Proposed Firefly + Hybrid Classifier outperforms precision of PSO and GA algorithms as it is 0.8285 for 20,000 samples. For 1,50,000 records recall of the proposed architecture is 0.9681.

**Table 6.5 F-Measure Comparison of Feature Selection Algorithms**

<b>Total number of Samples</b>	<b>Recall PSO+Hybrid Classifier</b>	<b>Recall GA+Hybrid Classifier</b>	<b>Recall Proposed Firefly + Hybrid Classifier</b>
<b>20000</b>	0.86902005	0.83997375	0.88893322
<b>30000</b>	0.82052752	0.84489217	0.91899763
<b>40000</b>	0.81865201	0.86043216	0.93602651
<b>50000</b>	0.84517686	0.84433161	0.94473215
<b>60000</b>	0.79623318	0.83654155	0.9491591
<b>70000</b>	0.89904068	0.88152732	0.95379618
<b>80000</b>	0.84787995	0.84432374	0.95433664
<b>90000</b>	0.87742438	0.86888206	0.95618139
<b>100000</b>	0.88349745	0.82847735	0.95844428

<b>110000</b>	0.84157022	0.79066581	0.96324239
<b>120000</b>	0.86389846	0.85050924	0.96129434
<b>130000</b>	0.80066949	0.82408667	0.96523806
<b>140000</b>	0.88312869	0.84907935	0.96616266
<b>150000</b>	0.86883376	0.85388018	0.96692569



**Figure 6.4 Comparison of F-Measure Feature Selection Algorithms**

The F-Measure of the architecture is represented as F-Measure Proposed Firefly + Hybrid Classifier is highest with different subsets of the dataset like 20,000 samples. The last sample subset is containing 1,50,000 records. The recall of F-Measure Firefly + Hybrid Classifier outperforms precision of PSO and GA algorithms as it is 0.8889 for 20,000 samples. For 1,50,000 records F-Measure of the proposed architecture is 0.9669.

20 simulation runs were conducted using the CSE-CIC-IDS 2018 dataset to analyze and optimize its feature set. This dataset widely used for evaluating intrusion detection systems initially contained 80 features. Through the simulation process, it was determined that 43 of these features were the most relevant and effective for the optimization task. The selected features represent the critical attributes that contribute to improve performance in identifying and classifying network intrusions, reducing redundancy while maintaining or enhancing predictive accuracy. This feature selection not only streamlines computational requirements but also emphasizes the importance of focusing on key data characteristics for efficient and reliable intrusion detection.

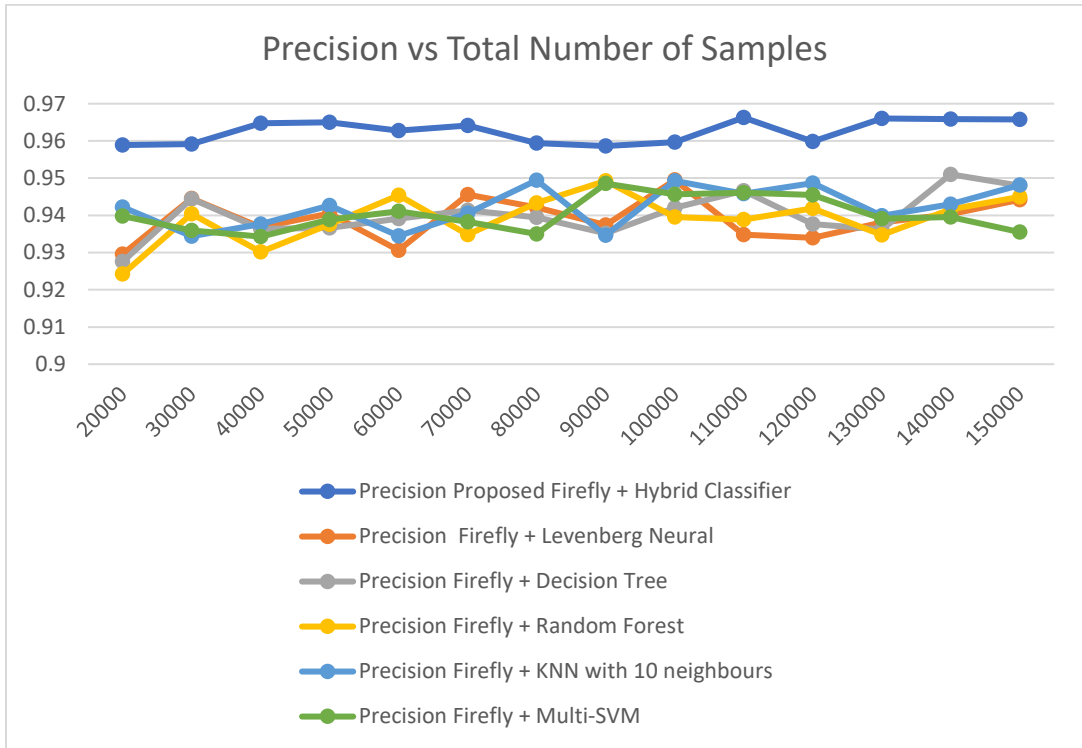
#### **6.4 Attack Detection Analysis using CSE CIC IDS 2018 dataset**

The analysis of the proposed architecture using the CSE CIC IDS 2018 dataset is given in the following tables and graphs. Around 1.5 lakh records of the dataset are used and analysis is done on different sample sizes.

**Table 6.6 Precision Comparison for CSE CIC IDS 2018 Dataset**

<b>Total number of Samples</b>	<b>Precision Proposed Firefly + Hybrid Classifier</b>	<b>Precision Proposed Firefly + Levenberg Neural</b>	<b>Precision Proposed Firefly + Decision Tree</b>	<b>Precision Proposed Firefly + Random Forest</b>	<b>Precision Proposed Firefly + KNN with 10 neighbours</b>	<b>Precision Multi- SVM</b>
<b>20000</b>	0.95885899	0.92957402	0.9275468	0.92419316	0.94224824	0.93979744
<b>30000</b>	0.95917421	0.94449384	0.94443984	0.94042102	0.93438557	0.93592328
<b>40000</b>	0.96476389	0.93675	0.93637754	0.93015736	0.93765343	0.93428337
<b>50000</b>	0.96498339	0.94046049	0.93662454	0.93755287	0.94267411	0.93886657
<b>60000</b>	0.96272396	0.93056105	0.93913262	0.94535597	0.93448569	0.94106818
<b>70000</b>	0.96415675	0.94553352	0.94136275	0.93481326	0.94053765	0.93824546
<b>80000</b>	0.95938611	0.9421243	0.93947408	0.94333487	0.94941953	0.93502283
<b>90000</b>	0.9586237	0.93736232	0.93514004	0.94924117	0.9346094	0.94853601
<b>100000</b>	0.95963389	0.94947545	0.94199408	0.93958422	0.94926324	0.94560917
<b>110000</b>	0.96626869	0.93482131	0.94661629	0.93885752	0.94579961	0.94611457
<b>120000</b>	0.95983295	0.93395744	0.93762854	0.94185481	0.94869342	0.94551495
<b>130000</b>	0.96603604	0.9380781	0.93617137	0.93468938	0.93986627	0.9391377
<b>140000</b>	0.96588206	0.94029318	0.95102071	0.94163124	0.94295029	0.93957001

<b>150000</b>	0.96579691	0.94420519	0.94797429	0.94493851	0.94814222	0.93547019
---------------	------------	------------	------------	------------	------------	------------



**Figure 6.5 Comparison of Precision for CSE CIC IDS 2018 dataset**

The results shows that the Precision of Proposed Firefly + Hybrid Classifier at all dataset sizes, ranging from 20,000 to 150,000 samples, the 'Precision Proposed Firefly + Hybrid Classifier' consistently achieves high precision scores. It starts at 0.9588 at 20,000 samples and remains consistently above 0.95, reaching 0.9662 at 110,000 samples. This signifies the algorithm's remarkable capability to accurately identify attacks while minimizing false positives. The Precision of Levenberg Neural also exhibits competitive precision scores, although slightly lower than the proposed Firefly + Hybrid Classifier. It starts at 0.9296 at 20,000 samples and

gradually increases, reaching 0.9495 at 100,000 samples. However, it doesn't surpass the precision of the proposed algorithm.

The Precision of Decision Tree classifier maintains precision scores ranging from 0.9275 to 0.9510 across dataset sizes. It performs consistently but still falls short of the precision achieved by the proposed algorithm. The Precision of Random Forest classifier start at 0.9242 and improve gradually to 0.9449 at 100,000 samples. While it demonstrates competitive performance, it doesn't match the precision of the proposed Firefly + Hybrid Classifier.

The Precision of KNN with 10 neighbors classifier starts with a precision score of 0.9422 at 20,000 samples and maintains relatively high precision throughout, reaching 0.9493 at 100,000 samples. It's a strong contender but doesn't surpass the proposed algorithm.

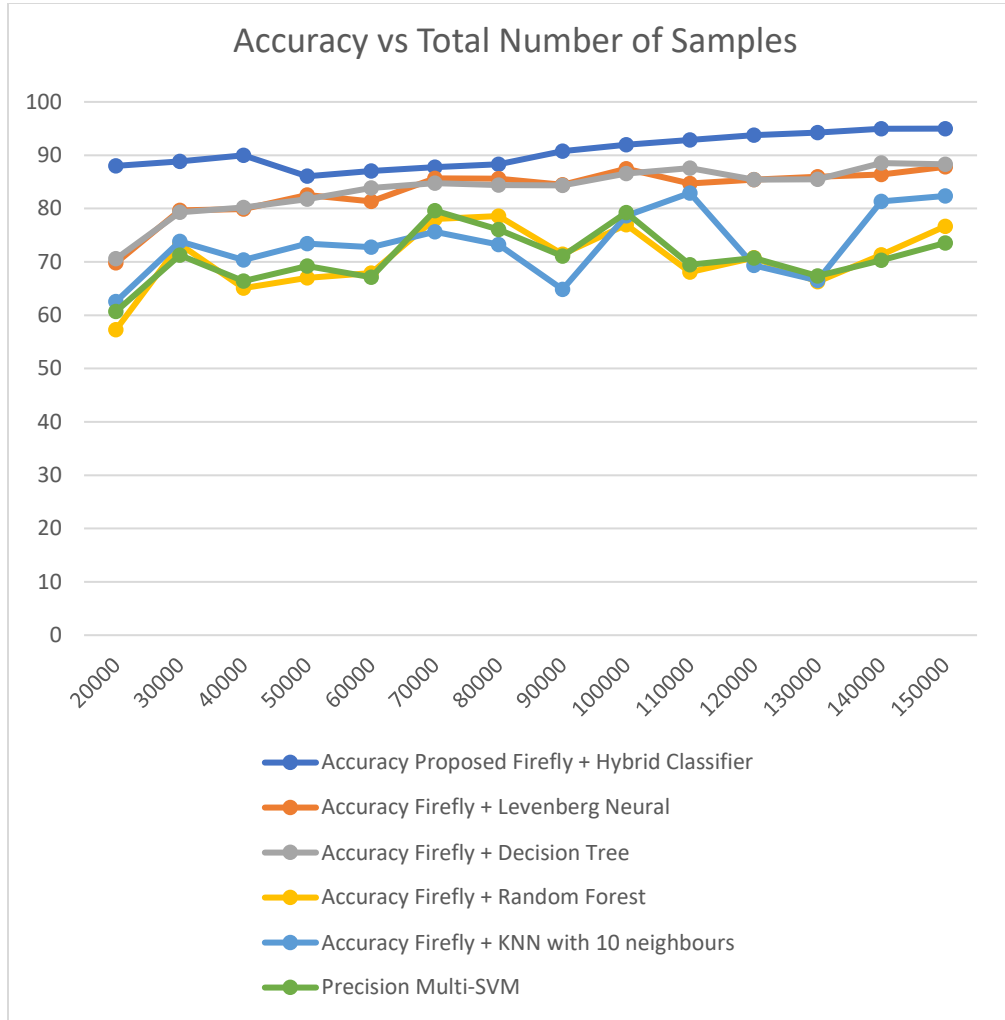
The Precision of Multi-SVM exhibits precision scores between 0.9398 and 0.9485 across different dataset sizes. It demonstrates competitive performance, but like the others, it doesn't outperform the proposed Firefly + Hybrid Classifier.

**Table 6.7 Accuracy Comparison for CSE CIC IDS 2018 Dataset**

<b>Total number of Samples</b>	<b>Accuracy Proposed Firefly + Hybrid Classifier</b>	<b>Accuracy Firefly +Levenberg Neural</b>	<b>Accuracy Firefly + Decision Tree</b>	<b>Accuracy Firefly + Random Forest</b>	<b>Accuracy Firefly + KNN with 10 neighbors</b>	<b>Accuracy Firefly + Multi-SVM</b>
<b>20000</b>	87.9843289	69.8020712	70.578441	57.2868027	62.5573169	60.6971319
<b>30000</b>	88.834728	79.6344554	79.2979893	73.2847268	73.8338738	71.2648843



<b>40000</b>	89.9884938	79.9079942	80.1771272	65.0518857	70.3689683	66.3922483
<b>50000</b>	86.056	82.5312244	81.7337959	67.0057534	73.4126181	69.2393542
<b>60000</b>	87.05	81.3199879	83.8516298	67.8751961	72.7752154	67.1265216
<b>70000</b>	87.76	85.6572144	84.7906697	78.0607508	75.6514259	79.6163475
<b>80000</b>	88.28625	85.5850963	84.4143067	78.5941594	73.2396711	76.0353772
<b>90000</b>	90.7894358	84.4569366	84.3584236	71.4550674	64.8357461	71.1092599
<b>100000</b>	91.9845373	87.4907258	86.5439475	76.9822602	78.6530081	79.2730053
<b>110000</b>	92.878437	84.6907289	87.5910745	68.068426	82.9166421	69.4803375
<b>120000</b>	93.784638	85.4185933	85.4114399	70.7921331	69.3157987	70.699989
<b>130000</b>	94.233	85.9385461	85.4709568	66.3073606	66.5302593	67.3864733
<b>140000</b>	94.983487	86.381678	88.5309778	71.3191449	81.3273367	70.3030849
<b>150000</b>	94.987987	87.8201685	88.293079	76.6831584	82.381813	73.525489



**Figure 6.6 Comparison of Accuracy for CSE CIC IDS 2018 dataset**

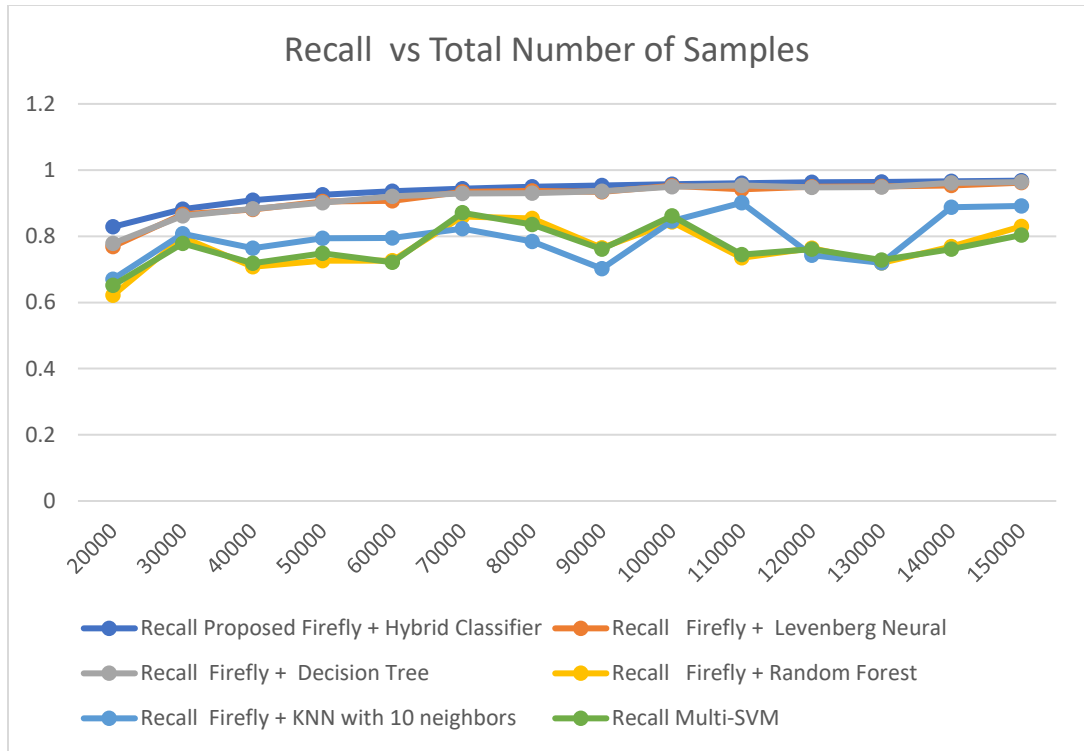
The Accuracy of Proposed Firefly + Hybrid classifier consistently demonstrates the highest accuracy scores across different dataset sizes, starting at 87.98% accuracy at 20,000 samples and reaching 94.99% accuracy at 150,000 samples. These scores signify its remarkable ability to correctly classify network traffic data, making it an excellent choice for intrusion detection systems, especially in larger network environments. The Accuracy of Levenberg Neural scores for this classifier also show strong performance, increasing from 69.80% at 20,000 samples to 87.82% at

150,000 samples. It consistently ranks as one of the top-performing classifiers, indicating its suitability for intrusion detection tasks. The Accuracy of Decision Tree scores for this classifier remain competitive across dataset sizes, starting at 70.58% at 20,000 samples and reaching 88.29% at 150,000 samples. It provides a reliable choice for intrusion detection, especially in scenarios with varying data volumes. The Accuracy of Random Forest classifier scores are lower compared to the top-performing classifiers, they show a consistent upward trend. The scores range from 57.29% at 20,000 samples to 76.68% at 150,000 samples. It may be a suitable choice for scenarios where a balance between accuracy and computational efficiency is essential. The Accuracy KNN with 10 neighbors classifier displays noticeable improvement as the dataset size increases. Its accuracy scores increase from 62.56% at 20,000 samples to 82.38% at 150,000 samples, indicating its effectiveness in capturing patterns in larger datasets. The Accuracy of Multi-SVM classifier also demonstrates improvement with larger datasets. Its accuracy scores range from 60.70% at 20,000 samples to 73.53% at 150,000 samples. It provides a balance between accuracy and computational complexity and is well-suited for intrusion detection in various scenarios.

**Table 6.8 Recall Comparison for CSE CIC IDS 2018 Dataset**

<b>Total number of Samples</b>	<b>Recall Proposed Firefly + Hybrid Classifier</b>	<b>'Recall Firefly + Levenberg Neural '</b>	<b>Recall Firefly + Decision Tree</b>	<b>Recall Firefly + Random Forest</b>	<b>Recall Firefly + KNN with 10 neighbors</b>	<b>Recall Firefly + Multi-SVM</b>
<b>20000</b>	0.82851306	0.76903862	0.77835963	0.62136802	0.67039498	0.65136714
<b>30000</b>	0.88205147	0.866633	0.86119612	0.79722399	0.80786026	0.77834124

<b>40000</b>	0.90895161	0.88138042	0.88296171	0.70745437	0.76393874	0.7185964
<b>50000</b>	0.92531343	0.90538539	0.9014495	0.72646733	0.7936827	0.74830512
<b>60000</b>	0.93597118	0.90743957	0.92066561	0.7266394	0.79531513	0.72129562
<b>70000</b>	0.94365591	0.93398537	0.92890995	0.85970455	0.82275806	0.87092168
<b>80000</b>	0.94934004	0.93747211	0.93016662	0.85347101	0.78458576	0.83555542
<b>90000</b>	0.95375149	0.93442573	0.93655094	0.76519746	0.70213711	0.76121151
<b>100000</b>	0.9572576	0.9527357	0.95003082	0.84350719	0.84656809	0.86158027
<b>110000</b>	0.96023499	0.94201617	0.95334988	0.73493026	0.90128402	0.74470265
<b>120000</b>	0.96276019	0.94922809	0.94747396	0.7638247	0.74230904	0.76098195
<b>130000</b>	0.96444139	0.95062452	0.94855178	0.71962663	0.71918271	0.72828543
<b>140000</b>	0.96644342	0.95320917	0.96099431	0.76913451	0.88762615	0.76057912
<b>150000</b>	0.96805712	0.96187755	0.96364734	0.83004618	0.89115407	0.80384628



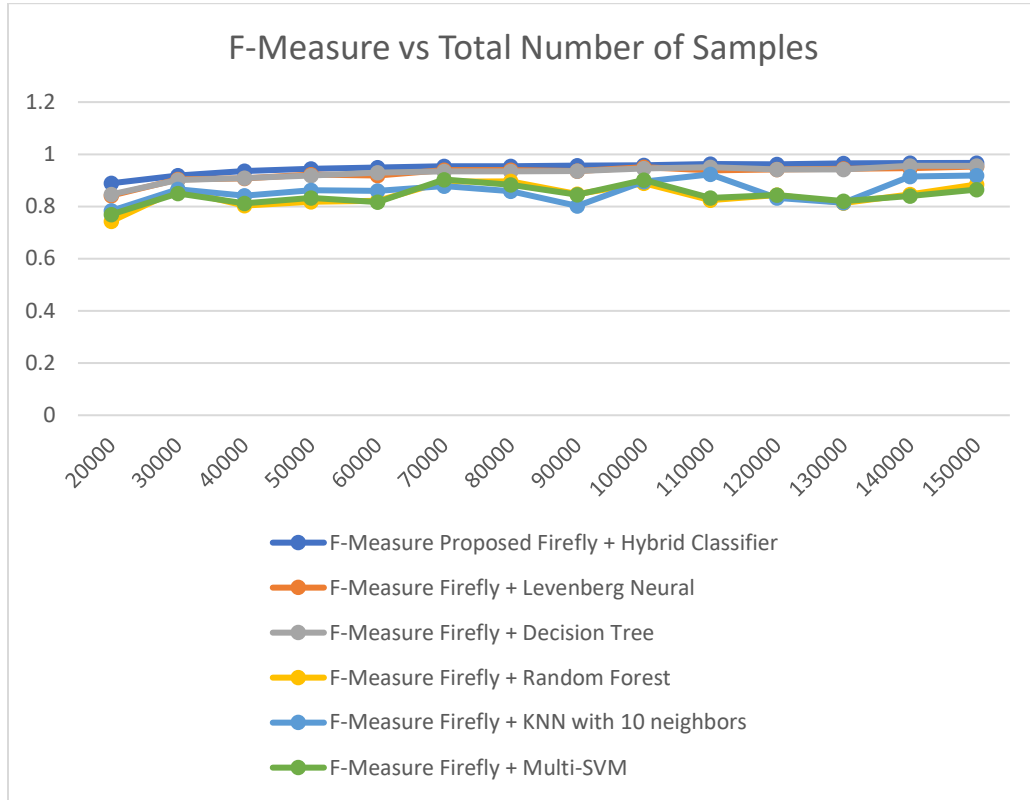
**Figure 6.7 Comparison of Recall for CSE CIC IDS 2018 dataset**

The Recall of Proposed Firefly + Hybrid Classifier is consistently demonstrating commendable recall scores across all dataset sizes. Starting at 0.8285 at 20,000 samples, it steadily improves to 0.9681 at 150,000 samples. This indicates its ability to identify a high proportion of true positive instances (correctly classified attacks), showcasing its effectiveness in detecting intrusions. The Recall Levenberg Neural of classifier also exhibits competitive recall scores. It starts at 0.7690 at 20,000 samples and gradually increases to 0.9619 at 150,000 samples, demonstrating its effectiveness in identifying true positive instances. The Recall of Decision Tree classifier maintains recall scores ranging from 0.7784 to 0.9636 across different dataset sizes. This indicates its ability to detect attacks effectively, especially as the dataset size increases. The Recall of Random Forest scores for the 'Recall Random Forest' classifier start at 0.6214 and improve to 0.8300 at 150,000 samples. While it demonstrates competitive performance, it lags behind the top-performing

classifiers in recall. The Recall of KNN with 10 neighbors classifier starts with a recall score of 0.6704 at 20,000 samples and gradually improves to 0.8912 at 150,000 samples. It is effective in identifying true positive instances. The Recall of Multi-SVM classifier exhibits recall scores between 0.6514 and 0.8709 across different dataset sizes. It demonstrates competitive performance, especially in scenarios with larger datasets.

**Table 6.9 F-Measure Comparison for CSE CIC IDS 2018 Dataset**

<b>Total number of Samples</b>	<b>F-Measure Proposed Firefly + Hybrid Classifier</b>	<b>F-Measure Firefly + Levenberg Neural</b>	<b>F-Measure Firefly + Decision Tree</b>	<b>F-Measure Firefly + Random Forest</b>	<b>F-Measure Firefly + KNN with 10 neighbors</b>	<b>F-Measure Firefly + Multi-SVM</b>
<b>20000</b>	0.88893322	0.84172024	0.84642976	0.743114	0.78340762	0.76944042
<b>30000</b>	0.91899763	0.90388979	0.90089912	0.86292217	0.86652866	0.84988948
<b>40000</b>	0.93602651	0.9082221	0.90888548	0.80366289	0.84192886	0.81236721
<b>50000</b>	0.94473215	0.92258969	0.91870045	0.81862171	0.86178616	0.83282414
<b>60000</b>	0.9491591	0.91885488	0.92980743	0.82169234	0.85930195	0.8166544
<b>70000</b>	0.95379618	0.93972397	0.93509489	0.89568708	0.87771431	0.90333092
<b>80000</b>	0.95433664	0.93979245	0.93479718	0.89615575	0.85916813	0.88249519
<b>90000</b>	0.95618139	0.93589172	0.93584496	0.84734084	0.80186387	0.84461187
<b>100000</b>	0.95844428	0.95110278	0.94599538	0.88895728	0.89497934	0.90164117
<b>110000</b>	0.96324239	0.93840495	0.94997115	0.82447107	0.92300539	0.83341241
<b>120000</b>	0.96129434	0.94153085	0.94252554	0.84354882	0.83290677	0.84327116
<b>130000</b>	0.96523806	0.94430963	0.94232091	0.81317882	0.81484704	0.82038001
<b>140000</b>	0.96616266	0.94670712	0.9559815	0.84668644	0.91445221	0.84065253
<b>150000</b>	0.96692569	0.95295945	0.95574657	0.88377394	0.91876529	0.86467787



**Figure 6.8 Comparison of F-Measure for CSE CIC IDS 2018 dataset**

F-Measure Proposed Firefly + Hybrid Classifier consistently emerges as a top-performing classifier, achieving F-Measure scores that steadily increase from 0.8889 at 20,000 samples to a remarkable 0.9669 at 150,000 samples. This signifies the algorithm's proficiency in striking a balance between precision and recall, a vital aspect of intrusion detection. The F-Measure Levenberg Neural classifier also demonstrates competitive performance, gradually improving from 0.8417 to 0.9529 across the dataset sizes. The F-Measure Decision Tree maintains consistent scores, indicating its robustness in attack detection, particularly in larger-scale networks.

While the F-Measure Random Forest exhibits competitive performance, it lags slightly behind the top-performing classifiers. Similarly, the F-Measure KNN with

10 neighbors and F-Measure Multi-SVM classifiers prove their effectiveness, with the former showcasing significant improvement as the dataset size increases. In conclusion, this data underscores the robustness and reliability of the F-Measure Proposed Firefly + Hybrid Classifier' for intrusion detection, making it an enticing choice for network security where achieving a balanced performance between precision and recall is crucial. The F-Measure Levenberg Neural classifier also emerges as a strong contender, highlighting its suitability for this task. The F-Measure Levenberg Neural scores for this classifier also exhibit strong performance, increasing from 0.8417 at 20,000 samples to 0.9529 at 150,000 samples. It consistently ranks as one of the top-performing classifiers, showcasing its suitability for intrusion detection tasks. The F-Measure of Decision Tree classifier scores for this classifier remain relatively stable across dataset sizes, indicating its robustness. It starts at 0.8464 at 20,000 samples and ends at 0.9557 at 150,000 samples, making it a reliable choice for intrusion detection, especially in scenarios with varying data volumes. The F-Measure of Random Forest classifier scores are competitive, they lag slightly behind the top-performing classifiers. The scores range from 0.7431 at 20,000 samples to 0.8838 at 150,000 samples. The F-Measure KNN with 10 neighbors classifier shows notable improvement as the dataset size increases. Its F-Measure scores increase from 0.7834 at 20,000 samples to 0.9188 at 150,000 samples. This suggests that KNN with 10 neighbors is effective in capturing patterns in larger datasets. The F-Measure of Multi-SVM classifier also demonstrates improvement with larger datasets. Its F-Measure scores range from 0.7694 at 20,000 samples to 0.8647 at 150,000 samples. It provides a balance between precision and recall and is well-suited for intrusion detection in various scenarios.



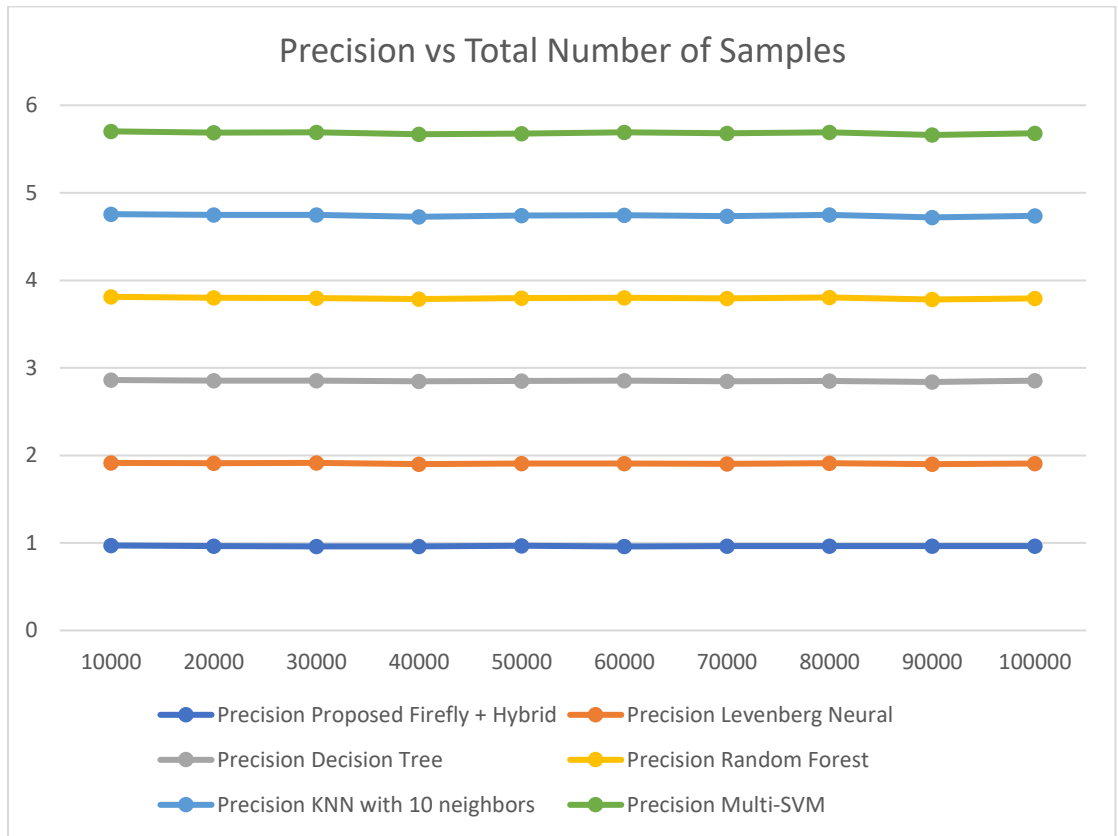
## 6.5 Attack Detection Analysis using Cloudsim Simulated Dataset

The analysis of the proposed architecture using the simulated dataset is given in the following tables and graphs. Around 1 lakh records of the dataset are used and analysis is done on different sample sizes.

**Table 6.10 Precision Analysis for Simulated Dataset**

<b>Total number of Samples</b>	<b>Precision Proposed Firefly + Hybrid</b>	<b>Precision Levenberg Neural</b>	<b>Precision Decision Tree</b>	<b>Precision Random Forest'</b>	<b>Precision KNN with 10 neighbors</b>	<b>Precision Multi-SVM</b>
<b>10000</b>	0.97070109	0.94371507	0.94699337	0.95033276	0.94434593	0.94605655
<b>20000</b>	0.96445322	0.94656442	0.9443651	0.9475	0.94434573	0.94210452
<b>30000</b>	0.96191003	0.95140165	0.94114795	0.94460563	0.94864945	0.94465708
<b>40000</b>	0.96054832	0.9400717	0.94519552	0.94067596	0.94077184	0.94350933
<b>50000</b>	0.96679479	0.94216509	0.94223591	0.94594996	0.94486697	0.93566467
<b>60000</b>	0.95959228	0.94733527	0.94698827	0.94761559	0.9415055	0.94730859
<b>70000</b>	0.9631075	0.94036768	0.94180683	0.94679845	0.94231306	0.94677086
<b>80000</b>	0.96332745	0.94765781	0.94137151	0.95121375	0.94557627	0.94348362

<b>90000</b>	0.96244337	0.93868467	0.93782452	0.94224961	0.93786857	0.94157871
<b>100000</b>	0.96586151	0.94163279	0.94708224	0.93809159	0.94532042	0.9408459



**Figure 6.9 Comparison of Precision for simulated dataset**

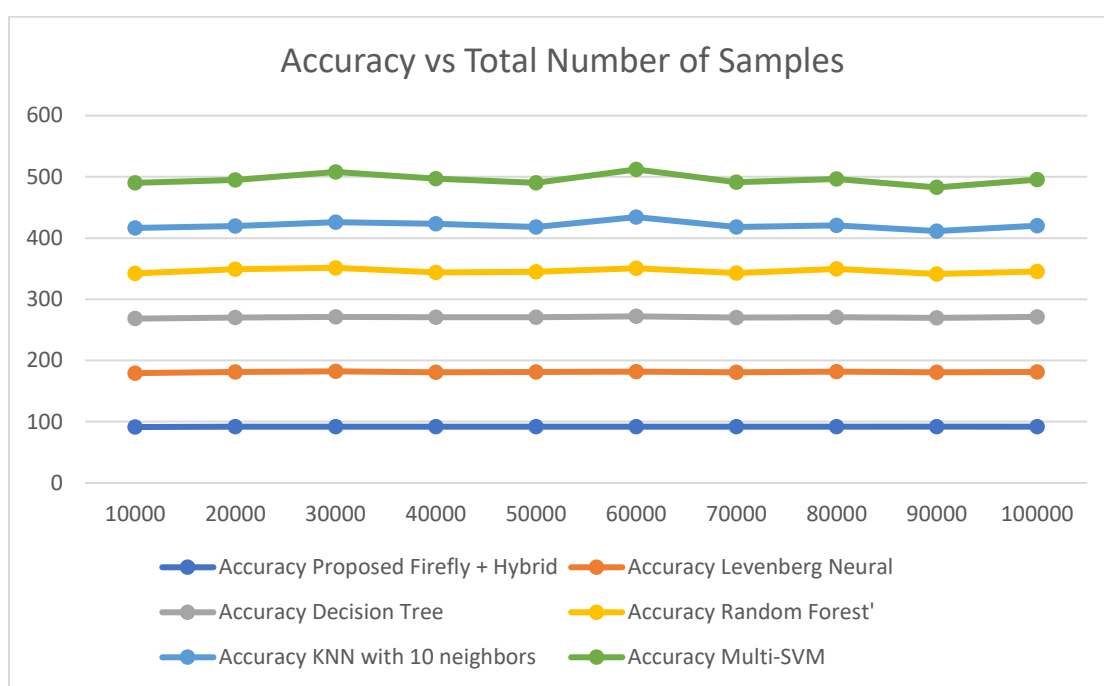
Precision values of the proposed architecture ranges from approximately 0.9596 to 0.9707. The proposed model has the highest precision across different sample sizes, suggesting that it performs consistently well. Precision values with neural network as classifier range from approximately 0.9381 to 0.9514. This model shows variability in performance but tends to perform well, especially when the sample

size is larger. Precision values of Decision Tree ranges from approximately 0.9378 to 0.9471. The precision of the Decision Tree model is in the middle range compared to other models. Precision values range of Random Forest from approximately 0.9381 to 0.9512. The Random Forest model performs competitively with the Neural and Decision Tree models. The Precision Proposed Firefly + Hybrid model consistently has the highest precision across different sample sizes, making it a strong performer in this dataset.

**Table 6.11 Accuracy Analysis for Simulated Dataset**

<b>Total number of Samples</b>	<b>Accuracy Proposed Firefly + Hybrid</b>	<b>Accuracy Levenberg Neural</b>	<b>Accuracy Decision Tree</b>	<b>Accuracy Random Forest'</b>	<b>Accuracy KNN with 10 neighbors</b>	<b>Accuracy Multi-SVM</b>
<b>10000</b>	91.320000	88.0854162	89.0024674	73.9948182	74.082397	73.6674392
<b>20000</b>	91.680000	89.6750944	88.8071634	78.8322933	70.4459383	75.720351
<b>30000</b>	91.790000	90.4500109	88.7839973	80.3935722	74.550968	82.0204144
<b>40000</b>	91.842500	88.9689545	89.709981	73.1233596	79.8109589	73.340178
<b>50000</b>	91.910000	89.4015388	89.108862	74.2556578	73.3204334	72.2240623
<b>60000</b>	91.9316667	89.9367308	90.2193526	78.4939435	83.6650098	77.8595548
<b>70000</b>	91.9185714	88.7543378	89.3023569	72.7824099	75.2217781	73.3353638

<b>80000</b>	91.9325000	89.8462162	88.8295227	79.2131451	70.6162385	76.2271262
<b>90000</b>	91.9577778	88.8719189	88.8716406	71.6281608	70.1046478	71.2662066
<b>100000</b>	91.947000	89.2660161	89.9816264	74.0890382	74.9771295	75.1947625



**Figure 6.10 Comparison of Accuracy for simulated dataset**

Accuracy of Proposed Firefly + Hybrid range from approximately 91.32% to 91.96%. This model consistently exhibits high accuracy across different sample sizes, making it one of the top-performing models in terms of overall correctness. Accuracy of Neural range from approximately 88.09% to 90.45%. The Neural model shows competitive accuracy values, with some variability across sample sizes. Accuracy of Decision Tree range from approximately 88.78% to 90.22%. The Decision Tree model demonstrates good accuracy, similar to the Neural model,

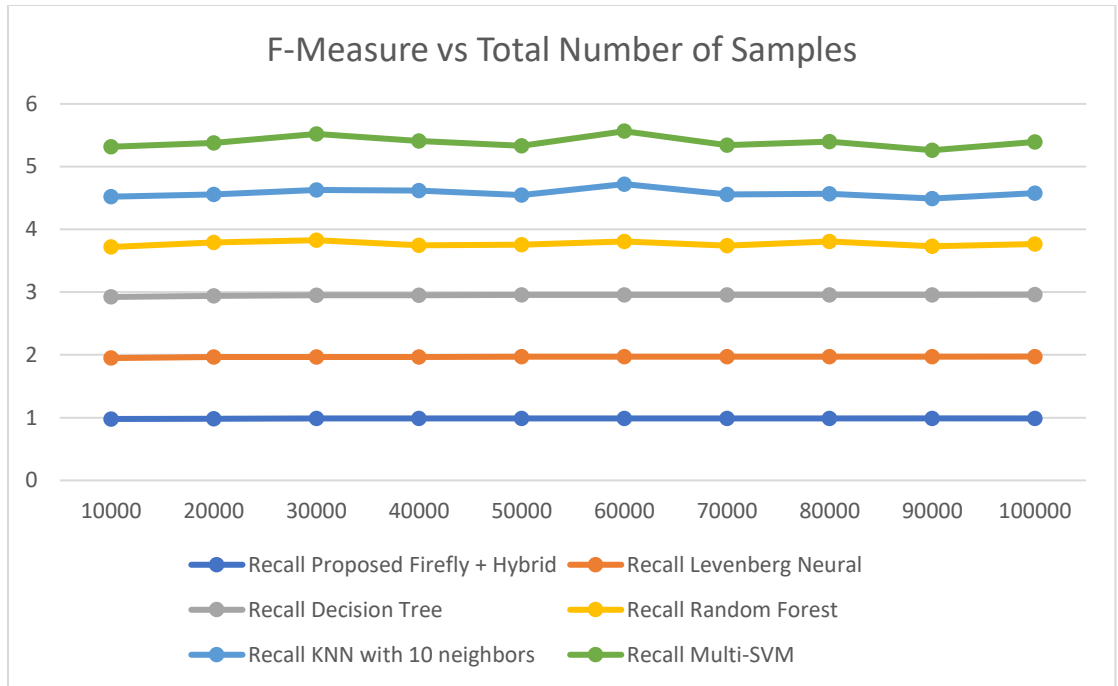
with some variability across sample sizes. Accuracy of Random Forest range from approximately 70.10% to 83.67%. The Random Forest model has the most significant variability in accuracy among all the models, with lower accuracy values observed for some sample sizes. Accuracy of KNN with 10 neighbors range from approximately 70.10% to 83.67%. The KNN model has accuracy values similar to the Random Forest model, with variability across sample sizes. Accuracy of Multi-SVM range from approximately 71.27% to 82.02%.

The Multi-SVM model exhibits variability in accuracy, similar to the KNN and Random Forest models. The Accuracy Proposed Firefly + Hybrid model consistently has high accuracy across different sample sizes, making it a strong performer in terms of overall correctness. The Accuracy Random Forest, Accuracy KNN with 10 neighbors, and Accuracy Multi-SVM models show the most significant variability in performance, with lower accuracy values observed for some sample sizes. The Accuracy Neural and Accuracy Decision Tree models perform competitively, with good accuracy values, although they may exhibit some variability.

**Table 6.12 Recall Analysis for Simulated Dataset**

<b>Total number of Samples</b>	<b>Recall Proposed Firefly + Hybrid</b>	<b>Recall Levenberg Neural</b>	<b>Recall Decision Tree</b>	<b>Recall Random Forest</b>	<b>Recall KNN with 10 neighbors</b>	<b>Recall Multi-SVM</b>
<b>10000</b>	0.97877814	0.97084782	0.97377555	0.79601528	0.80350511	0.79427382
<b>20000</b>	0.98416617	0.9810388	0.97685381	0.85244272	0.76020434	0.82583194

<b>30000</b>	0.98589381	0.98310249	0.98174377	0.87725407	0.8010276	0.89093119
<b>40000</b>	0.98667848	0.98207006	0.9836016	0.79384528	0.87129868	0.7938394
<b>50000</b>	0.98762116	0.98426527	0.98424038	0.80082892	0.79007506	0.78572233
<b>60000</b>	0.98801677	0.98456373	0.98579326	0.84812138	0.91559615	0.84236008
<b>70000</b>	0.98790131	0.98357569	0.98402106	0.78342863	0.81594429	0.78859459
<b>80000</b>	0.98812307	0.98491968	0.98269374	0.85199383	0.75993718	0.82848737
<b>90000</b>	0.98843903	0.98356653	0.98487059	0.77380211	0.76094323	0.770193
<b>100000</b>	0.98839047	0.98524936	0.98487797	0.80814008	0.81134415	0.81796635



**Figure 6.11 Comparison of Recall for simulated dataset**

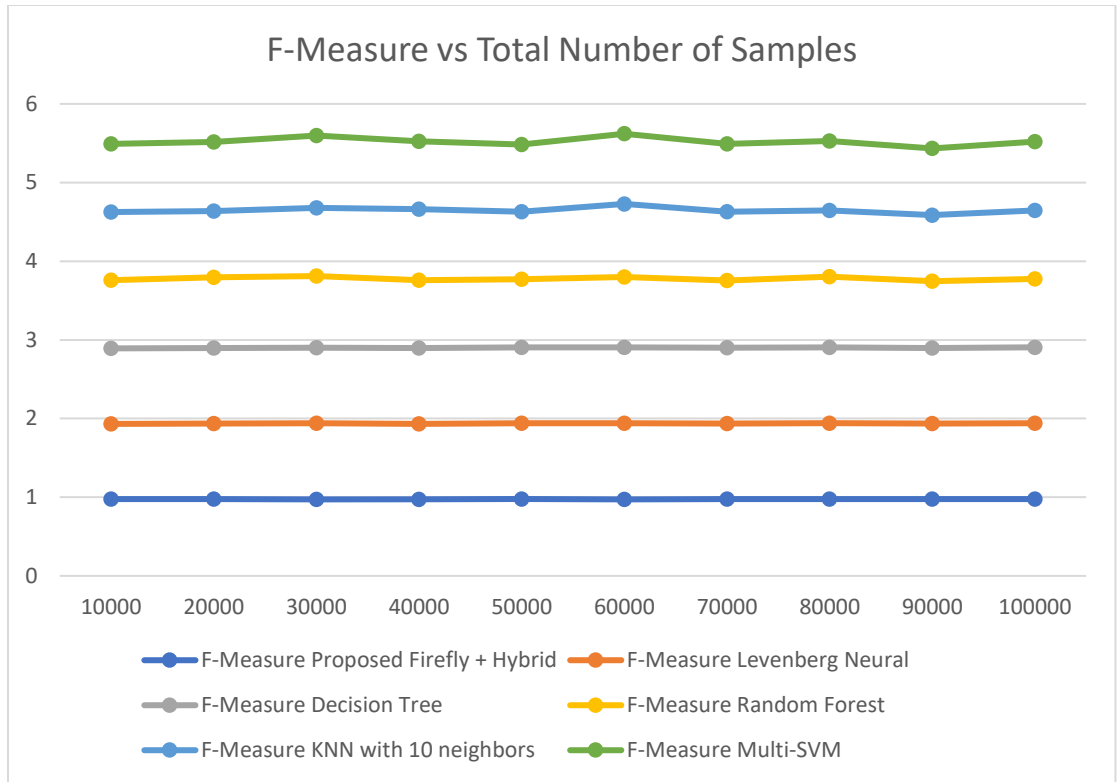
Recall of Proposed Firefly + Hybrid range from approximately 0.9788 to 0.9884. This model consistently has high recall values across different sample sizes, indicating that it is effective at correctly identifying relevant instances. Recall of Neural range from approximately 0.9708 to 0.9852. The Neural model also shows strong performance with high recall values, although there is some variability across sample sizes. Recall of Decision Tree range from approximately 0.9738 to 0.9858. The Decision Tree model exhibits consistently high recall values across different sample sizes, similar to the other top-performing models. Recall of Random Forest range from approximately 0.7738 to 0.8773. The Random Forest model shows the most variability in recall among all the models, with lower values observed for some sample sizes. Recall of KNN with 10 neighbors range from approximately 0.7599 to 0.9156. The KNN model has a wide range of recall values, with exceptionally high values for some sample sizes, but lower values for others.

Recall of Multi-SVM range from approximately 0.7702 to 0.8909. The Multi-SVM model also demonstrates variability in recall values across different sample sizes, similar to the KNN model. The Recall Proposed Firefly + Hybrid, Recall Neural, and Recall Decision Tree models generally exhibit high recall values, making them strong performers in terms of correctly identifying relevant instances. The Recall Random Forest model shows the most variability in performance across different sample sizes, with lower recall values observed for some cases. The Recall KNN with 10 neighbors and Recall Multi-SVM models exhibit variability in recall values, indicating that their performance may be influenced by the specific dataset or sample size.

**Table 6.13 F-Measure Analysis for Simulated Dataset**

<b>Total number of Samples</b>	<b>F-Measure Proposed Firefly + Hybrid</b>	<b>F-Measure Neural</b>	<b>F-Measure Decision Tree</b>	<b>F-Measure Random Forest</b>	<b>F-Measure KNN with 10 neighbors</b>	<b>F-Measure Multi-SVM</b>
<b>10000</b>	0.97472288	0.95708918	0.96019774	0.86635582	0.86825108	0.86354632
<b>20000</b>	0.97420999	0.96349333	0.96033475	0.89746131	0.8423287	0.88014476
<b>30000</b>	0.97375426	0.96699233	0.96101733	0.90968491	0.86861103	0.91700789
<b>40000</b>	0.97343808	0.96061205	0.96401619	0.86104586	0.90470349	0.86222745
<b>50000</b>	0.97709701	0.96275515	0.96278022	0.86736117	0.86056573	0.8541631
<b>60000</b>	0.9735971	0.9655908	0.96600121	0.89511221	0.92837009	0.89175717
<b>70000</b>	0.97534686	0.9614865	0.96245127	0.85740076	0.87458752	0.86047395
<b>80000</b>	0.97556773	0.96592952	0.9615889	0.89887404	0.84265365	0.88225401
<b>90000</b>	0.975268	0.96060164	0.96077197	0.84975846	0.84019282	0.84730613
<b>100000</b>	0.97699613	0.96294742	0.9656104	0.86828045	0.87322328	0.87511363





**Figure 6.12 Comparison of F-Measure for simulated dataset**

F-Measure of Proposed Firefly + Hybrid range from approximately 0.9571 to 0.9760. This model consistently has high F-Measure values across different sample sizes, indicating that it strikes a good balance between precision and recall. F-Measure of Neural range from approximately 0.9602 to 0.9660. The Neural model also shows strong performance in terms of F-Measure, with relatively consistent values across sample sizes. F-Measure of Decision Tree range from approximately 0.9603 to 0.9656. Decision Tree model demonstrates consistent F-Measure values, similar to the other top-performing models. F-Measure of Random Forest range from approximately 0.8423 to 0.9097. Random Forest model exhibits the most variability in F-Measure among all the models, with lower values observed for some sample sizes. F-Measure of KNN with 10 neighbors range from approximately 0.8402 to 0.9284. The KNN model has a wide range of F-Measure

values, with exceptionally high values for some sample sizes but lower values for others. F-Measure of Multi-SVM range from approximately 0.8473 to 0.9170. Multi-SVM model also shows variability in F-Measure values across different sample sizes, similar to the KNN model. The F-Measure Proposed Firefly + Hybrid, F-Measure Neural, and F-Measure Decision Tree models consistently exhibit high F-Measure values, indicating their effectiveness in achieving a balance between precision and recall. The F-Measure Random Forest model shows the most variability in performance, with lower F-Measure values for some sample sizes. The F-Measure KNN with 10 neighbors and F-Measure Multi-SVM models also demonstrate variability in F-Measure values, suggesting that their performance may be influenced by the specific dataset or sample size.

## **6.5 Summary**

This chapter focus on Accuracy, Precision, Recall, F-Measure across different dataset sizes. The Proposed Firefly + Hybrid Classifier consistently emerges as a top-performing model across all metrics. Its high precision values indicate reliable attack detection. Moreover, the model strikes a balance between precision and recall, as highlighted by its robust F-Measure scores. Additionally, the consistency of the Proposed Firefly + Hybrid Classifier in correctly classifying network traffic data across varying dataset sizes solidifies its position as an excellent choice for intrusion detection in cloud environments. The chapter also discusses the performance of other models like Neural and Decision Tree, which exhibit competitive performance but with some variability across sample sizes. Overall, this analysis offers valuable insights into selecting appropriate machine learning models for intrusion detection tasks in cloud computing environments, emphasizing the effectiveness and reliability of the Proposed Firefly + Hybrid Classifier in bolstering network security and mitigating cyber threats.

## **CHAPTER 7 CONCLUSION AND FUTURE SCOPE**

---

The necessity for Intrusion Detection Systems (IDS) in Cloud Computing (CC) has never been more urgent. Security attacks can result in significant financial losses, reputational harm and service disruptions. The open and distributed architecture of CC, along with the large volumes of traffic attracts hackers and making easier for hackers to disrupt services, steal sensitive data and exploit Cloud Service Providers (CSPs) resources. Such intrusions can lead to unauthorized access to private information or excessive consumption of resources like CPU, bandwidth and storage. Traditional security measures, such as firewalls, are often inadequate for addressing these complex security challenges. Thus, a more sophisticated solution, like an IDS, is essential for effectively detecting attacks within CC. As a result, considerable research efforts have been directed toward developing an effective IDS to counteract these threats.

### **7.1 CONCLUSION**

The proposed architecture introduces a novel approach to IDS by combining feature selection algorithms with a hybrid classifier. Specifically, a Hybridized Firefly Algorithm with Decision Tree is proposed as the feature selection method and its performance is evaluated against other optimization techniques such as Particle Swarm Optimization (PSO) and Genetic Algorithm (GA). The goal of the research is to optimize classification accuracy, thereby improving the overall effectiveness of the IDS. IDSs are essential for detecting unauthorized access and malicious behaviour in a network by continuously analyzing network traffic and system activities to identify anomalies and attacks in real time. However, developing an effective IDS poses challenges due to the evolving nature of cyber threats and the complexity of network environments. The proposed feature selection algorithm

draws inspiration from the flashing behaviour of the fireflies and aiming to identify the optimal subset of features that maximize classification accuracy.

A hybrid classifier that combines Neural Network (NN) with Decision Tree (DT) is used to enhance the detection capabilities of an IDS. The hybrid classifiers are known for their robustness and improved accuracy, as they leverage the diverse strengths of individual classifiers. The combination of proposed feature selection and a hybrid classifier for detection results in a powerful IDS capable of accurately identifying various types of attacks. The experimental results of research work demonstrate significant improvements in classification accuracy compared to the traditional approaches. By maximizing the classification accuracy, the developed IDS effectively distinguishes between normal and malicious activities, thereby reducing false positives and enhancing overall detection rates.

At 1,00,000 samples, the precision was 0.9658, outperforming, Levenberg Neural method was 0.9416, Random Forest was 0.9381, SVM was 0.9408 and KNN with 10 neighbors was 0.9453. These results indicate the proposed method's strong ability to accurately identify true positives while minimizing false positives.

At 1,00,000 samples, the accuracy was 91.95%, significantly higher than the Decision Tree was 89.98% and KNN with 10 neighbors was 74.98%, Levenberg Neural was 89.27%, Random Forest was 74.09% and SVM was 75.19%.

At 1,00,000 samples, the recall was 0.988, whereas the next best method, Levenberg Neural achieved 0.985, Random Forest was 0.808, SVM was 0.818 and KNN with 10 neighbors was 0.811. This high recall rate signifies the proposed algorithm's robust capacity to detect nearly all true attack instances, thus significantly reducing false negatives. This capability to detect a high number of

true positives is crucial in an IDS context, as missing actual threats can lead to severe security breaches.

With 1,00,000 samples, the proposed method attained an F-Measure of 0.977, surpassing other classifiers such as Levenberg Neural was 0.963, Decision Tree was 0.965, Random Forest was 0.868, SVM was 0.875 and KNN with 10 neighbors was 0.873. This high F-Measure demonstrates that the proposed method not only identifies a large number of true positives but also does so with a low rate of false positives, ensuring a balanced and effective detection capability.

Simulations were conducted in CloudSim, a robust platform for modelling and simulating cloud computing environments and services. This realistic approach allowed for testing the robustness and scalability of the IDS under various conditions. The proposed algorithm was also tested on simulated data generated within the CloudSim environment, in addition to validation over the CSE-CIC-IDS 2018 dataset. This dual-v

validation approach not only confirms the accuracy and efficiency of the modified firefly algorithm combined with the hybrid classifier in controlled datasets but also demonstrates its practical applicability in real-world, cloud-based scenarios. The simulation results underscore the algorithm's capability to effectively detect and mitigate cyber threats, providing further evidence of its potential to enhance network security across diverse computing environments.

The increasing sophistication and frequency of cyber threats demand the development of robust and effective Intrusion Detection Systems. Our proposed method, which combines the Modified Firefly Algorithm for feature selection with a hybrid classifier, offers a novel approach to improving IDS performance. By leveraging the comprehensive CSE-CIC-IDS 2018 dataset and conducting

simulations in CloudSim, we achieve significant improvements in classification accuracy, precision, recall, and overall effectiveness. This work not only advances IDS research but also provides a practical and scalable solution for protecting digital infrastructures. As cyber threats continue to evolve, ongoing research and innovation in IDS will be crucial to ensuring the security and integrity of network systems, ultimately safeguarding critical data and services from malicious attacks.

The results clearly demonstrate that the proposed feature selection algorithm combined with a hybrid classifier significantly outperforms traditional methods and other optimization algorithms. This makes it an invaluable tool in the fight against cyber threats, ensuring that IDS can effectively identify and mitigate potential attacks, thus protecting network integrity and maintaining the availability of essential services. The successful application of this method in both controlled datasets and simulated real-world environments further highlights its robustness and practical relevance in contemporary cybersecurity practices.

At 150,000 samples, the precision was 0.9658, outperforming, Levenberg Neural method was 0.9449, Random Forest was 0.9449, SVM was 0.9354 and KNN with 10 neighbors was 0.9442. These results indicate the proposed method's strong ability to accurately identify true positives while minimizing false positives.

At 150,000 samples, the accuracy was 94.99%, significantly higher than the Decision Tree was 88.29% and KNN with 10 neighbors was 82.38%, Levenberg Neural was 87.82%, Random Forest was 76.68% and SVM was 73.52%. This high accuracy vi indicates that the proposed method is exceptionally adept at correctly classifying both normal and malicious activities, providing comprehensive and reliable protection against network threats. For instance, at 20,000 samples, the accuracy was 87.98%, compared to 69.80% for the Levenberg Neural method and 70.58% for the Decision Tree.

At 150,000 samples, the recall was 0.9681, whereas the next best method, Levenberg Neural, achieved 0.9619, Random Forest was 0.9636, SVM was 0.8300 and KNN with 10 neighbors was 0.8038. This high recall rate signifies the proposed algorithm's robust capacity to detect nearly all true attack instances, thus significantly reducing false negatives. This capability to detect a high number of true positives is crucial in an IDS context, as missing actual threats can lead to severe security breaches.

With 1,50,000 samples, the proposed method attained an F-Measure of 0.9584, surpassing other classifiers such as Levenberg Neural was 0.9669, Decision Tree was 0.95295, Random Forest was 0.8838, SVM was 0.9188 and KNN with 10 neighbors was 0.8647. This high F-Measure demonstrates that the proposed method not only identifies a large number of true positives but also does so with a low rate of false positives, ensuring a balanced and effective detection capability. At 20,000 samples, the F-Measure was 0.8889, compared to 0.8417 for the Levenberg Neural method and 0.8464 for the Decision Tree.

When compared to other optimization algorithms like PSO and GA, the proposed feature selection also showed superior performance. For instance, in terms of precision, PSO combined with a hybrid classifier achieved 0.95188 at 100,000 samples, whereas the proposed method attained a higher precision of 0.9658. Similarly, recall for PSO and GA combined with a hybrid classifier at 100,000 samples was 0.83272 and 0.85425, respectively, while the proposed method achieved 0.9884, highlighting its effectiveness in identifying actual positives.

By using the comprehensive datasets like CSE-CIC-IDS 2018 and simulated dataset, the proposed architecture achieves significant improvements in the detection of attacks which contributes to the advancement of IDS research.

Simulated dataset is generated by using the cloudsim tool. CloudSim allows for the modelling and simulation of cloud computing environments and services, offering a comprehensive platform to test the robustness and scalability of the IDS under various conditions. This dual-validation approach confirms the accuracy and efficiency of the proposed architecture in controlled datasets and also demonstrates its practical applicability in real-world cloud-based scenarios. The simulation results underscore the algorithm's capability to effectively detect attacks and provides further evidence of its potential to enhance network security across diverse computing environments. This not only contributes to the advancement of IDS research but also provides a practical and scalable solution for safeguarding CC.

## 7.2 FUTURE SCOPE

### **Future Potential of the Proposed Cloud-Based IDS:**

#### **1.Utilizing AI and Machine Learning Capabilities:**

Incorporating AI and ML can significantly enhance the IDS by enabling it to detect sophisticated and previously unseen cyber threats. These technologies can facilitate real-time anomaly detection and help minimize false alerts.

#### **2.Continuous Adaptation to User Behavior:**

By implementing user behavior analytics, the IDS can evolve alongside the changing activity patterns of cloud users. This dynamic learning approach strengthens defense against both external and insider threats.

#### **3Support for Multi-Cloud and Hybrid Setups:**

With the growing trend of multi-cloud and hybrid infrastructure, future iterations of the IDS could offer unified monitoring and threat detection across diverse cloud platforms from a single interface.

#### **4.Automated Threat Response:**

Advanced versions may introduce automated actions such as isolating



compromised instances or blocking suspicious traffic, thereby shortening incident response times and containing potential damage.

**5.Assistance with Compliance Requirements:**

Enhancing the IDS to generate structured logs and incident reports will help organizations meet legal and regulatory standards like GDPR and HIPAA, ensuring transparent and accountable security practices.

**6.Improved Scalability and Efficiency:**

Future developments may focus on optimizing system performance, allowing the IDS to operate efficiently in high-demand cloud environments without compromising speed or accuracy. Another potential direction is that which creates an intrusion detection system that scales in response to the number of virtual machines in the cloud by expanding or contracting as needed.

**7.Seamless Integration with Security Ecosystems:**

Connecting the IDS with platforms like SIEM (Security Information and Event Management) and SOAR (Security Orchestration, Automation, and Response) will provide broader visibility and enable faster, more coordinated responses to security incidents.

**8.Adaptive Attack Detection:** An adaptive attack detection system can be a promising future direction in cloud security. This system helps to manage the dynamic conditions, such as changes in environmental configurations, computational resources and the locations where attack detection systems are deployed.

**9.Vulnerabilities Detection:** Another key area for future improvement is enhancing the detection of vulnerabilities through the development of a more efficient detection system

## REFERENCES

1. S. Singh, K. Saxena and Z. Khan, "Intrusion detection based on artificial intelligence techniques," *International Journal of Computer Science Trends and Technology*, vol. 2, no. 4, pp. 31-35, 2014.
2. A. Beloglazov, J. Abawajy and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755-768, 2012.
3. M. I. Alam, M. Pandey and S. S.Rautaray, "A comprehensive survey on cloud computing," *International Journal of Information Technology and Computer Science (IJITCS)*, vol. 7, no. 2, p. 68, 2015.
4. M. Kavis, "Architecting the cloud: design decisions for cloud computing service models (SaaS, PaaS, and IaaS)," *John Wiley & Sons, Inc.*, Hoboken, New Jersey, 2014.
5. S. Carlin and K. Curran, "Cloud computing security," in *Pervasive and Ubiquitous Technology Innovations for Ambient Intelligence Environments*, IGI Global, pp. 12–17, 2013.
6. A. R. Suraj, S. J. Shekar and G. S. Mamatha, "A robust security model for cloud computing applications," In *2018 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC)*, pp. 18–22, 2018.
7. P. Singh, S. Manickam and S. U. Rehman, "A survey of mitigation techniques against Economic Denial of Sustainability (EDoS) attack on cloud computing architecture," In *Proceedings of the 3rd International Conference on Reliability, Infocom Technologies and Optimization*, pp. 1–4, IEEE, 2014.
8. F. Kuang, W. Xu, and S. Zhang, "A novel hybrid KPCA and SVM with GA model for intrusion detection," *Applied Soft Computing*, vol. 18, no. 1, pp. 178–184, 2014.

9. C. Nkikabahizi, W. Cheruiyot and A. Kibe, "Classification and analysis of techniques applied in intrusion detection systems," *International Journal of Scientific Engineering and Technology*, vol. 6, no. 7, pp. 216–219, 2017.
10. V. Balamurugan and R. Saravanan, "Enhanced intrusion detection and prevention system on cloud environment using hybrid classification and OTS generation," *Cluster Computing*, vol. 22, no. 6, pp. 13027-13039, 2019.
11. P. Ghamisi and J.A. Benediktsson, "Feature selection based on hybridization of genetic algorithm and particle swarm optimization," *IEEE Geoscience and remote sensing letters*, vol. 12, no. 2, pp. 309–313, 2014.
12. A. S. Saljoughi, M. Mehrvarz and H. Mirvaziri, "Attacks and intrusion detection in cloud computing using neural networks and particle swarm optimization algorithms," *Emerging Science Journal*, vol. 1, no. 4, pp. 179–191, 2017.
13. X. S. Yang, Firey algorithm, *Nature-Inspired Metaheuristic Algorithms*, vol. 79, 2008.
14. K. Costa, C. Pereira, R. Nakamura, L. Pereira and J. Papa, "Boosting Optimum-Path Forest clustering through harmony Search and its applications for intrusion detection in computer networks," In *Proceedings of the 4th International Conference on Computational Aspects of Social Networks (CASoN)*, pp. 181–185, IEEE, 2012.
15. S Aljawarneh, M. Aldwairi and M. B. Yassein, "Anomaly based intrusion detection system through feature selection analysis and building hybrid efficient model," *Journal of Computational Science*, vol. 25, pp. 152–160, 2018.
16. C. Gong, J. Liu, Q. Zhang., H. Chen and Z. Gong, "The characteristics of cloud computing," In *2010 39th International Conference on Parallel Processing Workshops*, IEEE, pp. 275-279, 2010.
17. E. Besharati, M. Naderan and E. Namjoo, "LR-HIDS: logistic regression

- host-based intrusion detection system for cloud environments”, *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, pp. 3669-3692, 2019.
18. Y. S. Abdulsalam and M. Hedabou, "Security and privacy in cloud computing: technical review", *Future Internet*, vol. 14 no. 1, 2021.
  19. D. Rani and N. C. Kaushal, “Supervised machine learning based network intrusion detection system for Internet of Things,” in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–7, 2020.
  20. V. D. Ngo, T. C. Vuong, T. V. Luong and H. Tran, “Machine learning-based intrusion detection: feature selection versus feature extraction,” *Cluster Computing*, vol. 27, no. 3, pp. 2365-2379, 2024.
  21. O. Alomari and Z. A. Othman, “Bees algorithm for feature selection in network anomaly detection”, *Journal of applied sciences research*, vol. 8, no. 3, pp. 1748-1756, 2012.
  22. I. Ahmad, M. Hussain, A. Alghamdi and A. Alelaiwi, “Enhancing SVM performance in intrusion detection using optimal feature subset selection based on genetic principal components”, *Neural computing and applications*, vol. 24, pp. 1671-1682, 2014.
  23. M. Otair, O. T. Ibrahim, L. Abualigah, M. Altalhi, and P. Sumari, “An enhanced grey wolf optimizer based particle swarm optimizer for intrusion detection system in wireless sensor networks,” *Wireless Networks*, vol. 28, no. 2, pp. 721–744, 2022.
  24. S. K. Shandilya, B. J. Choi, A. Kumar and S. Upadhyay, “Modified Firefly Optimization Algorithm-Based IDS for Nature-Inspired Cybersecurity,” *Processes*, vol. 11, no. 3, pp. 715-731, 2023.
  25. M. A. Umar, Z. Chen, K. Shuaib and Y. Liu, “Effects of feature selection and normalization on network intrusion detection,” *Authorea Preprints*, 2024.

26. Y. K. Saheed, T. O. Kehinde, M. A. Raji and U. A. Baba, "Feature selection in intrusion detection systems: a new hybrid fusion of Bat algorithm and Residue Number System," *Journal of Information and Telecommunication*, vol. 8, no. 2, pp. 189-207, 2024.
27. M. Bakro, R. R. Kumar, M. Husain, Z. Ashraf, A. Ali, S. I. Yaqoob, M. N. Ahmed and N. Parveen, "Building a cloud-IDS by hybrid bio-inspired feature selection algorithms along with random forest model," *IEEE Access*, 2024.
28. S. Ganapathy, P. Vijayakumar, P. Yogesh and A. Kannan, "An Intelligent CRF Based Feature Selection for Effective Intrusion Detection.", *International Arab Journal of Information Technology (IAJIT)*, vol. 13, no. 1, 2016.
29. R. Patil, H. Dudeja and C. Modi, "Designing an efficient security framework for detecting intrusions in virtual network of cloud computing," *Computers & Security*, vol. 85, pp. 402–422, 2019.
30. L. Chen, M. Xian, J. Liu and H. Wang, "Intrusion Detection System in Cloud Computing Environment," *International Conference on Computer Communication and Network Security (CCNS)*, pp. 131-135, 2020.
31. Y. Liu and R. Ma, "Network anomaly detection based on BQPSO-BN algorithm," *IETE Journal of Research*, vol. 59, no. 4, pp. 334–342, 2013.
32. K. Wang, C.-Y. Huang, L.-Y. Tsai and Y.-D. Lin, "Behavior based botnet detection in parallel," *Security and Communication Networks*, vol. 7, no. 11, pp. 1849–1859, 2014.
33. J. Hussain, S. Lalmuanawma and L. Chhakchhuak, "A two stage hybrid classification technique for network intrusion detection system," *International Journal of Computational Intelligence Systems*, vol. 9, no. 5, pp. 863–875, 2016.

34. H. H. Pajouh, G. Dastghaibiyfard and S. Hashemi, "Two-tier network anomaly detection model: a machine learning approach," *Journal of Intelligent Information Systems*, vol. 48, no. 1, pp. 61–74, 2017
35. R. Kesavamoorthy and K. R. Soundar, "Swarm intelligence based autonomous DDoS attack detection and defense using multi agent system," *Cluster Computing*, pp. 1–8, 2019.
36. P. Ghosh, A. Karmakar, J. Sharma and S. Phadikar, "CS-PSO based intrusion detection system in cloud environment," *In Emerging Technologies in Data Mining and Information Security*, pp. 261–269, Springer, Berlin, Germany, 2019.
37. D. J. Prathyusha and G. Kannayaram, "A cognitive mechanism for mitigating DDoS attacks using the artificial immune system in a cloud environment," *Evolutionary Intelligence*, vol. 4, pp.1-2 2020.
38. G. S. Kushwah and V. Ranga, "Voting extreme learning machine based distributed denial of service attack detection in cloud computing," *Journal of Information Security and Applications*, vol. 53, 2020.
39. KB Virupakshar, M Asundi, K Channal, P Shettar, S Patil and DG Narayan, "Distributed denial of service (DDoS) attacks detection system for Open Stack-based private cloud," *Procedia Computer Science*, pp. 2297-2307, 2020.
40. S Rajagopal, PP Kundapur and KS Hareesha, "Towards Effective Network Intrusion Detection: From Concept to Creation on Azure Cloud," *IEEE Access*, pp. 19723-19742, 2021.
41. E.Arul and A. Punidha, "Supervised Deep Learning Vector Quantization to Detect Mem Cached DDoS Malware Attack on Cloud", *SN Computer Science*, vol. 2, no. 2, pp. 1-2, 2021.
42. G. Sreelatha, A. V. Babu, and D. Midhunchakkaravarthy, "Improved security in cloud using sandpiper and extended equilibrium deep transfer learning based intrusion detection," *Cluster Computing*, vol. 25, no. 5, pp.

3129–3144, 2022.

43. H. Ghani, B. Virdee, and S. Salekzamankhani, “A Deep Learning Approach for Network Intrusion Detection Using a Small Features Vector,” *Journal of Cybersecurity and Privacy* 2023, Vol. 3, Pages 451-463, vol. 3, no. 3, pp. 451–463, 2023.
44. Z. Long, H. Yan, G. Shen, X. Zhang, H. He and L. Cheng, “A Transformer-based network intrusion detection approach for cloud security”, *Journal of Cloud Computing*, vol. 13, no. 1, 2024.
45. A. V. Songa and R. K. Ganesh, "An integrated SDN framework for early detection of DDoS attacks in cloud computing," *Journal of Cloud Computing*, vol. 13, no. 1, 2024.
46. R. R. Dewangan, S. Soni and A. Mishal, “An approach of privacy preservation and data security in cloud computing for secured data sharing,” *Recent Advances in Electrical & Electronic Engineering*, vol. 18, no. 2, pp. 176–195, Feb. 2025.
47. K. V. K. Chithanya and L. Reddy, “Automatic intrusion detection model with secure data storage on cloud using adaptive cyclic shift transposition with enhanced ANFIS classifier,” *Cyber Security and Applications*, vol.3, 2025.
48. Park, H., EL Azzaoui, A., & Park, J, “AIDS-Based Cyber Threat Detection Framework for Secure Cloud-Native Microservices, “ *Electronics*, vol. 14, no. 2, 229, 2025
49. V. R. Kebande and H. S. Venter, “A cognitive approach for botnet detection using Artificial Immune System in the cloud,” In *2014 Third International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec)*, IEEE, pp. 52-57, 2014.
50. G. Somani, M. S. Gaur, D. Sanghi, M. Conti and R. Buyya, “DDoS attacks in cloud computing: Issues, taxonomy, and future directions,” *Computer communications*, vol. 107, pp. 30-48, 2017.

51. A. M. Al-Dulaimi, O. M. Abdulqader A. F. Zakharzhevskiy, "Threats in Cloud Computing System and Security Enhancement," In *2024 35th Conference of Open Innovations Association (FRUCT)*, IEEE, pp. 82-93, 2024.
52. N. Cao, C. Wang, M. Li, and K. Ren, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on Cloud Computing*, vol. 2, no. 1, pp. 29-42, Jan. 2013.
53. Q. Wang, M. He, M. Du, and S. S. M. Chow, "Searchable encryption over feature-rich data," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2836-2849, Dec. 2016.
54. V. Popic and S. Batzoglou, "A hybrid cloud read aligner based on MinHash and kmer voting that preserves privacy," *Nature Communications*, vol. 8, no. 1, pp. 47–59, 2017.
55. W. Liang, K. C. Li, J. Long, and X. Kui, "An industrial network intrusion detection algorithm based on multifeature data clustering optimization model," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 49-61, Jan. 2019.
56. Y. Miao, Y. Tang, and B. A. Alzahrani, "Airborne LiDAR assisted obstacle recognition and intrusion detection towards unmanned aerial vehicle: Architecture, modeling and evaluation," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 10873-10886, Oct. 2020.
57. A. Alshammari and A. Aldribi, "Apply machine learning techniques to detect malicious network traffic in cloud computing," *Journal of Big Data*, vol. 8, no. 1, pp. 21–39, 2021.
58. T. Li, D. Wu, J. Wang, Y. Zhao, and H. Han, "A systematic literature review of methods and datasets for anomaly-based network intrusion detection," *Computers & Security*, vol. 2022, pp. 13–35, 2022.



59. S. Songma, T. Sathuphan and T. Pamutha, "Optimizing Intrusion Detection Systems in Three Phases on the CSE-CIC-IDS-2018 Dataset", *Computers*, vol. 12, no.12, pp. 245-260 pages, 2023.
60. B. L. Farhan and A. D. Jasim, "Performance analysis of intrusion detection for deep learning model based on CSE-CIC-IDS2018 dataset", *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 26, no. 2, pp. 1165-1172, 2022.
61. I. Fister, I. Fister Jr, X. S. Yang and J. Brest, "A comprehensive review of firefly algorithms", *Swarm and evolutionary computation*, vol.13, pp.34-46, 2013.
62. R. Buyya, R., Ranjan and R. N. Calheiros, R. N., "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities", *In 2009 international conference on high performance computing & simulation, IEEE*, pp. 1-11, 2009.
63. R. N. Calheiros, R. Ranjan, A. Beloglazov, CAF R. De, R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", *Software: Practice and experience*, vol. 41, no.1, pp. 23-50, 2011.
64. A. L. Shalabi, Z. Shaaban and B. Kasasbeh, "Data Mining: A Preprocessing Engine", *J. Comput. Sci.*, vol.,2, pp. 735-739, 2006.
65. T. Goyal, A. Singh and A. Agrawal, "Cloudsim: simulator for cloud computing infrastructure and modelling", *Procedia Engineering*, vol.38, pp. 3566-3572, 2012.
66. D. M. reddy, M. S. Kalyani, P. Hemalatha, A. Sreeja, & M. Y. Devi, "Cloud Computing and Security using CloudSim", *In 2023 3rd International Conference on Smart Data Intelligence (ICSMDI)*, IEEE, pp. 187-194, 2023.
67. T. Stevens, "Cyber security and the politics of time", Cambridge University Press, 2016.

68. N. J. Miller and M. Aliasgari, "Benchmarks for evaluating anomaly-based intrusion detection solutions", California State University, Long Beach, 2018.
69. N. Caife, H. Carter, P. Traynor and K. R. Butler, "Cryptolock (and drop it): stopping ransomware attacks on user data", In 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), IEEE, pp. 303–312, 2016.
70. R. Koch, "Towards Next Generation Intrusion Detection", In 2011 3<sup>rd</sup> International Conference on Cyber Conflict, IEEE, pp. 1–18, 2011.
71. J. Rajahalme, A. Conta, B. Carpenter, S. Deering, "RFC 3697: IPv6 Flow Label Specification", In: The Internet Society, 2004.
72. I. Sharafaldin, A. H. Lashkari and A. A. Ghorbani, "Toward Generating A New Intrusion Detection Dataset and Intrusion Traffic Characterization", In: ICISSP, pp. 108–116, 2018.
73. CSE-CIC-IDS2018 on AWS. Canadian Institute for Cybersecurity. Available online: <https://www.unb.ca/cic/datasets/ids-2018.html> (accessed on 2 February 2022).
74. H. S. Gebremedhen, D. E. Woldemichael and F. M. Hashim, "A firefly algorithm based hybrid method for structural topology optimization", *Advanced Modeling and Simulation in Engineering Sciences.*, vol. 7, no. 1, pp. 1–20, 2020.
75. I. Fister, Jr. I. Fister, X.S. Yang and J. Brest, "A comprehensive review of firefly algorithms", *Swarm and evolutionary computation*, vol. 13, pp. 34–46, 2013.
76. K. Peng, V. C. M. Leung, L. Zheng, S. Wang, C. Huang, and T. Lin, "Intrusion detection system based on decision tree over big data in fog environment", *Wireless Communications and Mobile Computing*, vol. 2018.

77. C. Modi, D. Patel, B. Borisanya, A. Patel, and M. Rajarajan, "A Novel Framework for Intrusion Detection in Cloud," in *Proceedings of the Fifth International Conference on Security of Information and Networks*, pp. 67–74, 2012.
78. J. Wei, C. Long, J. Li and J. Zhao, "An intrusion detection algorithm based on bag representation with ensemble support vector machine in cloud computing", *Concurrency and Computation: Practice and Experience*, vol. 32, no. 24, pp. 14,2020.
79. Q. Schueller, K. Basu, M. Younas, M. Patel and F. Ball, "A hierarchical intrusion detection system using support vector machine for sdn network in cloud data center", in *Proceedings of the 28th International Telecommunication Networks and Applications Conference (ITNAC)*, pp. 6,2018.
80. B. Sundararaman, S. Jagdev, and N. Khatri, "Transformative role of arti5cial intelligence in advancing sustainable tomato (*Solanum lycopersicum*) disease management for global food security: a comprehensive review", *Sustainability*, vol. 15, no. 15, Article ID 11681, 2023.
81. G. Dhanush, N. Khatri, S. Kumar, and P. K. Shukla, "A comprehensive review of machine vision systems and anti- social intelligence algorithms for the detection and harvesting of agricultural produce", *Scientific African*, vol. 21, 2023.

## LIST OF PUBLICATIONS

1.P.Rana and I. Batra, “Detection of attacks in cloud computing environment—a comprehensive review,” In *2021 2nd International Conference on Intelligent Engineering and Management (ICIEM)*, IEEE, pp. 496-499, 2021.

Weblink: <https://ieeexplore.ieee.org/abstract/document/9445284/>

2.P. Rana, I. Batra, A. Malik, A. L. Imoize , Y. Kim, S. K.Pani, N. Goyal, A. Kumar and S. Rho, “Intrusion Detection Systems in Cloud Computing Paradigm: Analysis and Overview,” *Complexity*, vol.1, 2022.

Weblink: <https://onlinelibrary.wiley.com/doi/full/10.1155/2022/3999039>

3.P. Rana, Dr. I. Batra and Dr A. Malik, "An Innovative Approach to Select Features for Identifying Attacks in Cloud Computing Environments by a New Algorithm that Integrates the Principles of the Firefly Algorithm and K-Nearest Neighbour," In *Intelligent Circuits and Systems for SDG 3–Good Health and well-being*, pp. 211-217, CRC Press.

Weblink:<https://www.taylorfrancis.com/chapters/edit/10.1201/9781003521716-22/innovative-approach-select-features-identifying-attacks-cloud-computing-environments-new-algorithm-integrates-principles-firefly-algorithm-nearest-neighbour-pooja-rana-isha-batra-arun-malik>

4. P. Rana, I. Batra, A. Malik, I. H. Ra, O. S. Lee and A. S. Hosen, “Efficacious Novel Intrusion Detection System for Cloud Computing Environment,” *IEEE Access*, pp. 99223-99239, 2024.

Weblink:<https://ieeexplore.ieee.org/abstract/document/10587216>

5. P. Rana, I. Batra and A. Malik, “An Innovative Approach: Hybrid Firefly Algorithm for Optimal Feature Selection,” In *2024 International Conference on*

*Electrical Electronics and Computing Technologies (ICEECT)* ,vol. 1, pp. 1-4,  
IEEE.

Weblink: <https://ieeexplore.ieee.org/abstract/document/10739012>

***Graphical Abstract submitted as Copyright:***

Pooja Rana and Isha Batra, “An Innovative Architecture for Detection of Attacks in Cloud Computing Environment”.