A Hybrid Anomaly Characterisation and Sensitivity Analysis Framework for Software-Defined Network-Aided Communication

Chekwube EZECHI
LCU/PG/003043

Being a PhD Thesis Submitted to the Department of Computer Science, Faculty of Natural and Applied Sciences, Lead City University, Ibadan, Oyo State, Nigeria

In Partial Fulfillment of the Requirement for the Award of Doctor of Philosophy Degree (PhD) in Computer Science

2025

Lead City University DO NOT COPY

Certification

This is to certify that Chekwube EZECHI with matriculation number LCU/PG/003043 carried out this research work titled "A Hybrid Anomaly Characterisation and Sensitivity Analysis Framework for Software-Defined Network-Aided Communication" in the Department of Computer Science, Faculty of Applied Sciences, Lead City University, Ibadan, Oyo State, for the award of Doctor of Philosophy Degree (PhD) and that this has not been previously submitted.

Lead City University DO NOT COPY

_____ _____

Dr. Wilson Sakpere Date

Supervisor

_____ _____

Dr. Victoria B. Oyekunle Date

Head of Department

Dedication

I dedicate this thesis to the Almighty God, in gratitude for the gift of life and His boundless mercies.

Lead City University DO NOT COPY

Acknowledgement

I wish to express my sincere gratitude to Lead City University for granting me the opportunity to pursue and complete my PhD programme. I also appreciate the University Library for providing access to valuable research resources that greatly supported this study.

My deepest thanks go to the Department of Computer Science for offering an inspiring and supportive academic environment. I am profoundly grateful to my supervisors, Dr. W. Sakpere, Dr. M. O. Akinsolu, and Dr. Abimbola Sangodoyin, for their exceptional guidance, patience, and scholarly mentorship throughout this research. Their constructive insights have been instrumental to my academic growth. I extend heartfelt appreciation to the Departmental Postgraduate Coordinator, Dr. A. A. Waheed, for his patience and steadfast support. I am also grateful to my lecturers Prof. S. O. Akinola, Prof. P. A. Idowu, Dr. M. O. Agbaje, Dr. W. Ajayi, and other departmental staffs for their commitment and contribution to my academic journey.

My special thanks go to my family for their love and sacrifices: my father, Mr. Ray Ezechi; my mother, Mrs. Stella Ezechi; my sisters, Mrs. Chika Onyemelukwe (née Ezechi) and Ms. Ekene Ezechi; my uncle, Prof. O. C. Ezechi, and his wife, Dr. L. O. Ezechi. Their encouragement and prayers have been a constant source of strength. I also appreciate Dr. F. T. Akinsolu for his continued mentorship and support. Finally, I am thankful to my colleagues for their collaboration and shared experiences throughout this PhD journey.

While I am indebted to all who contributed to this work, I take full responsibility for any errors or shortcomings that may remain.

Lead City University DO NOT COPY

Abstract

Contemporary communication infrastructures increasingly rely on Software-Defined Networking (SDN) to deliver programmability, scalability, and dynamic resource management. These strengths also introduce vulnerabilities, exposing SDN controllers and data planes to anomalies such as Distributed Denial of Service (DDoS) flooding attacks. These attacks threaten availability, degrade performance, and undermine network resilience. This research develops a hybrid anomaly characterisation and sensitivity analysis framework tailored for SDN-enabled Internet of Things (IoT) environments. The DOE-GAN-SA framework integrates design of experiments (DOE), generative adversarial networks (GANs), anomaly detection and characterisation (ADC), and sensitivity analysis (SA), providing a unified methodology for analysing and mitigating anomalous behaviour. The study simulated normal and malicious traffic within an emulated SDN-IoT testbed built with Mininet, Oracle VirtualBox, and Ubuntu Linux. Popular DDoS vectors, including HTTP, TCP, and UDP flooding attacks, were launched with the Low Orbit Ion Cannon (LOIC) tool against custom topologies containing OpenFlow-enabled switches and hosts. Performance indicators such as throughput, jitter, and response time were collected. Latin Hypercube Sampling (LHS) and GANs were used to augment datasets and ensure statistical robustness. CART supported anomaly detection and classification, while Artificial Neural Networks (ANNs) performed global sensitivity analysis. Repeated experiments and hypothesis tests ensured statistical credibility. Results demonstrated that the DOE-GAN-SA framework effectively captured service degradations across both normal and DDoS conditions. The hybrid model achieved 99.93 % accuracy and an F1 score of 1.000 for anomaly detection. Sensitivity analysis produced a mean squared error of 9.14×10^{-4}, confirming that the ANN reliably learned the behaviour of throughput, jitter, and response time under multiple scenarios. Compared with existing methods, the framework delivered higher detection accuracy, improved dataset quality through synthetic augmentation, and reduced computational overhead, supporting its suitability for near real-time SDN-IoT deployment. Key contributions include a unified methodology for anomaly characterisation and sensitivity analysis in SDN, empirical validation of synthetic data augmentation for security studies, and identification of parameters that most influence network resilience. The study offers network administrators and security practitioners proactive tools for anomaly detection and performance optimisation, strengthening the development of resilient and secure SDN-IoT infrastructures. Future work will expand the framework to additional attack types and emerging 5G contexts.

Keywords: Software-Defined Networking (SDN), Internet of Things (IoT), Distributed Denial of Service (DDoS), Generative Adversarial Networks (GANs), Design of Experiments (DOE), Latin Hypercube Sampling (LHS), Artificial Neural Networks (ANNs)

Word Count: 355

Leadcity University DO NOT COPY

Table of Contents

Content		Page
Title Page		
Certification		ii
Dedication		iii
Acknowledgement		iv
Abstract		v
Table of Contents		vi
List of Tables		x
List of Figures		xii
Chapter One: Introduction		
1.1	Background to the Study	1
1.2	Statement of the Problem	5
1.3	Aim and Objectives of Study	6
1.4	Research Questions	6

Lead City University DO NOT COPY

1.5 Significance of the Study 7

1.6 Scope of the Study 7

1.7 Limitation of the Study 8

1.8 Operational Definition of Terms 8

Endnotes 11

Chapter Two: Literature Review

2.1 Conceptual Review 14

2.1.1 Open Flow 17

2.1.2 Traditional Networks and SDN 18

2.1.3 SDN Architecture 24

2.1.3.1 The Essential Features of SDN 27

2.1.3.2 Development Platforms for SDN 28

2.1.4 Vulnerabilities and Security Issues in SDN 28

2.1.4.1 Security Vulnerabilities in SDN by Architectural Plane 29

2.1.4.2 Identifying Security Flaws Using the STRIDE Model 31

Lead City University DO NOT COPY

2.1.4.3	Major Security Threats in the SDN Planes	35
2.1.5	Security Challenges in SDN	37
2.1.5.1	Application Plane	38
2.1.5.2	Control Plane	39
2.1.5.3	Data Plane	40
2.1.6	Security Solutions Across SDN Planes Based on ITU-T Recommendations	41
2.1.7	Anomaly Detection in SDN	43
2.1.7.1	Types of Anomalies in SDN	44
2.1.7.1.1	Denial of Service and Distributed Denial of Service (DoS/DDoS)	44
2.1.7.1.2	Traffic Anomalies	45
2.1.7.1.3	Topology Attacks	46
2.1.7.1.4	Control-Plane and Flow Table Attacks	46
2.1.7.2	Existing Detection Methods	46
2.1.7.2.1	Rule-Based Approaches	46

Lead City University DO NOT COPY

2.1.7.2.2.	Statistical Approaches	47
2.1.7.2.3	Machine Learning Approaches	47
2.1.7.2.4	Deep Learning Approaches	48
2.1.7.2.5	GAN-Based Detection	49
2.1.8	Machine Learning and GANs for Synthetic Data Generation	49
2.1.8.1	Machine Learning for Synthetic Data Generation	50
2.1.8.1.1	Programmatic and Rule-Based Approaches	50
2.1.8.1.2	Random Sampling and Resampling	50
2.1.8.1.3	Statistical and Probabilistic Models	50
2.1.8.1.4	Differential Privacy–Based Synthesis	51
2.1.8.1.5	Machine Learning Based Approaches	51
2.1.9	GAN Architectures for Synthetic Data Generation	52
2.1.9.1	Standard GANs	52
2.1.9.2	Deep Convolutional GANs (DCGANs)	53
2.1.9.3	Conditional GANs (cGANs)	53

Lead City University DO NOT COPY

2.1.9.4	Tabular GANs	53
2.1.9.5	BiGANs and f-AnoGANs	54
2.1.9.6	GANs with Privacy-Preserving Extensions	54
2.1.10	Applications of Synthetic Data in Security and SDN	54
2.1.11	Sensitivity Analysis of Detection Parameters in SDN	56
2.1.11.1	Local Sensitivity Analysis (LSA)	59
2.1.11.2	Global Sensitivity Analysis (GSA)	60
2.1.11.3	Artificial Intelligence-Driven Sensitivity Analysis in SDN	62
2.1.11.3.1	Machine Learning-Assisted Local Sensitivity Analysis in SDN	63
2.1.11.3.2	Machine Learning-Assisted Global Sensitivity Analysis in SDN	64
2.2	Methodological Review	66
2.3	Related Works	71
2.4	Summary of Gaps in Related Works	75
	Endnotes	76

Chapter Three: Methodology

Lead City University DO NOT COPY

3.1	Introduction	88
3.2.	SDN Architecture	88
3.3.	IoT Network Topology	89
3.4.	Simulation of Traffic Scenarios on the SDN-reliant IoT Network Model	92
3.5.	Performance Metrics of the SDN-reliant IoT Network Model	93
3.6.	Basic Techniques	94
3.6.1.	Design of Experiments (DOE)	95
3.6.2.	Latin Hypercube Sampling (LHS)	95
3.6.3.	Machine Learning (ML)	97
3.6.3.1	Neural Networks	98
3.6.3.2.	Artificial Neural Network (ANN)	98
3.6.3.3.	Generative Adversarial Networks (GANs)	101
3.6.3.4.	Classification and Regression Tree (CART)	103
	Endnotes	119

Chapter Four: Results and Discussions

Lead City University DO NOT COPY

4.1	Validation of Generated Synthetic Data	120
4.1.1.	Hypothesis Testing and Comparisons for the Validation of Generated Datasets	132
4.2.	ML-Driven SA	139
4.2.1.	Hypothesis Testing for Sensitivity Analysis	147
4.3	Comparisons with Other Approaches	148
4.4	Recommendations	149
	Endnotes	152

Chapter Five: Conclusion

5.1	Summary	154
5.2	Contribution to Knowledge	155
5.3	Area of Further Research	159
Bibliography		160
Appendixes		175
Bio-data		190
The University Compliance Certification		195

Lead City University DO NOT COPY

Lead City University DO NOT COPY

List of Tables

Table	Title	Page
2.1	Comparison of Traditional Networks and SDN Architectures	21
2.2.	Comparison of Security in Traditional Networks vs SDN	23
2.3	Overview of SDN Development Platforms	28
2.4	Security Vulnerabilities in SDN by Architectural Plane	30
2.5	Microsoft STRIDE attack types and security properties	33
2.6	SDN Security Solutions Based on ITU-T Security Recommendations	42
2.7	DDoS Attacks on layers of SDN Architecture	45
3.1.	Table of Hyperparameters	106
4.1.	Descriptive statistics of metrics over 900 observations (normal network instance)	121
4.2.	Descriptive statistics of metrics over 900 observations (TCP DDoS attack network instance)	122
4.3.	Descriptive statistics of metrics over 900 observations (UDP DDoS attack network instance)	123
4.4.	Descriptive statistics of metrics over 900 observations	

Lead City University DO NOT COPY

(HTTP DDoS attack network instance). 124

4.5. Descriptive statistics of PA over 50 independent runs 135

4.6. Descriptive statistics of F1 over 50 independent runs 138

4.7. Hypothesis test: IoT_{DB} vs. the combined datasets of IoT_{LHS} and IoT_{GAN}. 139

4.8. Descriptive statistics of the MSE values of the ANN models built over 50 independent runs 145

4.9. Hypothesis test: Other datasets vs IoT_{AUG}^{norm} (MSE_{norm}) 148

4.10. Comparative Assessment 149

Lead City University DO NOT COPY

List of Figures

Figure	Title	Page
2.1	Processing at a switch while receiving a new packet	17
2.2	OpenFlow enabled SDN device	18
2.3	Comparison of network device in IP network and SDN	20
2.4	Typical SDN architecture	25
2.5	CIA Triad	32
2.6	Illustration of a typical sensitivity analysis implementation	57
3.1.	Typical SDN architecture	88
3.2.	Illustration of the modeled SDN-reliant IoT network architecture (a typification)	90
3.3.	Typification of DDoS attack strategy	92
3.4	Flow diagram of the DOE-GAN-SA framework	106
4.1.	Box plots illustrating the distributions of original and synthetic throughput metrics across all scenarios in the modelled SDN-reliant IoT network.	125

Lead City University DO NOT COPY

4.2. Box plots illustrating the distributions of original and synthetic jitter metrics across all scenarios in the modelled SDN-reliant IoT network 127

4.3. Box plots illustrating the distributions of original and synthetic response time metrics across all scenarios in the modelled SDN-reliant IoT network 129

4.4. Parallel coordinates plots of simulated, LHS-based, and GAN-generated T_p,, J_t and $R_t$metrics across all network scenarios 130

4.5. Confusion matrices for the CART models constructed 133

4.6. Typical ROC curves for the CART models constructed 135

4.7. C_F^w plot against IoT network scenarios 139

4.8. Layout of the ANN models that was built 139

4.9. Convergence trend of the MSE values for the ANN models constructed using IoT_{AUG}^{norm} (the best validation performance is 4.2194 $\times 10^{-6}$ at epoch 240) 140

4.10. Gradient information trend for the ANN models constructed using IoT_{AUG}^{norm} 141

4.11. Error histogram (with 20 bins) for the ANN models

Lead City University DO NOT COPY

constructed using IoT_{AUG}^{norm}. 141

4.12. Plots of regression for ANN models constructed using IoT_{AUG}^{norm} 142

Lead City University DO NOT COPY

Chapter One

Introduction

1.1 Background to the Study

The use of Internet-based services and applications has grown significantly during the past 20 years. Approximately 66.2% of the world's population currently uses the Internet. Internet users have grown by 1.8 per cent over the past year, with 97 million new users coming online for the first time during 2023[1].

As a result, concerns about Internet security have significantly increased. There have always been many security risks to the Internet. Common anomalies on the Internet include Trojan horses, port scans, worms, and denial of service attacks, among others. Denial of service (DoS) attacks have drawn the attention of numerous researchers. A DoS attack is an intentional effort to prevent legitimate users from using specific network resources. Mafiaboy, a 15-year-old hacker, began launching denial-of-service attacks against a number of e-commerce websites, including Amazon and eBay[2], on February 7, 2000. Following then, further developments occurred in the strategy of a denial-of-service attacks.

Attackers started enlisting geographically dispersed groups of devices to perform Distributed Denial of Service (DDoS) attacks. When a DDoS attack occurs, the hacker exploits network weaknesses to introduce a harmful program known as a Trojan Horse into computer systems without the users' knowledge. Through the replication of this malicious program across numerous interconnected devices, hackers build an army of compromised computer systems that they can use to launch DDoS attacks. These infected computers are frequently referred to as bots, and the

Lead City University DO NOT COPY

collective term for these machines is known as botnet[3]. The bot-master[3,4,5] is a human operator who controls the entire botnet remotely. When a DDoS assault is initiated, the attacker instructs all of the compromised computers, which then begin to send pointless traffic in the victim's direction. The number of infected devices which can reach millions these days will determine how many pointless traffic packets the victim is likely to receive. As a result, the victim is considered to be under a DDoS attack since genuine users are unable to access the victim's resources.

DDoS can also be used as a powerful weapon when there is a conflict between two groups or two individuals by obstructing an opponent's applications and infrastructure; a person may intentionally become an attacker and carry out unwanted activities in response to a perceived injustice through this attack; and through cyber warfare (which is motivated by politics or geopolitics), a terrorist cell attempts to attack some of the sensitive zones to destroy the economic system. In the Russia–Ukraine conflict, Russia launched DDoS attacks against multiple military, government, and financial websites in Ukraine. These attacks caused several critical infrastructures and important network systems to collapse, significantly impacting Ukraine's social order. DoS and DDoS attacks have become a significant method of cyber warfare[6]. Therefore, effectively preventing and mitigating network DDoS attacks has become an urgent problem that needs to be addressed.

Rapid advancements in IT infrastructure have led to a significant growth in network size and complexity.This growth is propelled by the increasing emphasis on seamless and distributed connectivity, cloud-based applications, and real-time network monitoring and automation[7]. Recent statistics indicate that the number of connected devices is on a trajectory to reach several billion by 2025[8]. However, alongside these advancements in IT infrastructure, the size and complexity of

Lead City University DO NOT COPY

networks have grown significantly, presenting challenges in ensuring essential properties such as confidentiality, integrity, authentication, non-repudiation and availability of information[9].

The creation of more resilient, scalable, and secure networks has become the focus of industry and researchers in recent years[4]. To create a better network in the future, they implemented technologies like Named Data Networking (NDN)[10], programmable networks[11], Software-Defined Networking (SDN)[12], and HTTP as the narrow waist[13]. SDN is regarded as the best option among them for addressing the previously mentioned issues with the existing network. A network architecture called SDN offers hope for effective network infrastructure. First, it eliminates vertical integration by separating the data plane (the underlying routers and switches that transfer network traffic in accordance with network control logic) from the control plane, which is the network's control logic. Second, this separation makes the network control logic installed in logically centralised controllers and network switches into straightforward packet-forwarding devices that simplify network management and add flexibility, speed, and programmability. In contrast to traditional networks' static and dispersed environment, SDN is a step towards the creation of a dynamic and centralised network. The networks that are currently widely used are complex and challenging to administer[14]. In conventional IP-based networks, network managers must use vendor-specific commands[4] to configure each network device in order to apply high-level network policies. In the present IP-based networks, this makes it extremely difficult to enforce necessary regulations and reconfigure network devices. Vertical integration is another feature of these networks' rigidity. Tight coupling between the control plane, which determines how to handle network traffic, and the data plane, which implements traffic forwarding in accordance with the

Lead City University DO NOT COPY

control plane's decision, limits the network's flexibility and inhibits networking infrastructure innovation.

SDN is an approach aimed at addressing the complexity as they are implemented to mainly decouple control management and data forwarding in present-day contemporary data networks such as IoT and IIoT data networks[15,16]. It supports the flexibility, programmability, and openness of many data networks today[16], while also considering the cost, and robustness of the network to meet customers' growing demands. This distinct feature of SDNs has led to many state-of-the-art IoT network architectures such as 5G and beyond network architectures being SDN-based[17].

As a fast-growing technology, SDN simplifies network administration, management, and monitoring by using the OpenFlow protocol[14]. With the help of the controller, SDN equips network administrators with a global view of the network and supports the interactive programming of the underlying data plane elements, simultaneously. Hence, the SDN controller is the intelligence of the network that exercises direct control over data plane devices through the application programming interface (API)[14]. Despite the programmability, flexibility, and decentralised control of SDNs, which are very critical to their successful deployment, these features tend to be at odds with making SDNs more secure[18]. It has its concerns and challenges in the form of network security, scalability and supportability. Among all these problems, security is at the forefront. Since the centralised controller is responsible for the management of the network, the failure of this controller would hamper the whole network.

Consequently, many SDN architectures still suffer from several security breaches as the centralised control and communication between controllers and switches might be the target which include

Lead City University DO NOT COPY

(but are not limited to) increased potential for DDoS flooding attacks[19]. SDN's primary advantage is centralized network control, but this is severely threatened by Distributed Denial of Service (DDoS) attacks. When a DDoS attack occurs, the SDN controller becomes overwhelmed and unable to manage normal network traffic, leading to a loss of centralized control. DDoS flooding attacks use a large number of compromised hosts to either consume network bandwidth or make target nodes or end-user devices unavailable[20]. These attacks are highly dangerous and typically involve botnets, where end users are often unaware their devices are being used for such malicious purposes.

1.2 Statement of the Problem

Contemporary communication systems are increasingly reliant on Software-Defined Networking (SDN) for their operation and management[21]. SDN enables dynamic and centralized control of network resources, making it possible to adapt networks to changing requirements efficiently. However, ensuring the reliability and resilience of SDNs in the face of anomalies, such as network failures, or unforeseen operational issues, remains a critical challenge[20,22]. Anomaly like DDoS flooding attacks in SDNs can lead to network downtime, data breaches, and disruptions, making it imperative to develop advanced methodologies to address these vulnerabilities[23,24].

DDoS flooding attacks lack a unified rationale and can be driven by competition, political motives, or monetary incentives. The tools for executing these attacks, such as HULK and HOIC, are sophisticated and readily available, making the need for advanced defense mechanisms critical. Many solutions have been proposed to detect and mitigate DDoS flooding attacks, including those targeting HTTP, TCP, and UDP flooding[25]. However, open issues still exist. Given the anticipated rise in DDoS attacks with the transition to 5G networks, there is an urgent need for a DDoS attack

Lead City University DO NOT COPY

detection framework that can quickly learn and adapt to evolving attack patterns while transitioning smoothly from current networks to 5G and beyond. This framework should aim to reduce computational time and increase detection accuracy.

1.3 Aim and Objectives of the Study

This research aims to develop a robust a hybrid framework for anomaly characterization and sensitivity analysis in contemporary software-defined network aided communication systems. The research objectives include:

i. Design a model that can provide the characterisation of irregularities in SDN specifically DDoS flooding attacks
ii. Conduct sensitivity analysis on SDN performance metrics under normal SDN operations and when the SDN is subjected to popular variants of DDoS flooding attacks
iii. Integrate anomaly characterization with sensitivity analysis into a hybrid framework to enable simultaneous detection and evaluation of SDN vulnerabilities under DDoS flooding attack scenarios.
iv. Validate the hybrid model's performance using metrics such as accuracy, f1 score.
v. Benchmark the attained result with existing models.

1.4 Research Questions

The research questions that this research tends to answer are highlighted below:

1. How should the characterisation of irregularities in SDN be done?

Lead City University DO NOT COPY

2. Why is the sensitivity analysis of SDN performance metrics important for network management and security?

3. How can anomaly characterisation and sensitivity analysis be effectively integrated into a hybrid framework to enable simultaneous detection of irregularities and evaluation of vulnerabilities in SDN under diverse DDoS flooding attack scenarios?

4. How should the sensitivity of SDN be assessed based on current challenges?

5. How does the proposed model's performance compare to existing approaches in SDN environments?

1.5 Significance of the Study

The study will help in the security of SDN-reliant IoT network architecture by introducing a framework which enhances network performance management and security. The framework enables proactive measures against potential disruptions, ensuring robust and resilient networks. It equips network administrators and security professionals with the tools needed to identify and mitigate potential issues, ultimately bolstering the reliability and efficiency of SDN deployments. This research not only contributes to the broader understanding of anomaly detection and mitigation strategies in complex networked systems but also paves the way for more stable, adaptive, and secure communication infrastructures critical for modern digital ecosystems.

1.6 Scope of the Study

This study focuses on how to efficiently identify DDoS assaults in SDN using statistical and machine learning methods. The open-source Low Orbit Ion Canon (LOIC) attack tool is used to launch DDoS flooding assaults on the network to examine the vulnerabilities of the SDN network

Legacy University DO NOT COPY

using Kali Linux. Oracle Virtual Box and Linux operating system (Ubuntu) were used to conduct all simulations in an isolated network, and Mininet was used to mimic custom network topologies. The fact that every DDoS flooding assault was planned from an internal network should be noted. As a result, DDoS attacks brought about by external networks are not addressed and fall outside of the scope of this work.

1.7 Limitation of Study

The study on the characterization of anomalies and global sensitivity analysis in Software-Defined Networks (SDNs) is confined to examining Distributed Denial of Service (DDoS) flooding attacks within a specified domain. This limitation narrows the focus to comprehensively understanding the behaviour of SDNs under the relentless assault of DDoS flooding, shedding light on how anomalies manifest and propagate within this context.

1.8 Operational Definition of Terms

DoS/DDoS (Denial of Service / Distributed Denial of Service): A type of cyberattack where an adversary floods a network, server, or service with excessive traffic, rendering it unavailable to legitimate users. While DoS originates from a single source, DDoS leverages multiple distributed systems.

SDN (Software-Defined Networking): A networking paradigm that separates the control plane from the data plane, enabling centralized programmability, flexibility, and dynamic network management through software applications.

Lead City University DO NOT COPY

OpenFlow: A standardized communication protocol that allows the SDN controller to directly interact with the forwarding plane of network devices, enabling flow-level management and configuration.

API (Application Programming Interface): A defined set of rules and protocols that allows different software applications to communicate and interact, often used in SDN to enable applications to control network behavior programmatically.

HTTP (Hypertext Transfer Protocol): An application-layer protocol for transmitting hypermedia documents such as HTML, widely used for communication between web browsers and servers. It is a frequent target in application-layer DoS/DDoS attacks.

TCP (Transmission Control Protocol): A reliable, connection-oriented transport layer protocol that ensures ordered and error-checked delivery of data packets between applications across networks.

UDP (User Datagram Protocol): A lightweight, connectionless transport layer protocol that transmits datagrams without guaranteeing delivery or order, making it suitable for time-sensitive applications but more vulnerable to flooding attacks.

HULK (HTTP Unbearable Load King): A DoS tool that generates unique and randomized HTTP requests to overwhelm web servers, aiming to exhaust resources and deny service.

HOIC (High Orbit Ion Cannon): A network stress-testing and DDoS attack tool that floods targets with large volumes of HTTP traffic, often used in coordinated attacks.

Lead City University DO NOT COPY

LOIC (Low Orbit Ion Cannon): An open-source network stress-testing tool commonly exploited to conduct DoS/DDoS attacks by sending massive amounts of TCP, UDP, or HTTP traffic.

Mininet: A network emulator that creates realistic virtual networks on a single machine, widely used for SDN research, prototyping, and performance testing under controlled environments.

Lead City University DO NOT COPY

Endnotes

1. DataReportal, *Internet Use in 2024.* Available online**:** https://datareportal.com/reports/digital-2024-deep-dive-the-state-of-internet-

adoption#:~:text=There%20are%205.35%20billion%20people,the%20first%20time%20during%202023

2. M. A. Dennis, D*enial of Service Attack*. **Encyclopedia Britannica,** 2025. Available online: https://www.britannica.com/technology/denial-of-service-attack

3. M. Gelgi, Y. Guan, S. Arunachala, M. Samba Siva Rao & N. Dragoni, *Systematic Literature Review of IoT Botnet DDOS Attacks and Evaluation of Detection Techniques.* **Sensors,** 24(11), 2024, 3571.

4. J. Singh & S. Behal, *Detection and Mitigation of DDoS Attacks in SDN: A Comprehensive Review, Research Challenges and Future Directions.* **Computer Science Review,** 37, 2020, 100279.

5. D. Uhříček, K. Hynek, T. Čejka & D. Kolář, *BOTA: Explainable IoT Malware Detection in Large Networks.* **IEEE Internet of Things Journal**, 10(10), 2022, 8416-8431.

6. S. Shivaji, *DDoS Attack Detection: Strategies, Techniques, and Future Directions.* **J. Electrical Systems,** 20(9s), 2024, 2030-2046.

7. P. K. Dutta Pramanik, B. K. Upadhyaya, A. Kushwaha & D. Bhowmik, *Harnessing IoT: Transforming Smart Grid Advancements.* **IoT for Smart Grid: Revolutionizing Electrical Engineering,** 2025(1), 2025, 127-174.

8. J. S. Yalli, M. H. Hasan & A. A. Badawi, *Internet of Things (IoT): Origins, Embedded Technologies, Smart Applications, and Its Growth in the Last Decade.* **IEEE Access,** 12, 2024, 91357-91382.

9. M. Selvi, K. L. Mayur & S. K., SVN, *A Comprehensive Review on Authentication, Threats and Privacy Preserving Challenges for Securing Smart Transportation Infrastructure*. **Peer-to-Peer Networking and Applications** 18(6), 2025, 1-32.

10. M. S. M. Shah, Y. B. Leau, M. Anbar & A. A. Bin-Salem, *Security and Integrity Attacks in Named Data Networking: A Survey*. **IEEE Access**, 11, 2023, 7984-8004.

11. N. Anerousis, P. Chemouil, A. A. Lazar, N. Mihai & S. B. Weinstein, *The Origin and Evolution of Open Programmable Networks and SDN.* **IEEE Communications Surveys & Tutorials,** 23(3), 2021, 1956-1971.

12. P. Fondo-Ferreiro, F. Gil-Castineira, F. J. González-Castaño & D. A. Candal-Ventureira, *Software-Defined Networking Solution for Interconnecting Network Functions in Service-Based Architectures.* **IEEE Access,** 10, 2022, 19905-19916.

13. T. Alasail & O. Dakkak, *Exploring the Landscape of SDN-Based DDoS Defense: A Holistic Examination of Detection and Mitigation Approaches, Research Gaps and Promising Avenues for Future Exploration.* **International Journal of Advanced Natural Sciences and Engineering Researches,** 7(4), 2023, 327-349.

Lead City University DO NOT COPY

14. A. K. Jain, H. Shukla & D. A. Goel, *Comprehensive Survey on DDoS Detection, Mitigation, and Defense Strategies in Software-Defined Networks*. **Cluster Computing,** 27(9), 2024, 13129-13164.

15. S. Ghosh, T. Dagiuklas, M. Iqbal & X. Wang, *A Cognitive Routing Framework for Reliable Communication in IoT for Industry 5.0.* **IEEE Transactions on Industrial Informatics,** 18, 2022, 5446–57. Available online: https://doi.org/10.1109/TII.2022.3152646

16. K. Nisar, E. R. Jimson, M. H. A. Hijazi, I. Welch, R. Hassan, A. H. M. Aman, A. H. Sodhro, S. Pirbhulal & S. Khan, *A Survey on the Architecture, Application, and Security of Software Defined Networking: Challenges and Open Issues.* **Internet of Things,** 12, 2020, 100289. Available online: https://doi.org/10.1016/j.iot.2020.100289

17. A. K. Sarica & P. Angin, *Explainable Security in SDN-Based IoT Networks.* **Sensors,** 20(24), 2020, 7326. Available online: https://doi.org/10.3390/s20247326

18. S. Ahmad & A. H. Mir, *Scalability, Consistency, Reliability and Security in SDN Controllers: A Survey of Diverse SDN Controllers.* **Journal of Network and Systems Management**, 29(1), 2021, 9.

19. Z. A. Bhuiyan, S. Islam, M. M. Islam, A. A. Ullah, F. Naz & M. S. Rahman, *On the (in) Security of the Control Plane of SDN Architecture: A Survey.* **IEEE Access**, 11, 2023, 91550-91582.

20. A. O. Sangodoyin, M. O. Akinsolu, P. Pillai & V. Grout, *Detection and Classification of DDoS Flooding Attacks on Software-Defined Networks: A Case Study for the Application of Machine Learning.* **IEEE Access**, 9, 2021, 122495-122508.

21. S. R. Mishra, B. Shanmugam, K. C. Yeo & S. Thennadil, *SDN-Enabled IoT Security Frameworks—A Review of Existing Challenges.* **Technologies,** 13(3), 2025, 121. Available online: https://www.mdpi.com/2227-7080/13/3/121

22. N. H. Thanh, N. N. Tuan, D. A. Khoa, L. C. Tuan, N. T. Kien, N. X. Dung & F. Wamser, *On profiling, Benchmarking and Behavioral Analysis of SDN Architecture Under DDoS Attacks.* **Journal of Network and Systems Management**, 31(2), 2023, 43.

23. N. S. Musa, N. M. Mirza, S. H. Rafique, A. M. Abdallah & T. Murugan, *Machine Learning and Deep Learning Techniques for Distributed Denial of Service Anomaly Detection in Software-Defined Networks—Current Research Solutions.* **IEEE Access,** 12, 2024, 17982–18011. Available online: https://doi.org/10.1109/ACCESS.2024.3360868

24. S. Badotra & B. Chandramouli, *A DDoS Vulnerability Analysis System against Distributed Denial-of-Service Attacks in Software-Defined Networking.* **Electronics,** 11(19), 2022, 3120. Available online: https://doi.org/10.3390/electronics11193120

Leadcity University DO NOT COPY

25. M. A. Aladaileh, M. Anbar, I. H. Hasbullah, Y. W. Chong & Y. K. Sanjalawe, *Detection Techniques of Distributed Denial of Service Attacks on Software-Defined Networking Controller—A Review.* **IEEE Access,** 8, 2020, 143985–143995. Available online: https://doi.org/10.1109/ACCESS.2020.3015331

Lead City University DO NOT COPY

Chapter Two

Literature Review

2.1 Conceptual Review

Computer networks are fundamentally organized into three interrelated functional planes: the data plane, control plane, and application plane (see Figure 2.3). Each plane plays a distinct and essential role in the overall network architecture. The data plane consists of networking devices such as switches and routers responsible for the efficient forwarding of data packets. The control plane is responsible for determining how data should flow through the network. It does so by using protocols that populate the forwarding tables of devices in the data plane. The application plane

hosts software services used to monitor, configure, and manage the network remotely. It is where network policies are defined; these policies are enforced by the control plane and executed by the data plane[1].

In traditional network architectures, these planes particularly the control and data planes are tightly coupled within each physical device. This integration results in a highly decentralized structure. While this approach was originally considered beneficial for ensuring resilience in early Internet designs, it has since revealed considerable limitations. The static and monolithic nature of traditional networks has led to architectures that are rigid, complex to manage, and resistant to rapid innovation[2]. One major drawback of this tightly coupled model is the operational complexity it imposes. Network administrators are often required to manually configure each device using vendor-specific, low-level commands even when implementing high-level policies. This makes network management not only labor-intensive but also error-prone. Consequently, the development and deployment of new routing mechanisms or network protocols can take as long as 5 to 10 years to reach production environments[1]. Furthermore, the lack of separation between control logic and data forwarding functions has constrained the scalability and flexibility of traditional networks. These limitations have contributed to a vertically integrated industry landscape, where both hardware and software are often bound together in proprietary systems. This environment hinders openness, interoperability, and the pace of technological advancement.

The emergence of Software-Defined Networking (SDN) seeks to address these challenges by decoupling the control and data planes and introducing programmability through the application plane. This paradigm shift paves the way for more agile, centralized, and automated network management. This bifurcation provides the foundation for a highly programmable, scalable, and

Lead City University DO NOT COPY

flexible network environment, overcoming many of the limitations associated with traditional network models. The term Software-Defined Networking was originally introduced in the context of research on OpenFlow at Stanford University, where the goal was to centralize network control logic while simplifying individual forwarding devices[3]. In its original conception, SDN refers to an architecture in which a logically centralized control plane often referred to as a network operating system (NOS) manages the network's forwarding state remotely. This controller dictates network behavior, while switches and routers function primarily as packet-forwarding elements[4]. The separation of control and forwarding logic is made possible through well-defined programming interfaces (APIs), which facilitates communication between the control and data planes. Among these interfaces, the OpenFlow protocol has emerged as a widely adopted standard for enabling such interactions [5,6]. Figure 2.1 illustrates the packet-handling process in an OpenFlow-enabled switch, one of the foundational technologies underpinning SDN. When a packet arrives, the switch first checks its internal flow table to determine whether a predefined rule exists. If a match is found, the packet is forwarded accordingly. If not, the switch sends a packet_inmessage to the controller to request guidance. The controller then generates an appropriate flow rule, returns it to the switch via a packet_out message, and the rule is stored in the flow table for future reference [7]. Although SDN and OpenFlow originated in academic research settings[6], they have rapidly transitioned into the commercial domain. Today, many enterprise-grade switches support the OpenFlow API. Major technology firms including Google, Facebook, Microsoft, Verizon, and Deutsche Telekom have endorsed this paradigm by contributing to the Open Networking Foundation (ONF), an organization dedicated to advancing SDN technologies[2]. Researchers have explored many aspects of SDN architecture in detail[8,9,10], and comprehensive introductions to OpenFlow are available in[8,9].

Lead City University DO NOT COPY

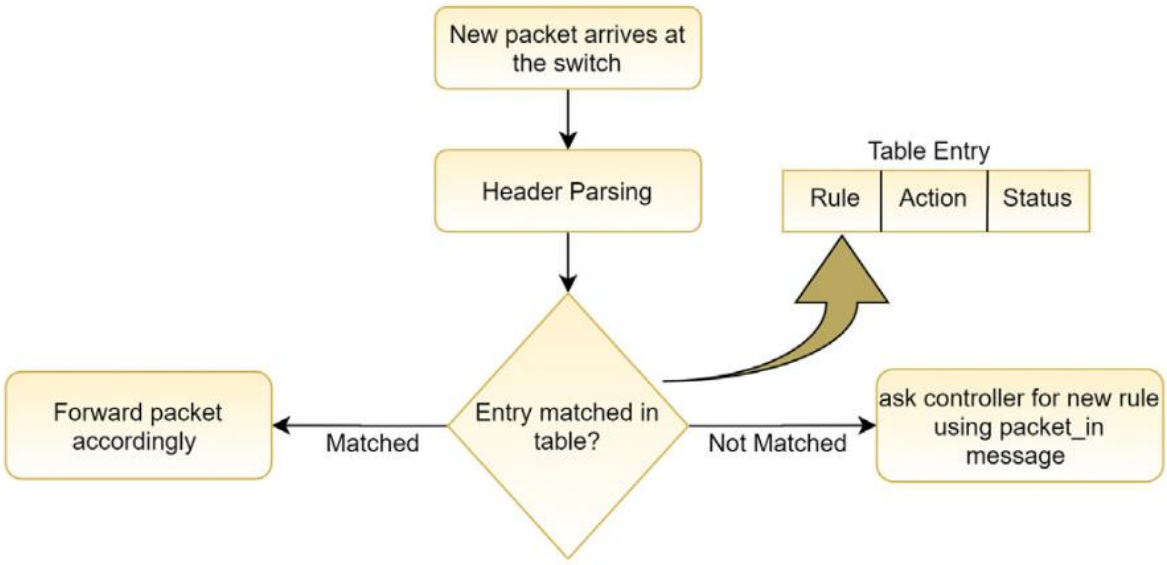


Figure 2.1 Processing at a switch while receiving a new packet

2.1.1 OpenFlow

Most switches and routers available in the market are vendor specifics, running their own IOS and do not typically provide an open software platform. This creates the need to virtualise their hardware or software next to nothing. This closed source IOS does not give room for researchers to experiment with new ideas and network vendors are understandably nervous about disclosing their extensive and fragile distributed protocols and algorithms[11]. The reason for this is not far-fetched; it simply lowers the barrier for new competitors.

The OpenFlow started as an academic experiment and rapidly gained significance which subsequently led to the consortium of industry giants to form Open Networking Foundation (ONF)[5]. OpenFlow brings to the fore the concept known as Application Programming Interface (API) which allows direct access to manipulation of the forwarding plane of network devices such

Lead City University DO

as routers and switches, both physical and virtual[12]. An OpenFlow switch typically consists of a flow table as shown in Figure. 2.2, which performs packet lookup and forwarding. Each table in an open flow switch holds a set of entries that consist of the following:

- Header fields or match fields, with information found in packet header, ingress port, and metadata used to match incoming packets.
- Counters, used to collect statistics of particular flow, such as number of received packets, number of bytes, and duration of the flow.
- A set of instructions to be applied after a match rule that dictates how to handle matching packets[14].

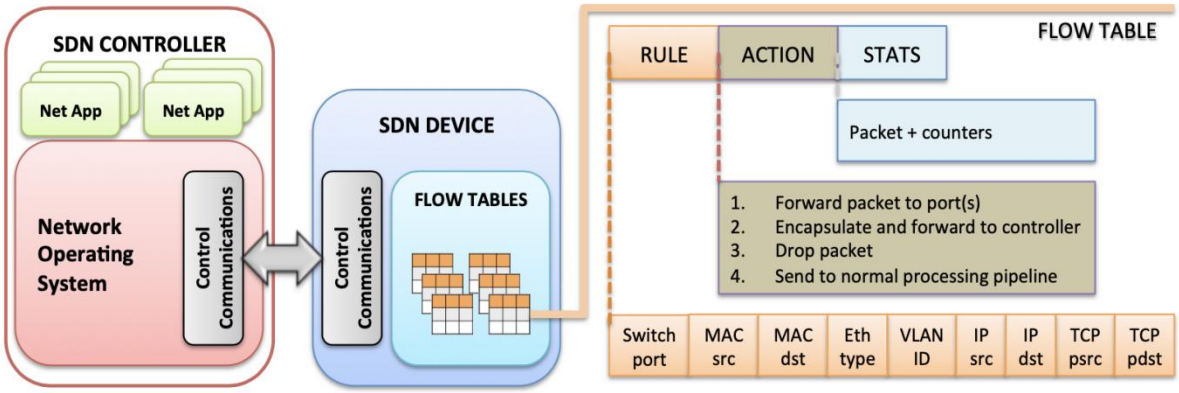


Figure 2.2 OpenFlow enabled SDN device

Inside an OpenFlow device, there is a set of algorithm that defines what goes where when a packet arrives based on the matching flow rule. A flow rule is a combination of different matching rule. If there is no default matching rule, then, the packet will be discarded. The priority of rules follow the natural sequence number of the table and the row order. It is worthy of note that the throughput

Lead City ... DO NOT COPY

of commercial OpenFlow switch is relatively low (500-1000 flow-mod per second) which is a limiting factor[12].

2.1.2 Traditional Networks and SDN

In traditional network architectures, each network device integrates both the data forwarding functions and the embedded control logic. As a result, implementing changes to network policies requires manual reconfiguration of each individual device often using vendor-specific command-line interfaces. This process is not only labor-intensive and error-prone but also poses a significant barrier to rapid policy updates and innovation[11,12,13].

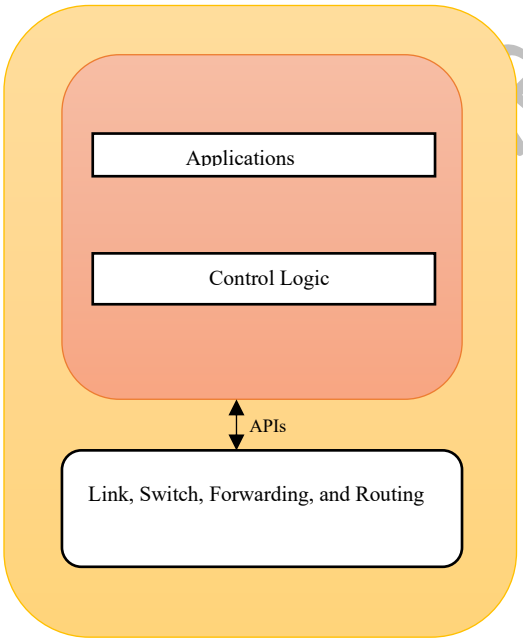
In contrast, SDN abstracts and centralizes the control logic through a logically centralized controller, often referred to as the Policy Maker. This controller manages the network holistically and communicates with the underlying infrastructure using standardized southbound application programming interfaces (APIs), such as OpenFlow[14,15,16]. As a result, administrators can deploy and modify policies remotely and dynamically across the entire network with minimal effort[17].

Moreover, the centralized nature of SDN enables rapid policy deployment and system-wide updates, overcoming the hardware limitations and proprietary constraints commonly found in traditional network devices[18,19]. Vendor-specific switches in traditional networks often lack the flexibility needed for modern dynamic networking environments, whereas SDN provides a uniform and programmable control interface irrespective of the underlying hardware[20]. Additionally, SDN architectures support a more robust testing environment for new policies and network functions. Traditional networks typically offer limited capacity for experimental policy deployment due to the risk of service disruption and rigid hardware dependencies. In contrast,

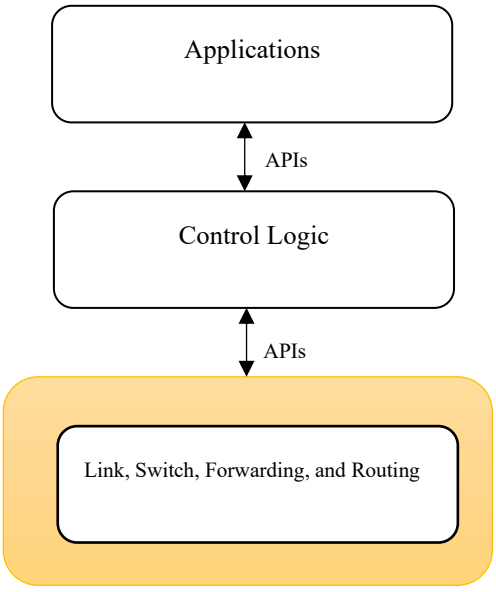
Lead City University DO NOT COPY

SDN allows for greater flexibility and safe experimentation, which is particularly valuable in research, security enforcement, and quality of service optimization[21,22].

ersity DO NOT COPY



Network Device in IP Network



Network Device in SDN

Figure 2.3 Comparison of network device in IP network and SDN.

A summary of the key differences between traditional networks and SDN is presented in Table 2.1 highlighting the enhanced programmability, centralized control, and innovation potential enabled by SDN.

Lead City University DO NOT COPY

Table 2.1 Comparison of Traditional Networks and SDN Architectures

Parameter	Traditional Network	Software-Defined Network (SDN)
Network Management	Requires individual device updates; labor-intensive and complex.	Centralized management enables streamlined device configuration.
Programmability	Limited due to proprietary hardware and static configuration.	Highly programmable via centralized controllers and open interfaces.
Performance	Constrained by lack of centralized control and network-wide visibility.	Optimized through dynamic, policy-driven control across the entire network.

Configuration	Manual, vendor-specific commands required for each device.	Unified and automated configuration via controller and southbound APIs.
Innovation Speed	Slower due to hardware dependency and lack of abstraction.	Faster due to decoupled control and data planes and flexible software-based design.
Testing Environment	Limited due to static, production-bound hardware environments.	Supports isolated, dynamic testing environments for policy experimentation.
Global Network View	No holistic network visibility; control logic is device-specific.	Controller maintains a global, real-time view of the entire network infrastructure.

Traditional networking architectures typically allocate a dedicated computing unit to manage each individual network task. While this method is effective in small-scale or static network environments, it becomes increasingly inefficient and difficult to scale as the network grows[12]. Managing and maintaining such systems across large infrastructures introduces substantial complexity, rigidity, and operational overhead[15]. SDN offers a more flexible and programmable alternative that addresses several of the limitations inherent in traditional network models[18]. By decoupling the control plane from the data plane, SDN centralizes network intelligence and enables dynamic, policy-driven management. This architectural shift simplifies configuration, improves scalability, and enhances overall security responsiveness. Table 2.2 provides a comparative overview of how traditional networks and SDN handle key security challenges[23].

Lead City University DO NOT COPY

Lead City University DO NOT COPY

Table 2.2. Comparison of Security in Traditional Networks vs SDN

Security Challenge	Traditional Network Approach	SDN-Based Approach	Benefits of SDN
Perimeter Security	Perimeter defined through physical objects. Each device is statically configured and operates autonomously.	Perimeter defined at the application layer. Different policies can be applied to external and internal traffic. A single configuration is possible for all security devices within a domain.	Policies are decoupled from the physical perimeter and can be enforced based on application-layer attributes. Security complexity remains manageable even with

			changes in the logical perimeter.
New Security Threats	Security signatures must be identified manually. Operators rely on local tools to detect threats. Malicious users may shift to another vulnerable port to launch further attacks.	End-to-end visibility is provided through a centralized controller. Operators can react to threats quickly. Fine-grained countermeasures can be applied in real time.	Significant reduction in the number of network devices required for security processing. Enhances response time and reduces capital expenditure.
Scalability	Requires a proportional increase in physical hardware to meet growing user demands.	Uses virtualised processes to scale effectively without requiring extensive hardware expansion.	Improves resource allocation and utilization, enhancing overall network scalability.
Patch Management	Challenging due to limited resources on embedded devices and inconsistent update capabilities.	Centralized patch management enables rapid and uniform responses to new threats.	Simplifies and accelerates the deployment of security updates across the network.

2.1.3 SDN Architecture

SDN represents a paradigm shift in the design and operation of computer networks, marking one of the most significant technological transformations in the field to date[2]. The foundational principle of SDN is the decoupling of the network's control logic from the underlying physical devices. Unlike traditional Internet Protocol (IP)-based networks where control and data forwarding functions are integrated within individual devices SDN introduces a layered approach that enhances programmability, simplifies network management, and promotes the development

Lead City University DO NOT COPY

of interoperable systems. A typical SDN architecture is structured into three logical layers: the application layer, the control layer, and the data layer (see Figure 2.3). Each layer is responsible for a specific set of tasks and communicates with adjacent layers through well-defined interfaces.

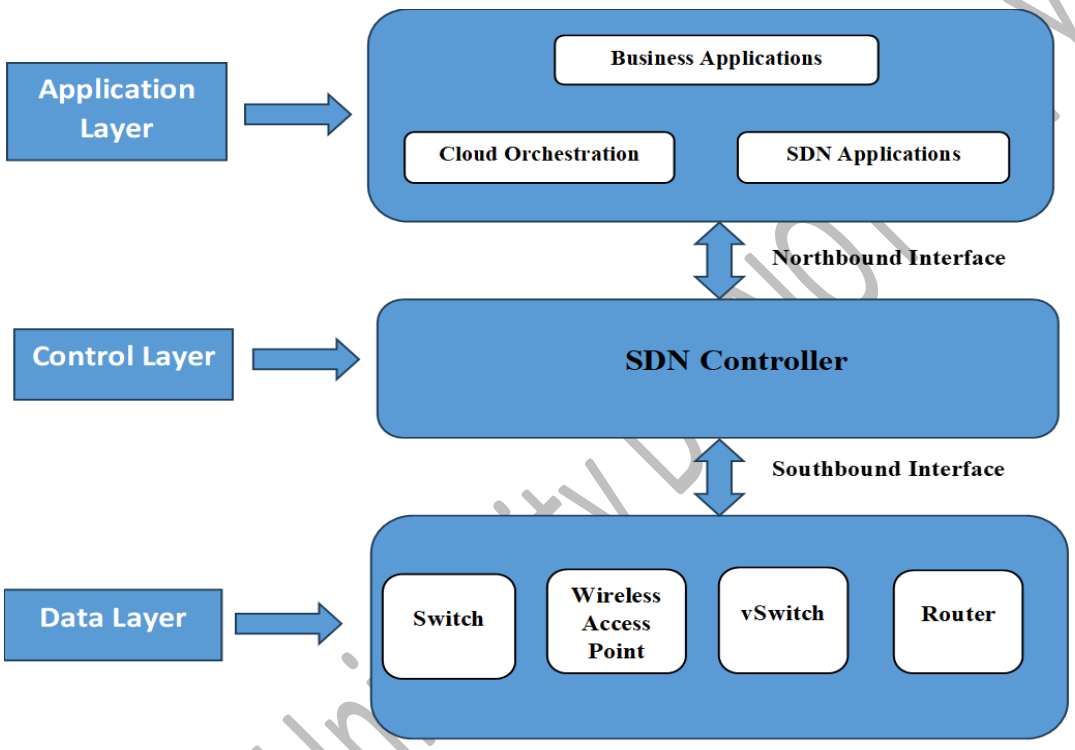


Figure 2.4 Typical SDN architecture.

- **Application Layer:** Positioned at the top of the architecture, this layer hosts software applications that govern business logic and security policies. It is responsible for functions such as firewall management, mobility support, intrusion detection and prevention (IDS/IPS), traffic metering, routing, Quality of Service (QoS) enforcement, and load balancing. These applications interact with the control layer via standardized northbound interfaces, allowing developers to implement high-level policies without direct interaction with the underlying hardware[25].

Lead City University Library

- **Control Layer:** This middle layer functions as the network's central intelligence. Often referred to as the Network Operating System (NOS) or simply the SDN controller, it orchestrates the overall behavior of the network. Through programmable logic, it determines how data flows should be handled whether to forward, drop, or reroute packets and enforces those decisions across the infrastructure[26]. Although logically centralized, the control layer can be physically distributed to enhance scalability and resilience. Communication within and across controllers occurs via eastbound and westbound interfaces, while interaction with the data layer is achieved using southbound APIs such as OpenFlow.

- **Data Layer:** At the foundation of the architecture, the data layer consists of both physical and virtual network devices including switches, routers, and access points that are responsible for the actual forwarding of packets. These devices operate under the direction of the control layer, executing network policies and forwarding rules as defined by the controller. Virtual switches commonly found in this layer include Open vSwitch, Indigo, Pica8, Nettle, and OpenFlow-compliant implementations[27].

2.1.3.1 The Essential Features of SDN

Software-Defined Networking (SDN) is distinguished by five core features that collectively redefine traditional networking paradigms. These features include centralised control, plane separation, network automation and virtualisation, device simplification, and openness[27].

Centralised Control: With the control plane centralised in the SDN controller, the controller maintains complete visibility of network topology and host connections. This comprehensive awareness empowers network administrators to implement custom forwarding policies and advanced monitoring protocols only limited by the ability of software running on it.

Lead City University DO NOT COPY

Plane Separation: with the data plane separated from the control plane, a level of packet forwarding intelligence has been added to the controller. Consequently, this led to a reduction in long convergence time for changes made in SDN network.

Network Virtualisation: SDN supports extensive network virtualisation, allowing multiple logical networks to operate over a shared physical infrastructure[28]. Each virtual network may utilise a simplified topology, abstracted from the complexity of the underlying hardware. This enhances scalability, portability, and cost-efficiency, particularly for small and mid-sized enterprise environments.

Simplified Device Design: By offloading complex control logic to the central controller, SDN simplifies the architecture of data plane devices. This enables forwarding devices to execute instructions with minimal memory requirements and without the need for embedded autonomous decision-making software. The result is a streamlined network infrastructure that can scale more efficiently and react faster to controller directives. These characteristics collectively enhance SDN's ability to provide flexible, manageable, and programmable networking environments setting it apart from conventional architectures that are limited by static configuration and hardware dependency.

Openness: Openness in SDN refers to the use of standardised interfaces across all compass points of Northbound (application layer), Southbound (infrastructure layer), Eastbound (inter-controller), and Westbound (policy or orchestration layers). These standardised APIs eliminate proprietary dependencies, foster interoperability, and encourage innovation by allowing vendors and developers to contribute extensions and improvements to the platform.

2.1.3.2 Development Platforms for SDN

Lead City University DO NOT COPY

The development and evaluation of Software-Defined Networking (SDN) architectures often require robust simulation and emulation environments. These platforms enable researchers, developers, and network administrators to test network topologies, design control logic, and analyze the performance and behavior of SDN components under various conditions without the need for costly physical infrastructure. A wide range of SDN development tools are currently available, each offering different capabilities in terms of simulation, emulation, licensing, and operating system compatibility. Table 2.3 provides an overview of some widely used SDN development platforms, highlighting their key features and distinctions[29].

Table 2.3 Overview of SDN Development Platforms

Platform	Latest Version	Vendor	Website	Operating System Support	Emulation Mode	Simulation Mode	License Type
Mininet	2.2.1	Stanford University, ON.Lab	www.mininet.org	Ubuntu, Fedora	Yes	No	Free
Trema	0.10.1	NEC Corporation	trema.github.io/trema/	GNU/Debian, Ubuntu, Fedora	No	No	Free (LGPL v2.1)
NS-3	3.26	NS-3 Project	www.nsnam.org	GNU/Linux	No	Yes	Free
Estinet	9.0	Estinet Technologies Inc.	www.estinet.com	Linux, Fedora, Windows, Mac	Yes	Yes	Proprietary

Lead City University DO NOT COPY

2.1.4 Vulnerabilities and Security Issues in SDN

While SDN offers significant improvements in flexibility, programmability, and policy enforcement compared to traditional networks, it also introduces novel security challenges. Among these is the architectural separation of the control and data planes, which, if not properly secured, can expose the network to a variety of attacks. These vulnerabilities must be proactively addressed prior to deployment to ensure a resilient SDN infrastructure.

2.1.4.1 Security Vulnerabilities in SDN by Architectural Plane

SDN introduces new security challenges due to the centralized control and programmability it affords, particularly through the OpenFlow protocol and third-party applications[30]. The vulnerabilities stem primarily from the significant authority granted to SDN controllers and applications. Table 2.4 categorizes common attack types according to the respective SDN architectural planes (Application, Control, and Infrastructure) and assesses their severity levels along with potential countermeasures.

Notably, Distributed Denial of Service (DDoS) attacks represent a critical threat, particularly against the control plane, where such attacks can severely disrupt controller functionality. As highlighted in Tables 2.4 and 2.5, the lack of a standardized vulnerability classification framework persists in SDN research. However, the Common Vulnerability Scoring System (CVSS) has been applied in some studies [31].

Table 2.4 Security Vulnerabilities in SDN by Architectural Plane

Lead City University DO NOT COPY

SDN Plane	Attack Type	Severity Level	Recommended Countermeasures
Application Plane	Information leakage	Low	Implement strong encryption protocols
	Application abuse	Medium	Regularly update and patch software
	API abuse	Low	Enforce strong encryption and access control
	Communication hijacking	Low	Use secure communication channels
	Masquerading	Low	Apply strong authentication and encryption
	DDoS attack	High	Employ islanding, rate limiting, and packet dropping techniques
Control Plane	Network manipulation	High	Deploy distributed controllers from multiple vendors
	DDoS attack	High	Implement islanding, rate limiting, and packet dropping
Infrastructure Plane	DDoS attack	High	Apply islanding, rate limiting, and packet dropping
	Compromised network elements	Medium	Use robust encryption and network segmentation
	System compromise	Low	Change vendor default credentials

Lead City University DO NOT COPY

	Communication hijacking	Medium	Utilize secure encryption methods

2.1.4.2 Identifying Security Flaws Using the STRIDE Model

Security in networked systems is traditionally guided by the CIA triad: Confidentiality, Integrity, and Availability. These three pillars form the foundation of secure system design and operation, as illustrated in Figure 2.5. Confidentiality ensures that data is accessible only to authorized individuals. Integrity guarantees that information remains unaltered from its original or last authorized state. Availability ensures that data and services are accessible to authorized users in a timely and usable manner.

In the context of security analysis, three critical concepts frequently arise:

1. Vulnerability: a weakness or flaw in the system that could be exploited.
2. Threat: a potential event or actor capable of violating system security.
3. Countermeasures : protective techniques or controls designed to mitigate threats and reduce vulnerabilities.

Within SDN environments, two core features pose heightened security risks. First, the ability to dynamically control the network via software introduces a broader attack surface, especially if access control is weak. Second, the controller functions as the network's brain. If compromised, it can give an attacker full control over the entire infrastructure[32]. To systematically classify potential attack vectors, the STRIDE threat model is commonly employed. STRIDE an acronym for Spoofing, Tampering, Repudiation, Information

Lead City University DO NOT COPY

Disclosure, Denial of Service, and Elevation of Privilege offers a comprehensive taxonomy for identifying and analyzing security threats in modern networks[33]. Table 2.3 summarizes these categories as they apply to SDN.

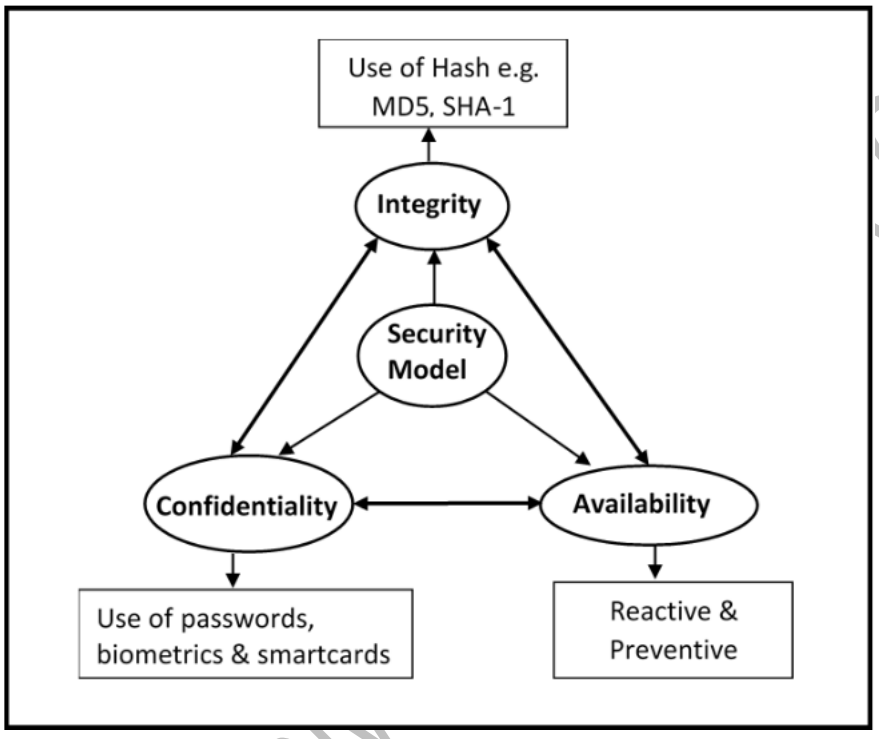


Figure 2.5 CIA Triad

Although the STRIDE model provides a structured framework for threat identification, it is important to note that security threats evolve continuously. The rise of sophisticated adversaries, such as grey-hat hackers, means that emerging attack patterns frequently extend beyond traditional models, necessitating continuous monitoring, adaptive security mechanisms, and updated threat intelligence.

Lead City University COPY

Table 2.5 Microsoft STRIDE attack types and security properties

Attack type	Security property
Spoofing	Authentication
Tampering	Integrity
Repudiation	Non-repudiation
Information disclosure	Confidentiality
Denial of service	Availability
Elevation of privileges	Authorisation

Lead City University DO NOT COPY

The STRIDE threat model categorizes security threats into six distinct classes, each addressing a particular aspect of system vulnerability. These categories are critical for understanding potential attack vectors in SDN environments:

- **Spoofing:** Spoofing involves a malicious actor impersonating a legitimate user or application to gain unauthorized access to the network. This type of attack compromises authentication mechanisms and can lead to significant breaches if not properly mitigated.

- **Tampering:** Tampering refers to the deliberate alteration of data during transmission or storage. This compromises data integrity, potentially leading to incorrect network behavior or false decision-making in automated systems.

- **Repudiation:** Repudiation attacks exploit the absence of proper logging and accountability mechanisms. When a system fails to record user actions effectively, malicious users may perform unauthorized operations and deny responsibility, thereby evading detection.

- **Information Disclosure:** This attack targets the confidentiality of sensitive information. Unauthorized access to or leakage of data undermines user trust and can lead to severe legal and reputational consequences for the organization.

- **Denial of Service (DoS):** A DoS attack seeks to make a system or network resource unavailable to legitimate users, either temporarily or permanently. In enterprise SDN environments, such attacks can significantly disrupt operations and lead to financial losses.

- **Elevation of Privilege:** In this scenario, an attacker gains access to resources or functionalities reserved for users with higher privilege levels. This can lead to full system compromise, especially in centralized SDN architectures where elevated access equates to broader control.

Lead City University DO NOT COPY

Although each attack category varies in severity and impact, implementing robust and layered security measures remains essential. Proactive threat modeling, regular auditing, and continuous security monitoring are recommended to mitigate these threats and ensure the resilience of SDN-based systems.

2.1.4.3 Major Security Threats in the SDN Planes

Seven threat vectors were identified and three are more specific to SDN namely[2]:

Threat Vector 1 – Forged or Malicious Traffic Flows: Attackers may inject forged or fake traffic into the network with the intent of overwhelming SDN switches and controllers. These fabricated flows can exhaust flow table entries, degrade performance, and introduce latency across the network.

Mitigation Strategy: Implement an Intrusion Detection System (IDS) equipped with real-time root-cause analysis capabilities to detect and respond to abnormal traffic patterns.

Threat Vector 2 – Exploitation of Switch Vulnerabilities: Compromised or poorly configured switches may drop, delay, or reroute packets maliciously, thereby causing overload on neighboring switches or controllers.

Mitigation Strategy: Deploy autonomic trust management mechanisms that continuously evaluate and verify the trustworthiness of the switch operating systems and firmware.

Threat Vector 3 – Attacks on Control Plane Communications: This threat targets vulnerabilities in the communication protocols (e.g., TLS/SSL) between SDN controllers and devices.

Lead City University DO NOT COPY

Attackers may exploit these weaknesses to intercept, manipulate, or disrupt controller-device messages.

Mitigation Strategy: Utilize threshold cryptography to secure communications across controller replicas, thereby enhancing confidentiality and resilience.

Threat Vector 4 – Controller-Level Exploits: The SDN controller, serving as the central point of intelligence and decision-making, is a high-value target. A successful attack here can compromise the entire network. Standard IDS mechanisms may prove insufficient.

Mitigation Strategies:

- Secure all sensitive assets (e.g., cryptographic keys, credentials) within the controller.
- Employ diversity in controllers, including variations in programming languages and communication protocols, to reduce systemic vulnerabilities.

Threat Vector 5 – Absence of Trust Mechanisms Between Controllers and Applications: Unlike network devices, management applications often lack standardized mechanisms for trust certification. The interoperability of multi-vendor controllers and applications introduces additional risk.

Mitigation Strategy: Adopt robust autonomic trust management frameworks that can issue and manage certificates, ensuring that application integrity is maintained throughout its lifecycle.

Threat Vector 6 – Vulnerabilities in Administrative Workstations: Administrative stations, if compromised, can serve as launchpads for coordinated attacks due to their high privileges and centralized access.

LeadCity University DO NOT COPY

Mitigation Strategy: Enforce strong access controls, such as multi-factor authentication and dual-authorization protocols, requiring two distinct users to validate access to critical control servers.

Threat Vector 7 – Inadequate Remediation Resources and Recovery Mechanisms: A lack of trusted systems for analyzing and recovering from network failures may result in prolonged downtime and repeated attacks.

Mitigation Strategy: Integrate comprehensive logging, monitoring, and tracing mechanisms within both the control and data planes to facilitate root-cause analysis and secure recovery procedures.

A thorough understanding of these threat vectors is critical for conducting effective risk assessments and designing resilient SDN architectures. Notably, **Threat Vectors 3, 4, and 5** are **unique to SDN**, emerging primarily due to the decoupling of the control and data planes and the centralization introduced by the controller. These SDN-specific risks are summarized in Table 2.4.

2.1.5 Security Challenges in SDN

The logically centralized control and global visibility of the network state offered by SDN significantly enhance network security. Administrators can dynamically manipulate traffic flows and resolve conflicts in real time using a centralized controller[34,35,36]. However, this same centralization introduces new security vulnerabilities. In particular, the controller becomes a critical point of failure and a high-value target for various attacks, such as DoS and DDoS[37]. As SDN technologies continue to be deployed, the associated security challenges are expected to grow in complexity. These vulnerabilities span the three core layers of the SDN architecture: the application plane, control plane, and data plane. A detailed analysis of the security challenges associated with each layer is presented below.

Lead City University DO NOT COPY

2.1.5.1 Application Plane

The software-driven control of networks and the centralized controller architecture cornerstones of SDN both enable innovation and introduce significant security concerns. One key issue is the absence of standardized, secure Open APIs for managing network services through the control plane[38]. This gap allows applications to pose serious threats to network security.

Although OpenFlow provides some flow-based security detection mechanisms through security applications, the available implementations lack maturity and effectiveness[39]. Additionally, the diversity in development environments, paradigms, and programming models leads to interoperability challenges and potential conflicts in security policies. Key concerns at the application plane include:

- Authentication and Authorization: In many SDN architectures, particularly those based on OpenFlow, controller functions are implemented as third-party applications. These applications have privileged access to network resources and services[40]. Therefore, robust authentication mechanisms are essential. However, managing authentication for a large number of diverse applications remains a significant challenge.

- Access Control and Accountability: Since many services are implemented via applications, robust access control and accountability mechanisms are required to ensure network security. Some applications may rely on or embed other applications, enabling potential bypass of access controls. Applications can be categorized into three types[41]: Network-Sensitive Applications, Services for the Network, and Packaged Network Services. Nested applications (i.e., applications embedded within others) present a particularly severe challenge for access control and tracking[42].

Lead City University DO NOT COPY

2.1.5.2 Control Plane

In SDN, the control plane implemented as a logically centralized controller handles all decision-making related to traffic forwarding. As such, it becomes a primary target for attackers. Major security threats include:

- Threats from Malicious Applications: Applications running in the management plane can introduce serious vulnerabilities. The controller is responsible for authenticating and authorizing application access to network resources. Without appropriate isolation, tracking, and auditing mechanisms, malicious applications can cause extensive harm[41]. A customized security evaluation process for different application types interacting via the northbound API is necessary but remains largely unexplored[43].

- Scalability as a Security Risk: In OpenFlow-based SDN, the controller must install flow rules for each new data path. In high-speed networks (e.g., with 10 Gbps links), this can rapidly lead to controller overload and performance bottlenecks[44,45]. Current controller implementations often struggle with high volumes of new flows, making them prime targets for DoS and DDoS attacks. The lack of scalability thus constitutes a major security vulnerability.

- Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) Attacks: These attacks aim to make network resources unavailable to legitimate users. A demonstration was carried out on how SDN's separation of control and data planes can be exploited for DoS attacks[46] . While some have proposed using secondary controllers[47], these too are susceptible to attack. In fact, introducing multiple controllers without proper safeguards can lead to cascading failures, as described in[48].

Lead City University DO NOT COPY

2.1.5.3 Data Plane

In OpenFlow-based SDN, the data and control planes are decoupled. Data plane devices are limited to forwarding traffic based on flow rules issued by the controller. This structural simplicity introduces several security risks:

- Distinguishing Genuine from Malicious Flow Rules: Since switches only forward packets and do not generate flow rules themselves, they are inherently unable to differentiate between legitimate and malicious rules—making the data plane highly dependent on the integrity of the control plane.

- Limited Flow Table Capacity: Each switch has finite memory allocated for flow table entries. Attackers can exploit this by triggering flow table saturation, effectively executing a resource exhaustion or saturation attack.

- Dependence on the Control Plane: The security of the data plane is intrinsically tied to the security of the control plane[45]. A compromised controller can propagate malicious rules to all connected switches, compromising the entire network.

- Vulnerability of the Switch-Controller Link: The communication link between the data plane and the control plane is critical. If the controller becomes unavailable—due to a failure or an attack the switches lose their ability to receive new flow rules, effectively rendering the data plane non-functional. As such, the switch-controller communication channel represents a high-value target for attackers.

Lead City University DO NOT COPY

2.1.6 Security Solutions Across SDN Planes Based on ITU-T Recommendations

The International Telecommunication Union (ITU) emphasizes that a secure network must be resilient against malicious attacks, maintain high availability, ensure system reliability, and provide timely responses to threats[49]. In line with this, the ITU-T outlines eight essential security dimensions that collectively offer protection against major security threats. These dimensions are: access control, authentication, non-repudiation, data confidentiality, communication security, data integrity, availability, privacy. When systematically applied across SDN planes (data, control, and application), these security dimensions contribute to a comprehensive, end-to-end defense strategy. Table 2.5 presents a compilation of security solutions mapped to the ITU-T security dimensions. Each solution includes the mechanism employed and remarks on the current state of development, deployment, or ongoing challenges. These solutions span both research prototypes and practical implementations.

Lead City University DO NOT COPY

Table 2.6 SDN Security Solutions Based on ITU-T Security Recommendations

	Security Dimension	Solution Name(s)	Mechanism Used	Remarks
S	Authentication	FortNOX[51], FSL[52]	Role-based authentication and authorization; admission control policies	Still an open challenge; lacks uniform enforcement
T	Data Integrity	OFHIP[53], VeriFlow, FortNOX, Isolation[54], Others[55]	IPSec (ESP), flow rule legitimacy verification, traffic isolation, identity management	Identity management systems remain underdeveloped
R	Non-Repudiation	OFHIP[53], VAVE[48], Others[56]	Use of permanent identifiers (e.g., HIP, LISP), source address validation	Mechanisms are experimental; further research required
I	Data Confidentiality	OF-RHM[57], FortNOX[51], IBC[58]	Random host mutation, identity-based cryptography, flow rule enforcement	Lacks dedicated systems and mature applications
D	Availability	DISCO[59], McNettle[60], MAESTRO[61], Others	Distributed control plane, parallel processing in multicore systems	Limited research on enhancing availability through integrated security
	Access Control	PermOF[40], FRESCO[62], FSL[52], Resonance[63]	Policy enforcement frameworks, dynamic access	Application plane access control and multi-controller coordination needed

Lead City University DO NOT COPY

E			control, ACL-based architectures	
	Privacy	OF-RHM[57]], Isolation[54]], ident++[64]	Host mutation, traffic isolation, user-defined privacy policies	Lacks systematic privacy enforcement mechanisms
	Communication Security	TLS[65]	Secure controller-switch communication via encryption	Configuration complexity remains a concern

2.1.7 Anomaly Detection in SDN

Unlike traditional networks, SDN relies on a logically centralised controller that coordinates forwarding devices through standardised protocols such as OpenFlow. While this simplifies network management and allows dynamic reconfiguration, it also creates single points of failure and novel attack surfaces[11]. Anomalies in SDN can manifest in many forms, including Distributed Denial of Service (DDoS) attacks, control-plane saturation, traffic irregularities, topology spoofing, and low-rate stealthy attacks. Detecting these anomalies is essential to maintain network reliability, availability, and security. Over the last decade, researchers have proposed a spectrum of detection techniques ranging from traditional rule-based thresholds to advanced machine learning (ML) and deep learning models [66,13].

This section explores anomaly detection in SDN by categorising anomalies and systematically reviewing detection approaches, including rule-based, statistical, ML-based, and emerging GAN-driven frameworks.

2.1.7.1 Types of Anomalies in SDN

Lead City University DO NOT COPY

2.1.7.1.1 Denial of Service and Distributed Denial of Service (DoS/DDoS)

DoS and DDoS attacks remain the most critical threats to SDN. These attacks flood the controller or forwarding devices with illegitimate traffic, exhausting processing resources and degrading network services. Unlike conventional IP networks, SDN is especially vulnerable to packet_in flooding attacks. When switches encounter new flows, they send packet_in messages to the controller for path computation. Attackers exploit this by generating large numbers of new flows, overwhelming the controller's processing capacity [67]. In addition, amplification-based DDoS and reflection attacks also target SDN infrastructures. Low-rate DDoS (LDDoS) attacks have emerged as stealthy variants, sending traffic patterns that mimic legitimate behaviour but degrade performance over time [66]. Their detection is more complex because they blend into normal traffic flows.

Lead City University DO NOT COPY

Table 2.7 DDoS Attacks on layers of SDN Architecture

SDN Plane	Representative DDoS Attacks

Data Plane	Switch-level DDoS, TCP-SYN flood, TCP flood, ICMP flood, Ternary Content Addressable Memory exhaustion (TCAM)
Control Plane	OpenFlow bandwidth exhaustion, amplification-based attacks, Controller resource depletion
Application Plane	Northbound API exhaustion, application-layer DDoS (e.g., Slowloris attack, HTTP flooding)

DDoS (Distributed Denial of Service) attacks can target various layers of the SDN architecture, exploiting the unique characteristics and interfaces of each plane. Table 2.6 outlines representative DDoS attack types based on the plane they affect.

2.1.7.1.2 Traffic Anomalies

Traffic anomalies are deviations from expected network flow patterns. They can be symptomatic of misconfigurations, congestion, or malicious activity. Examples include unusual spikes in bandwidth consumption, abnormal packet size distributions, or long-lived flows. Although not always attacks, they can affect quality of service. Traditional anomaly detection leverages entropy-based measures of IP distributions to detect such irregularities[67].

2.1.7.1.3 Topology Attacks

Lead City University DO NOT COPY

Topology poisoning or spoofing is a unique class of anomaly in SDN. Attackers inject false link or node information into the control plane, causing the controller to compute incorrect paths or redirect traffic. For example, by sending falsified LLDP packets, adversaries can create phantom links. These anomalies undermine the controller's global view of the network and enable man-in-the-middle attacks or black-holing of traffic[11].

2.1.7.1.4 Control-Plane and Flow Table Attacks

Flow tables on switches are finite resources. Attackers exploit this limitation through flow table exhaustion, generating a flood of new flows that rapidly consume table entries. Once exhausted, legitimate flows are dropped or delayed. Similarly, control-plane saturation occurs when the controller is overwhelmed with requests, degrading its ability to respond in real-time[13].

2.1.7.2 Existing Detection Methods

2.1.7.2.1 Rule-Based Approaches

Early SDN anomaly detection relied heavily on rule-based systems. These approaches involve thresholding flow features such as packet counts, connection rates, or entropy of source IP distributions. For instance, information-theoretic methods detect anomalies by computing entropy variations in network flows; sharp entropy deviations often indicate DDoS attacks[67].

The advantages of rule-based methods are simplicity, low computational overhead, and ease of deployment. However, they struggle to detect zero-day attacks or low-rate anomalies, since adversaries can craft traffic that remains within thresholds. They also lack adaptability to the dynamic traffic conditions inherent in SDN[11].

2.1.7.2.2. Statistical Approaches

Lead City University DO NOT COPY

Statistical anomaly detection employs probability distributions, correlation analyses, and information-theoretic measures. Techniques such as chi-square testing, correlation-based flow analysis, and Kullback–Leibler divergence allow comparison between real-time traffic and established baselines[11]. For example, entropy analysis can highlight anomalies by measuring randomness in IP address distributions. Similarly, correlation-based methods examine flow consistency, detecting anomalies when expected relationships between flow variables break down. While statistical methods are explainable and require fewer computational resources, they depend on stable baseline traffic profiles. In highly dynamic SDN environments with fluctuating workloads, maintaining such baselines is challenging, reducing detection accuracy [13].

2.1.7.2.3 Machine Learning Approaches

ML-based approaches have gained prominence due to their ability to learn complex traffic patterns.

- **Supervised ML:** Algorithms such as Random Forest (RF), Support Vector Machine (SVM), K-Nearest Neighbour (KNN), and Naïve Bayes (NB) are widely applied. Comparative studies on the CICDDoS2019 dataset show RF and KNN outperform in accuracy but require higher execution times. SVM strikes a balance between detection accuracy and computational efficiency [68].

- **Unsupervised ML**: Clustering methods, including k-means and DBSCAN, detect anomalies without labelled data, making them suitable for LDDoS and unknown attack types [66].

- **Hybrid ML:** Integrating multiple classifiers or combining ML with feature selection has been shown to reduce false positives and improve scalability.

Lead City University DO NOT COPY

These models benefit from adaptability and generalisation but face challenges such as reliance on training datasets, sensitivity to feature selection, and difficulty in handling imbalanced traffic data [13].

2.1.7.2.4 Deep Learning Approaches

Deep learning has emerged as a powerful extension of ML, leveraging multi-layered architectures for automatic feature extraction.

- **Convolutional Neural Networks (CNNs):** Applied to traffic matrices to capture spatial correlations between flow features.
- **Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM):** Model temporal dependencies in traffic streams, useful for detecting stealthy LDDoS.
- **Hybrid models:** SDN-Defend demonstrates a lightweight hybrid of CNN and Extreme Learning Machine (ELM), providing near real-time anomaly detection with reduced computational complexity[67].

Deep learning provides superior accuracy compared to traditional ML. However, it requires large labelled datasets, is computationally intensive, and often lacks interpretability [11].

2.1.7.2.5 GAN-Based Detection

Generative Adversarial Networks (GANs) are increasingly employed in SDN anomaly detection. GANs consist of a generator, which produces synthetic samples, and a discriminator, which distinguishes real from fake. By training both adversarially, GANs learn complex data distributions.

Lead City University DO NOT COPY

DadGAN [69] is a supervised GAN-based framework tailored for SDN anomaly detection. It uses an autoencoder-based generator to capture traffic patterns and a discriminator to classify anomalies. Performance metrics such as precision, recall, and F1-score demonstrated its robustness against point anomalies. Beyond detection, GANs are also used to generate synthetic training data to overcome dataset scarcity[70]. This dual role of anomaly detection and data augmentation makes GANs especially valuable in SDN environments where labelled datasets are limited.

2.1.8 Machine Learning and GANs for Synthetic Data Generation

The rapid evolution of software-defined networking (SDN), IoT, and next-generation (NextG) systems has highlighted a pressing challenge: the scarcity of high-quality labelled datasets for training anomaly detection and intrusion prevention systems. Real-world data are often scarce, imbalanced, or sensitive, limiting their utility for research and operational deployment [70]. Synthetic data generation has thus emerged as a transformative approach, enabling researchers and practitioners to create artificial yet statistically valid datasets that replicate the patterns, distributions, and anomalies found in genuine network traffic [70]. Machine learning (ML) methods, from statistical sampling to advanced neural networks, provide systematic mechanisms for generating synthetic data at scale. Among these, generative adversarial networks (GANs) have gained prominence, producing data distributions indistinguishable from real-world inputs through adversarial training [71]. In security-sensitive contexts such as SDN anomaly detection, synthetic data allow for (i) augmentation of rare attack traffic, (ii) simulation of adversarial scenarios, and (iii) privacy-preserving data sharing across organisations [72].

This section reviews the spectrum of ML-driven synthetic data generation, beginning with traditional rule-based and statistical approaches, moving through probabilistic models and autoencoders, before focusing on GANs as the dominant paradigm.

Leadcity University DO NOT COPY

2.1.8.1 Machine Learning for Synthetic Data Generation

2.1.8.1.1 Programmatic and Rule-Based Approaches

Early methods of synthetic data generation involved programmatic rule-based systems, where explicit logical constructs governed the relationships within the generated datasets [70]. These systems allowed predictable replication of specific data structures and patterns, making them valuable for structured network traffic emulation. However, while deterministic, they lack flexibility and often fail to capture the stochastic complexity of real-world traffic anomalies.

2.1.8.1.2 Random Sampling and Resampling

Another fundamental technique involves random sampling, where observations are drawn from existing datasets to create new distributions while preserving statistical features such as mean, variance, and correlations . This approach is useful for addressing class imbalance problems, such as underrepresented denial-of-service (DoS) traffic, without directly exposing sensitive information. Random oversampling and bootstrapping methods remain prevalent in SDN datasets such as CICDDoS2019 and UNSW-NB15.

2.1.8.1.3 Statistical and Probabilistic Models

Statistical approaches extend these principles by employing parametric and non-parametric models. Parametric models assume well-defined distributions (e.g., Gaussian or regression-based) and generate data by sampling estimated parameters . Non-parametric models, such as kernel density estimation and nearest-neighbour resampling, make fewer assumptions about underlying distributions, enabling the generation of more heterogeneous and irregular patterns [70].

Lead City University DO NOT COPY

These methods excel when the underlying probability distribution of traffic features, such as flow duration or packet inter-arrival time, is known or approximable. In SDN anomaly detection, probabilistic replay of flow table statistics can recreate both benign and attack behaviours for classifier training [13].

2.1.8.1.4 Differential Privacy–Based Synthesis

Synthetic data generation has also incorporated differential privacy (DP), where calibrated noise is added to maintain statistical fidelity while preventing individual record re-identification . DP-based ML synthesis is particularly valuable in security domains where sensitive packet headers or user identifiers cannot be directly shared. By ensuring outputs remain statistically indistinguishable whether a specific individual's data are included or excluded, DP preserves both utility and confidentiality.

2.1.8.1.5 Machine Learning Based Approaches

Machine learning has dramatically expanded synthetic data generation beyond sampling and parametric models. Neural networks, variational autoencoders (VAEs), and large language models (LLMs) now enable data synthesis across structured, unstructured, and time-series domains[70].

- **Autoencoders and VAEs** compress inputs into latent spaces, reconstructing data while introducing controlled variability . In anomaly detection, VAEs can simulate traffic with subtle deviations, aiding in training models sensitive to low-rate distributed denial-of-service (LDDoS) patterns.

- **LLMs and generative transformers** extend synthetic data to text-heavy security applications, such as generating phishing emails or log entries for adversarial training [70].

Lead City University DO NOT COPY

In SDN-specific contexts, ML-based systems have been applied to synthesise flow statistics under attack conditions, improving classifier robustness [73]. Simulation environments such as Mininet are often used to validate these generated datasets before operational deployment [74].

2.1.9 GAN Architectures for Synthetic Data Generation

Generative adversarial networks (GANs), first proposed by [75], represent a significant leap in generative modelling. A GAN consists of two neural networks, a generator and a discriminator, locked in an adversarial game: the generator produces synthetic data, while the discriminator attempts to distinguish between real and generated samples. Through this competition, the generator learns to produce data distributions increasingly indistinguishable from real-world datasets [71].

2.1.9.1 Standard GANs

The vanilla GAN architecture employs multilayer perceptrons for both generator and discriminator. While effective for basic synthetic data tasks, early GANs often suffered from instability during training, particularly mode collapse, where the generator produced limited diversity[72]. In security applications, this instability can yield unrealistic network traces that fail to represent attack diversity.

2.1.9.2 Deep Convolutional GANs (DCGANs)

Deep convolutional GANs incorporate convolutional layers to improve feature extraction and synthesis. DCGANs have proven especially effective in domains where data can be represented as image-like matrices, such as flow feature heatmaps or packet payload representations[72]. Their stability and ability to capture spatial hierarchies make them suitable for simulating high-dimensional SDN traffic.

Lead City University DO NOT COPY

2.1.9.3 Conditional GANs (cGANs)

Conditional GANs extend the vanilla model by incorporating auxiliary information, such as class labels or network conditions, into both generator and discriminator. This allows targeted data generation, for example, producing synthetic traffic labelled as DDoS or benign depending on training requirements[72]. In anomaly detection, cGANs enable balancing datasets by oversampling rare attack categories, ensuring classifiers do not become biased towards majority classes.

2.1.9.4 Tabular GANs

While many GAN innovations focused on images, recent work has developed architectures optimised for structured, tabular data. CTGANs (Conditional Tabular GANs) and TGANs (Tabular GANs) are designed to capture mixed data types continuous and categorical commonly found in SDN flow tables and network logs[70]. Their ability to handle imbalanced categorical variables makes them invaluable for anomaly detection datasets where rare attack types coexist with abundant benign flows.

2.1.9.5 BiGANs and f-AnoGANs

Bidirectional GANs (BiGANs) and variants such as f-AnoGAN integrate encoder pathways, mapping real data into latent spaces. These models provide not only synthetic data generation but also anomaly scoring by measuring reconstruction errors. In SDN contexts, BiGANs can detect traffic that deviates from learned latent distributions, serving as both generators and detectors[72].

2.1.9.6 GANs with Privacy-Preserving Extensions

Recent advances embed differential privacy into GAN training (DP-GANs), ensuring synthetic samples preserve global statistical properties while preventing leakage of sensitive individual records. In cross-organisational settings, federated GANs allow collaborative synthetic data

Lead City University DO NOT COPY

generation without centralising raw datasets[70]. These features are particularly relevant in sectors such as healthcare or finance, where network data are highly sensitive.

2.1.10 Applications of Synthetic Data in Security and SDN

GANs and ML-based synthetic data generation techniques have found extensive applications in cybersecurity, particularly in augmenting datasets for anomaly detection, privacy preservation, and adversarial training.

Augmentation of Limited Network Traces

One of the primary uses of GANs in SDN security is augmenting scarce datasets. Attack traffic such as low-rate DDoS or botnet command-and-control flows are rare in real-world captures but critical for robust anomaly detection[66]. GANs generate synthetic attack samples that enrich training corpora, reducing class imbalance and improving classifier recall[68]. For example, studies using CICDDoS2019 have demonstrated that classifiers trained with GAN-augmented data achieve detection accuracies 5–10% higher than those trained on raw datasets alone[13].

Privacy-Preserving Data Generation

GANs also serve as privacy-preserving alternatives to raw data sharing. In multi-organisation collaborations, sharing network traces poses confidentiality risks. By generating synthetic datasets statistically similar to original traffic but without exposing identifiable flows, GANs facilitate cross-institutional research without breaching privacy[70]. DP-GANs further guarantee differential privacy, providing formal assurances of individual record protection.

Adversarial Training and Robustness

Adversarial robustness is an emerging frontier where GANs are used to simulate sophisticated evasive traffic. Attackers increasingly deploy adversarial examples designed to bypass anomaly

Lead City University DO NOT COPY

detectors. GANs can generate such adversarial flows, which are then incorporated into training to harden detectors against real-world evasion attempts[71]. This approach mirrors practices in computer vision, where adversarial samples strengthen model resilience.

Cross-Domain Adaptation in Wireless and NextG Contexts

GAN-generated synthetic data extend beyond SDN into wireless and NextG contexts. For example, GANs are applied to spectrum sharing, jamming detection, and user authentication by synthesising wireless signals under diverse conditions[71]. In SDN-enabled 5G and IoT deployments, synthetic data address the heterogeneity of traffic by modelling interactions across multiple domains.

Benchmarking and Simulation

Synthetic data are also used for benchmarking new detection algorithms. Instead of relying on outdated datasets like KDD Cup 1999, GANs allow researchers to simulate modern attack scenarios aligned with current SDN architectures[11]. By incorporating features such as OpenFlow-specific packet_in flooding or flow table overflows, synthetic datasets provide realistic testbeds for anomaly detection research.

2.1.11 Sensitivity Analysis of Detection Parameters in SDN

SDN is a paradigm change that abstracts lower-level functions, enabling network administrators to coordinate network services[1]. This abstraction outperforms traditional network topologies in promoting a dynamic and effective approach to network management. It gets harder to optimise security and performance as networks get bigger and more complex[76]. With its advanced techniques to increase SDN's efficiency, SA becomes an essential tool in this situation[77]. In order to achieve this, SA identifies important parameters and evaluates how they affect network

Lead City University DO NOT COPY

performance. An important part of developing and assessing models is clearly defining and understanding how changes in model parameters impact predictions. A crucial aspect of this process is SA, which looks at how important input parameters are to the model's output and measures how each input affects the final results [78]. SA typically involves analysing the relationships between numerous sources of uncertainty in model inputs and uncertainties in model outputs, whether they be numerical or not [79]. Although SA and uncertainty analysis (UA) are closely linked, UA is different in that UA focuses on describing the model's predictiveness uncertainties without identifying the assumptions that contribute most to the uncertainties [77]. Since uncertainties must be estimated before they can be assigned, UA should ideally come before SA [80]. Nevertheless, this order is not necessarily required, and the quantification of uncertainty may not be necessary for model optimisation applications [80]. There are several classifications for SA techniques in the literature. For example, SA approaches are categorised by [81] and [82] according to their framework as either deterministic or statistical SA and their scope as either local or global SA. Statistical frameworks for SA are typically thought to originate from experiment design in [83,84,85] with both local and global SA approaches being classified based on the parameter space of interest. As a result, SA methods are also divided into local and global SA methods in this study.

Lead City University DO NOT COPY

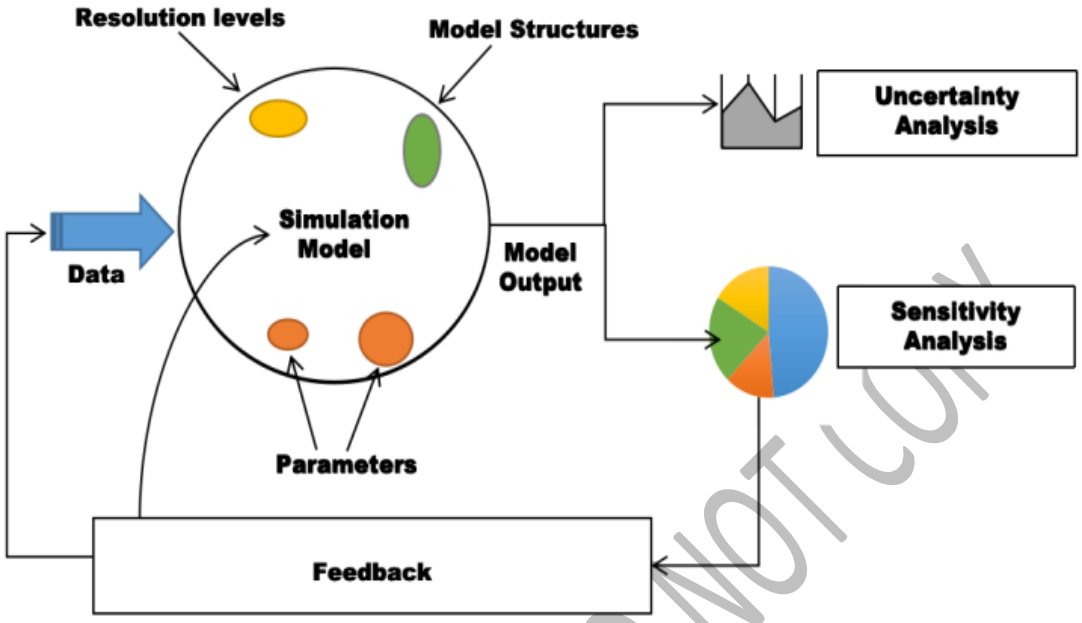


Figure 2.6 Illustration of a typical sensitivity analysis implementation.

Source: Author's Computation 2026

In light of the growing convergence of AI and SDN, several useful approaches for implementing AI-driven SA in this setting have recently been proposed. The performance optimisation, security, robustness, and characterisation of SDN environments have all benefited greatly from these AI-driven approaches [86]. This study discusses some of the contemporary techniques that have been documented in recent literature with the goal of highlighting their real-world effectiveness and practical applications.

The origins of SA date back to scientific and engineering studies in the early 20th century, where it was used to assess the performance and reliability of systems under varying conditions [87]. SA is now an essential component of contemporary analytical procedures thanks to developments in computational techniques. The primary goal of SA in many applications is to determine how different values of independent variables impact a given dependent variable under a certain set of

Lead City University DO NOT COPY

assumptions [78]. SA is an important technique for understanding and improving models, which makes it vital in decision-making processes since it assists in identifying the factors that have the greatest influence on how complex systems behave [88].

In general, SA approaches can be classified into two categories: local or global [88]. While global SA (GSA) assesses the effects throughout the whole range of input possibilities, leading to a more comprehensive view of the dynamics between inputs and outputs, local SA (LSA) looks at the effects of small deviations around a particular input benchmark [88]. Quantifying the amount of output uncertainty resulting from various input uncertainty sources is a component of SA [88]. A basic example of how input variables and output responses interact is shown in Figure 1 [80], which establishes the foundation for a typical SA process. Figure 1 shows how the model processes the several sources of uncertainty to produce an empirical output distribution (shown by the grey curve).

The variability of the model's output, as indicated by its variance, can then be analyzed according to its sources, facilitating a thorough SA. LSA and GSA are further discussed as follows:

2.1.11.1 Local Sensitivity Analysis (LSA)

LSA assesses the local impact of input factor changes on a model's response by focusing on the sensitivity bothering on a set of parametric values [89,90]. Other parametric values are usually fixed while analysing the sensitivity of an input factor in LSA. LSA is frequently applied by assessing the impact of small deviations from nominal input parametric values on the model's output [91]. As shown in [92], this disturbance is frequently applied sequentially to a single parameter, approximately predicting the first-order partial derivative of the model's output with respect to the perturbed

Lead City University DO NOT COPY

parameter. Effective adjoint methods that handle a variety of parameters can be used to estimate this derivative, as stated in [93].

In addition to being numerically efficient, sensitivity coefficients generated from deterministic LSA have the advantage of being easy to understand, regardless of the method employed [94]. Their intuitiveness is mostly due to their equivalency to the derivative of the output with respect to each parameter around a predetermined point (i.e., nominal parameter values). Consequently, these parametric coefficients can be readily compared across a wide range of modelled systems, irrespective of the range of parameter variations as demonstrated in [81]. LSA may also involve using the partial derivatives of the model's response to the input parameters in order to assess the sensitivity for a specific fixed set of input parametric measures or values [95,96].

Taking into consideration the model, F, for the system is described as follows:

$$y = F(x, \gamma)$$

The dependent variable, y, is affected by the independent variation of x and the parameters $\gamma = [\gamma_1, \cdots, \gamma_r]$ of F, as revealed by the LSA. In this context, the central concept of LSA is to quantify the impact of the model parameter, γ, or pattern characteristics, x_i, $i = 1, \cdots, N$, on the output value y_j, $j = 1, \cdots, N$, where N and J represents the number of outputs and features, respectively, following a training procedure [97,98]. The coefficients ($S_{j,i}$) that quantify this influence can be determined as follows:

$$S_{j,i}^{(p)} = \frac{\partial y_j(x_1^{(p)}, x_2^{(p)}, \ldots, x_N^{(p)})}{\partial y_j}$$

Lead City University DO NOT COPY

The pth training pattern, $p = 1, \cdots, P$, in the equation above gives the sensitivity value of the jth model output on the ith attribute of the input vector x. When the inputs to the model behave linearly or additively, or when there is minimal uncertainty in the input data, applying LSA can be particularly effective [99].

2.1.11.2 Global Sensitivity Analysis (GSA)

GSA examines how different sources of input uncertainty contribute to the overall uncertainty in a model's output [79]. By systematically exploring the entire input parameter space, GSA aims to describe the output response surface in detail and assess the relative influence of each input factor. This process often focuses on identifying key features of the system, such as saddle points, local minima, and maxima, within a global deterministic framework [81,93]. By pinpointing these critical regions, GSA provides insights into the system's behavior under a wide range of conditions. Various statistical approaches have been developed for GSA, including models that analyse output variance [100,101], correlations [102], or elementary effects [103] to evaluate the level of dispersion and interdependence among variables [84,88].

GSA methods also offer a statistical characterisation of the output response surface, providing valuable insights into how uncertainties in the inputs propagate through the model [83,84]. For instance, variance-based approaches quantify the proportion of output variance attributable to each input parameter [100,101], while correlation-based techniques measure the strength and direction of linear relationships between inputs and outputs [102]. Elementary effects methods assess output sensitivity by systematically varying one input parameter at a time across its entire range and observing the resulting changes [103]. Importantly, GSA examines the effects of varying input

Lead City University DO NOT COPY

parameters uniformly and simultaneously over their full range [83,84]. This comprehensive perspective enables GSA to capture the overall impact of input uncertainties on model behaviour, making it a robust tool for identifying the primary sources of variability and for enhancing the reliability and robustness of complex models.

Because different GSA methods employ distinct characterisations, their results can sometimes be inconsistent and may not align with those produced by Local Sensitivity Analysis (LSA) approaches [94]. This inconsistency arises from the lack of a standardised definition for sensitivity coefficients across GSA techniques, which can occasionally lead to contradictory or non-comparable rankings of parameter importance [84]. Additionally, GSA results can be significantly affected by the choice of input parameter ranges and the assumed probability distributions of those inputs [82,93]. Despite these challenges, GSA remains a powerful tool for analysing both linear and nonlinear model responses and for uncovering interactions among multiple input parameters [104]. One of the most widely used GSA techniques is the variance-based method, often referred to as the Sobol' method. This approach decomposes the total variance V(Y) of the model output Y into fractions that can be attributed to individual input parameters or combinations of parameters [88]. The total variance in the Sobol' method is typically expressed as:

$$V(Y) = \sum_{i=1}^{k} V_i + \sum_{i<j} V_{ij} + \sum_{i<j<l} V_{ijl} + \cdots + V_{12\ldots k}$$

where V_i denotes the portion of the total variance attributable to the *i-th* input alone, V_{ij} represents the variance due to the interaction between the *i-th* and *j-th* inputs, and V_{ijl} accounts for the variance arising from interactions among the *i-th, j-th,* and *l-th* inputs. More generally, $V_{12\ldots k}$ refers to the variance associated with the interaction among all k inputs parameters.

Lead City University DO NOT COPY

2.1.11.3 Artificial Intelligence-Driven Sensitivity Analysis in SDN

Within the context of SDN, SA serves as an indispensable tool for network administrators, providing critical insights into how various network parameters and configurations affect overall network performance [92]. Such insights are vital for optimising network operations, troubleshooting technical challenges, and informing strategic decisions related to network upgrades and modifications [92]. The integration of AI, particularly ML techniques, has significantly influenced numerous domains, including SDN. In this domain, both LSA and GSA play key roles in elucidating the impact of different parameters on network behaviour and performance [104]. AI-driven SA, which harnesses ML methods, enables a more robust and comprehensive assessment of these relationships. This section provides an overview of recent approaches that employ ML-assisted LSA and GSA techniques within SDN environments.

2.1.11.3.1 Machine Learning-Assisted Local Sensitivity Analysis in SDN

One widely used approach for ML-assisted LSA involves gradient-based methods [105]. These methods estimate the gradient of a given performance metric with respect to specific input parameters, providing valuable information about local parameter influences. Techniques such as gradient boosting machines (GBMs) and neural networks are commonly employed to approximate these gradients efficiently [106,107]. In SDN environments, GBMs can effectively model the relationships between network parameters and performance metrics, enabling administrators to identify which parameters exert the greatest local impact on network performance [107]. Similarly, neural networks, particularly those using backpropagation, can be trained to estimate gradients of performance metrics with respect to input parameters [107]. This supports precise LSA and facilitates

Lead City University DO NOT COPY

the fine-tuning of individual network configurations in SDN deployments. When training neural networks for sensitivity analysis, various batch training algorithms may be applied, each offering distinct trade-offs in terms of speed, accuracy, and memory requirements [92]. Among these, the Levenberg–Marquardt algorithm is often regarded as one of the most efficient training methods for artificial neural networks (ANNs) due to its high speed and precision. However, its considerable memory requirements can pose limitations for certain applications [108,109].

The study presented in [92], employed the Levenberg–Marquardt algorithm to train an ANN that evaluated the sensitivity of SDN performance metrics specifically throughput, jitter, and response time under different distributed denial-of-service (DDoS) attack scenarios. The evaluation involved comparing predicted target values with actual measurements while introducing additive white Gaussian noise to the performance metrics. The results demonstrated that these SDN metrics are sensitive to DDoS attacks, highlighting the importance of robust training algorithms for accurate LSA. Furthermore, the balanced trade-off between speed and computational cost provided by the Levenberg–Marquardt algorithm is expected to benefit emerging 5G and future 6G networks, which demand higher data rates and ultra-reliable low-latency performance [110]. Achieving increased throughput, reduced latency, and enhanced capacity in these networks will require efficient real-time data processing algorithms, including advanced sensitivity analysis techniques for tuning network parameters in SDN environments [111].

Another common LSA technique is perturbation analysis, which involves introducing small changes to individual parameters and observing the resulting variations in performance metrics [112]. When combined with ML models, perturbation analysis can yield detailed insights into local parameter sensitivities [112]. For instance, support vector machines (SVMs) can model the boundary

Lead City University DO NOT COPY

conditions of network performance, allowing local sensitivity to be determined by systematically perturbing input parameters and assessing the changes in outputs[112]. Additionally, tree-based ML models such as Random Forests and GBMs can generate feature importance scores that indicate how specific parameters influence the model's predictions[113]. By examining these scores, network administrators can identify the most influential parameters at a local level, enabling them to prioritise targeted adjustments to optimise SDN performance[113].

2.1.11.3.2 Machine Learning-Assisted Global Sensitivity Analysis in SDN

Global Sensitivity Analysis (GSA) typically operates by simultaneously varying all input parameters across their entire range to assess their collective impact on model outputs [78]. Within the context of Software-Defined Networking (SDN), this comprehensive approach is vital for understanding the overall behaviour of the network and for pinpointing the most influential parameters affecting performance. By systematically exploring the complete parameter space, GSA provides a holistic view of how various inputs shape network behaviour, thereby supporting more informed decision-making and more effective optimisation strategies in SDN environments. Among the diverse methods within GSA, variance-based techniques such as the Sobol' method are especially prominent. These approaches decompose the variance of the output performance metrics into individual contributions from each input parameter [114]. When integrated with Machine Learning (ML) models, the effectiveness of variance-based methods can be significantly enhanced, as ML techniques can approximate complex variance components with high accuracy. For example, Sobol' indices, which quantify the share of output variance attributable to each input, can be efficiently computed using ML models such as neural networks and gradient boosting machines (GBMs), providing valuable insights for network optimisation[114].

Lead City University DO NOT COPY

Monte Carlo simulations also play a central role in GSA by enabling extensive sampling of the input space to estimate sensitivity indices[115]. Their computational efficiency can be further improved by employing ML techniques. For instance, by training predictive models on representative data subsets, it becomes possible to make accurate predictions over a much larger parameter space, thereby facilitating broader and more cost-effective GSA. In this regard, Gaussian processes are particularly useful for modelling the distribution of network performance metrics under varying parametric conditions [115]. Monte Carlo simulations performed on such surrogate models can yield a comprehensive global sensitivity profile [115]. Surrogate modelling approaches such as Kriging or Gaussian Process Regression are especially effective for approximating complex network performance landscapes [116]. By providing accurate estimates with significantly reduced computational cost, these models make it feasible to perform GSA extensively within SDN environments. Such techniques have proven to be highly practical, as demonstrated in [116].

A recent study presented in [117] illustrates an emerging approach in ML-assisted GSA, combining exploratory data analysis (EDA) and linear regression to characterise disparities in SDN performance metrics across emulated scenarios. This work demonstrates how changes in SDN scenarios, particularly under different distributed denial-of-service (DDoS) attack conditions, lead to significant fluctuations in performance metrics, revealing their sensitivity to the attack scenarios and highlighting interactions between various parameters. These findings underscore the value of ML-assisted GSA for identifying critical factors that affect network reliability and performance. By providing a clearer understanding of the global sensitivity of network performance to multiple input factors, GSA enables network administrators to prioritise configurations and allocate

Lead City University DO NOT COPY

resources effectively to maximise efficiency and reliability [118]. For contemporary 5G networks and the more demanding 6G networks of the future, ML-assisted GSA will be indispensable for managing increased system complexity and for ensuring that networks can meet the stringent requirements of emerging applications[119].

2.2 Methodological Review

This review critically examines and compares two major works that have significantly contributed to the methodological evolution of SDN security analysis [12] , which developed an intelligent, machine-learning-based framework integrating Artificial Neural Networks (ANNs), Local Sensitivity Analysis (LSA), and reactive mitigation strategies and [117], which proposed a low-cost, inference-driven statistical approach using Exploratory Data Analysis (EDA) and Regression-Based Sensitivity Analysis (RSA) . Reviewing these studies reveals both the strengths and limitations of earlier SDN analysis methodologies, thereby justifying the need for a more hybrid, statistically rigorous, and scalable approach namely the Design of Experiments, Generative Adversarial Networks, and Sensitivity Analysis (DOE-GAN-SA) framework.

Methodological Review of Design and Analysis of Anomaly Detection and Mitigation Schemes for Distributed Denial of Service Attacks in Software Defined Network.

The work by [12] employs a multifaceted methodology combining experimental emulation, machine learning, and sensitivity analysis to investigate, detect, and mitigate DDoS attacks in Software-Defined Networks (SDNs). The methodological approach integrates empirical testing with computational modelling, thereby addressing both the diagnostic and preventive dimensions of SDN security. The research begins with the emulation of an SDN testbed using Mininet, configured to mimic an enterprise network environment. The topology consists of multi-tier

Lead City University DO NOT COPY

OpenFlow switches managed by an OpenDaylight (Nitrogen) controller operating on an Ubuntu server, hosted within Oracle VirtualBox on an Intel Core i7 system with 16 GB RAM. Legitimate traffic is generated using iperf and ping, while attack traffic comprising SYN, ACK, and Slowloris floods is generated using tools such as hping3, Pentmenu, and LOIC[12]. The study incorporates three intensity levels (25%, 50%, 75%) for each attack type to simulate scalable threat conditions over a 360-second observation window per test scenario. Metrics including throughput, jitter, response time, and controller CPU utilisation are continuously logged.

The experimental process follows a three-phase lifecycle reconnaissance, attack execution, and mitigation. In the reconnaissance stage, Nmap and Zenmap are used to map host IPs and open ports, simulating the initial probing behaviour of an adversary. The attack execution stage leverages LOIC to flood the network, allowing the observation of degradation in throughput and controller performance. The final mitigation phase introduces a reactive flow-rule blocking mechanism, implemented within the SDN controller to dynamically isolate malicious flows. This procedural flow not only mirrors real-world cyberattack lifecycles but also enables systematic measurement of the network's recovery performance after mitigation[12].

For analytical processing, the study employs a Feed-Forward Artificial Neural Network (ANN) to model non-linear relationships among the performance metrics. The ANN is trained using normalised input data to classify network states as either normal or under attack. A Local Sensitivity Analysis (LSA) is performed through the injection of Additive White Gaussian Noise (AWGN) into each variable individually throughput, jitter, and response time to assess their respective influences on model accuracy. The Mean Squared Error (MSE) deviations are used as sensitivity indices to determine which metrics most affect detection performance. The findings

Lead City University DO NOT COPY

show that all three parameters exhibit sensitivity to noise perturbations, but jitter emerges as the most critical factor, suggesting its reliability as an early indicator of DDoS-induced network instability [12].

This methodology represents a hybrid experimental computational paradigm. The emulation-based experiments provide empirical grounding, while the ANN and LSA models introduce a data-driven analytical layer for pattern recognition and sensitivity quantification. The integration of attack simulation, detection, and mitigation establishes a closed-loop methodological cycle, moving beyond passive observation to active defence evaluation. The study also quantifies post-mitigation effects by monitoring CPU load and throughput restoration, allowing practical assessment of mitigation efficiency.

The main strength of [12] methodology lies in its inclusiveness it encompasses both the attack and response phases of cyber incidents within an SDN. Furthermore, the use of ANN provides the capacity to model complex, non-linear dependencies that traditional regression models might overlook. However, though statistically robust, is limited by its assumption of linearity, making it unsuitable for modelling non-linear interactions among SDN parameters. The linear regression framework does not account for complex dependencies that emerge under adaptive and evolving attack scenarios.

Methodological Review of Behavioral Study of Software-Defined Network Parameters Using Exploratory Data Analysis and Regression-Based Sensitivity Analysis

The study by[117] presents a statistically grounded, experiment-driven methodology for understanding the behavioural characteristics of SDNs under DDoS attack scenarios. The

Lead City University DO NOT COPY

methodological foundation is built upon network emulation, quantitative data analysis, and inferential statistical modelling, with the central goal of providing an inference-driven assessment of SDN performance metrics. The authors employ a custom SDN topology designed and emulated within the Mininet environment. This setup replicates a realistic network infrastructure, comprising ten OpenFlow switches, sixteen hosts, and a Floodlight controller operating on Ubuntu servers. The experiment runs on a 32 GB RAM Xeon processor, ensuring sufficient computational resources for simultaneous attack simulation and data collection. Legitimate traffic is generated using the iperf and ping utilities, while malicious flooding is introduced via the Low Orbit Ion Cannon (LOIC) tool to simulate HTTP, TCP, and UDP attack conditions [117]. Each flood scenario is executed for 15 minutes, producing datasets that reflect the dynamic interplay between throughput, jitter, and response time under stress.

Data extraction and pre-processing are performed using KNIME, an open-source graphical data-mining platform. The methodology proceeds in two analytical phases: EDA and RSA. The EDA phase employs descriptive statistics, histograms, probability density functions (PDFs), covariance, and correlation matrices to visualise changes in SDN performance metrics across four operational states normal, HTTP flood, TCP flood, and UDP flood. This descriptive layer reveals distinct distributional shifts: for instance, throughput declines markedly under TCP and HTTP floods, while jitter peaks during UDP floods. These visual patterns form the empirical baseline for subsequent inferential modelling[117].

In the RSA phase, the authors introduce a novel Adaptive Univariate Response Value (AURV) to represent the aggregated response of the SDN system. Each metric throughput, jitter, and response time is first standardised using Z-score normalisation, and the AURV serves as the dependent

Lead City University DO NOT COPY

variable in a multiple linear regression model implemented via MATLAB's fitlm function. The model demonstrates strong explanatory power, with a coefficient of determination ($R^2 = 0.926$) and a low root mean square error (RMSE = 0.235), confirming the reliability of the statistical relationship between SDN parameters and overall network behaviour. The regression coefficients and their associated p-values (all $\ll 0.05$) indicate statistically significant relationships between metrics, revealing response time as the most sensitive indicator and the throughput–response time interaction as the strongest combined influence [117].

The integration of EDA and RSA provides a robust dual-layer analytical approach EDA facilitates pattern recognition and anomaly identification, while RSA quantifies metric sensitivity and interaction effects. This hybrid analytical design exemplifies a balance between interpretability and inferential precision. The methodology is also computationally efficient, with EDA and RSA processes completing within seconds on standard hardware, suggesting potential for real-time deployment in SDN monitoring frameworks.

Nevertheless, the study employs machine learning (ANN) but focuses on local sensitivity and single-dimensional perturbations, which restricts its ability to model high-dimensional interactions or to generalise across unseen network states. Moreover, the ANN model depends heavily on manually collected data, which limits scalability and the exploration of rare or zero-day attack conditions.

2.3 Related Works

Software-Defined Networking (SDN) has become a foundational paradigm for supporting the scalability of heterogeneous Internet of Things (IoT) environments, primarily due to its architectural separation of the control and data planes. Despite this advantage, much of the current

Lead City University DO NOT COPY

research still relies extensively on heuristic-based parameter tuning, often neglecting the incorporation of adaptive feedback mechanisms that connect anomaly detection with flow-rule reconfiguration[120]. Early work such as [121] demonstrated the feasibility of deploying SDN for vehicular IoT systems using the multipurpose infrastructure for network applications (MINA)-SDN controller. However, their approach did not adequately consider critical challenges such as sensor energy consumption and the integration of embedded security mechanisms. Subsequent contributions, including the survey by [122], stressed the importance of slice isolation and controller scalability for supporting large-scale IoT networks. Similarly, [123] proposed software-defined APIs for smart city infrastructures, facilitating shared use of gateways and cloud services with supporting quantitative evaluations. More recently, [124] introduced a multi-tier architecture that significantly reduced packet loss in smart home deployments. Extending this trajectory, a meta-analysis of over 160 studies by [125] identified micro-service–based controllers, flow table compression, and energy-aware routing as essential for ensuring secure and efficient SDN-based IoT systems. Collectively, these works converge on programmability, slice isolation, and controller scalability as central strategies for secure IoT infrastructures. However, none directly address the challenges of data-scarce sensitivity analysis (SA) or integrate anomaly detection and classification (ADC) mechanisms within SDN environments a gap this thesis aims to address through the proposed DOE-GAN-SA framework.

Parallel research into Generative Adversarial Networks (GANs) reveals promising, though underexplored, potential for their application in SA and ADC within SDN-IoT networks. Since their introduction [75], GANs have demonstrated considerable capacity for producing realistic synthetic datasets, including network traffic [126,127], thereby enabling controlled experimentation on

Lead City University DO NOT COPY

system behaviours. Nevertheless, these applications rarely address SA or ADC directly [128]. More recent developments highlight GANs' effectiveness in augmenting datasets for high-dimensional systems, ultimately improving classification accuracy [129]. Despite such advances, their adoption for SA in SDN-IoT environments remains at a nascent stage. Conventional SA methods such as variance-based approaches and Monte Carlo simulations [130, 131], although theoretically robust, are computationally demanding and impractical in latency-sensitive IoT contexts [132,133]. Variance-based SA, for example, provides valuable insights into factor contributions to system variance [134,135] but becomes infeasible when handling complex interactions across numerous parameters [136]. Researchers have therefore explored sample-efficient alternatives such as Latin Hypercube Sampling (LHS), which ensures broader coverage of the parameter space and reduces computational demands [137,138,139]. However, LHS alone does not sufficiently account for correlations among input variables [138, 139,117], potentially reducing accuracy in SDN-IoT settings where such correlations are common.

Combining LHS with GANs presents a novel pathway for overcoming these limitations. GAN-generated datasets can preserve the statistical properties of real network traffic, while LHS ensures structured coverage of the parameter space [140]. Together, these complementary techniques offer a hybrid approach to SA, capable of addressing complexity, high dimensionality, and real-time constraints. While GANs have demonstrated success in diverse synthetic data applications [70,141], their integration with LHS for SA in SDN-reliant IoT systems remains largely unexplored. Existing SA frameworks, though effective, are often specialised and lack extensibility to tasks such as data augmentation or integrated anomaly detection. These limitations underscore the necessity of innovative hybrid approaches that combine Design of Experiments (DOE) principles

Lead City University DO NOT COPY

with machine learning, offering scalability, precision, and broader applicability. The DOE-GAN-SA framework proposed in this work seeks to fill this methodological gap by uniting DOE-driven sampling with GAN-based synthetic data generation for robust ADC and SA in SDN-IoT systems.

Recent surveys have classified detection methods into signature-based, anomaly-based, and hybrid categories [142]. Signature-based models excel in recognising known patterns but fail against emerging variants, while anomaly-based systems adapt to novel threats but often suffer high false-positive rates. Hybrid methods combine both strengths but are frequently constrained by the limited availability of comprehensive datasets. Furthermore, mitigation strategies such as filtering, traceback, and scrubbing offer partial solutions but demonstrate limited scalability when confronted with hyper-volumetric attack traffic, highlighting the need for more adaptive, AI-driven defences[142]. Research focusing specifically on SDN has underscored the vulnerability of controllers to saturation and packet-in flooding [143]. Here, machine learning algorithms including support vector machines (SVMs), decision trees, and deep neural networks have achieved detection accuracies exceeding 95% in controlled environments, though scalability and generalisation remain problematic due to reliance on small-scale emulation platforms such as Mininet. Complementary evaluations by [144] using benchmark datasets (e.g., CICDDoS2019, UNSW-NB15) reported that ensemble and hybrid deep learning models achieved detection rates above 99% with false positive rates below 1%. Nonetheless, challenges in feature engineering, computational cost, and real-time deployment persist. To overcome these, researchers propose approaches such as transfer learning, online learning, and explainable AI to address dynamic adversarial behaviours and maintain operational trust.

Lead City University DO NOT COPY

Taken together, the literature illustrates that although SDN and GAN research provide innovative pathways for anomaly detection and sensitivity analysis, the persistent threat of DDoS attacks underscores the urgency for scalable, data-driven solutions. Across both domains, a recurring research gap is evident: the absence of integrated frameworks that combine synthetic data generation, adaptive anomaly detection, and resilience against large-scale adversarial conditions. The DOE-GAN-SA framework presented in this thesis responds directly to these gaps by integrating LHS-driven DOE with GAN-based synthetic traffic generation, thereby facilitating both SA and ADC while simultaneously enhancing resilience against advanced attacks such as DDoS in SDN-IoT contexts. This hybrid methodology not only improves the robustness of SA but also provides a scalable and resource-efficient solution for modern, intelligent network infrastructures.

2.4 Summary of Gaps in Related Works

The reviewed literature establishes SDN as a core enabler of scalable IoT systems, with programmability, slice isolation, and controller scalability consistently identified as critical factors. However, most existing studies rely on heuristic tuning and do not integrate adaptive feedback mechanisms for anomaly detection or classification within SDN architectures. Parallel research on GANs highlights their potential in synthetic data generation and dataset augmentation but shows limited application in supporting sensitivity analysis (SA) or anomaly detection in SDN-IoT environments. Traditional SA techniques, while effective, are computationally prohibitive for real-time IoT systems, and alternatives such as Latin Hypercube Sampling (LHS) only partially resolve these limitations.

Lead City University DO NOT COPY

Additionally, research on DDoS detection underscores the vulnerabilities of SDN and IoT infrastructures, with machine learning and deep learning models achieving high detection accuracy in controlled settings but facing scalability and generalisation challenges in real-world deployments. Across both domains, the literature reveals a common gap: the absence of integrated frameworks that combine synthetic data generation, adaptive anomaly detection, and robust SA for resilient, large-scale networks. This thesis directly addresses these gaps through the proposed DOE-GAN-SA framework, which unifies LHS-based sampling with GAN-driven synthetic data generation to deliver scalable, data-efficient, and attack-resilient analysis tailored for SDN-IoT systems.

Lead City University DO NOT COPY

Endnotes

1. D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky & S. Uhlig, *Software-Defined Networking: A Comprehensive Survey.* **Proceedings of the IEEE** 103(1), 2014, 14–76.

2. M. Kaloudis, *Evolving Software Architectures from Monolithic Systems to Resilient Microservices: Best Practices, Challenges and Future Trends.* **International Journal of Advanced Computer Science & Applications**, 15(9), 2024.

3. N. Gupta, M. S. Maashi, S. Tanwar, S. Badotra, M. Aljebreen & S. Bharany, *A Comparative Study of Software Defined Networking Controllers using Mininet.* **Electronics**, 11(17), 2022, 2715.

4. T. Darwish, T. A. Alhaj & F. A. Elhaj, *Controller placement in software defined emerging networks: a review and future directions.* **Telecommunication Systems**, 88(1), 2025, 18.

5. Open Networking Foundation, *OpenFlow Switch Specification Version 1.5.1*. 2025. Available online: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf.

6. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker & J. Turner, *OpenFlow: Enabling Innovation in Campus Networks.* **ACM SIGCOMM Computer Communication Review** 38(2), 2008, 69–74.

7. Open Networking Foundation, *Homepage*. 2025. Available online: https://www.opennetworking.org.

8. R. Wazirali, R. Ahmad & S. Alhiyari, *SDN-openflow topology discovery: An overview of performance issues*. **Applied Sciences**, 11(15), 2021, 6999.

Lead City University DO NOT COPY

9. M. Hussain, N. Shah, R. Amin, S. S. Alshamrani, A. Alotaibi & S. M. Raza, *Software-defined networking: Categories, analysis, and future directions.* **Sensors**, 22(15), 2022, 5551.

10. N. H. Hussein, S. P. Koh, C. T. Yaw, S. K. Tiong, F. Benedict, T. Yusaf & T. C. Hong, *SDN-based VANET routing: A comprehensive survey on architectures, protocols, analysis, and future challenges.* **IEEE Access,** 13, 2024, 126801-126861

11. J. Singh & S. Behal, *Detection and Mitigation of DDoS Attacks in SDN: A Comprehensive Review, Research Challenges, and Future Directions.* **Computer Science Review,** 37(100279), 2020, 1-15.

12. A. O. Sangodoyin, *Design and Analysis of Anomaly Detection and Mitigation Schemes for Distributed Denial of Service Attacks in Software-Defined Networks: An Investigation into the Security Vulnerabilities of Software-Defined Networks and the Design of Efficient Detection and Mitigation Techniques for DDoS Attack Using Machine Learning Techniques*. **Doctoral diss.**, **University of Bradford**, 2019. Available online: https://bradscholars.brad.ac.uk/entities/publication/c64312ab-f49c-4bf3-97b8-64221b672b90.

13. M. Najafimehr, S. Zarifzadeh & S. Mostafavi, *DDoS Attacks and Machine-Learning-Based Detection Methods: A Survey and Taxonomy*. **Engineering Reports,** 5(1), 2023.

14. A. K. Jain, H. Shukla & D. Goel, *A Comprehensive Survey on DDoS Detection, Mitigation, and Defense Strategies in Software-Defined Networks*. **Cluster Computing,** 2024, 1–30. Available online: https://doi.org/10.1007/s10586-024-04596-z.

15. C. Singh & A. K. Jain, *A Comprehensive Survey on DDoS Attacks Detection & Mitigation in SDN-IoT Network.* **e-Prime - Advances in Electrical Engineering, Electronics and Energy,** 8, 2024.

16. S. Gao, Z. Peng, B. Xiao, A. Hu, Y. Song & K. Ren, *Detection and Mitigation of DoS Attacks in Software Defined Networks*. **IEEE/ACM Transactions on Networking,** 28(3), 2020, 1419–1433.

17. J. Wang, L. Wang & R. Wang, *A Method of DDoS Attack Detection and Mitigation for the Comprehensive Coordinated Protection of SDN Controllers.* **Entropy,** 25(8), 2023, 1210. Available online: https://doi.org/10.3390/e25081210.

18. B. Aryal, R. Abbas & I. B. Collings, *SDN Enabled DDoS Attack Detection and Mitigation for 5G Networks.* **Journal of Communications,** 16(7), 2021, 267–275. Available online: https://doi.org/10.12720/jcm.16.7.267-275.

19. Y. Wang, T. Hu, G. Tang, J. Xie & J. Lu, *SGS: Safe-Guard Scheme for Protecting Control Plane Against DDoS Attacks in Software-Defined Networking.* **IEEE Access,** 7, 2019): 34699–34710. Available online: https://doi.org/10.1109/ACCESS.2019.2895092.

Leadcity University DO NOT COPY

20. S. Mahrarch & A. Haqiq, *DDoS Flooding Attack Mitigation in Software Defined Networks.* **International Journal of Advanced Computer Science and Applications (IJACSA),** 11, 1 (2020).

21. T. E. Ali, Y.-W. Chong & S. Manickam, *Machine Learning Techniques to Detect a DDoS Attack in SDN: A Systematic Review.* **Applied Sciences,** 13, 5 (2023): 3183. Available online: https://doi.org/10.3390/app13053183.

22. D. Karki & B. R. Dawadi, *Machine Learning based DDoS Detection System in Software-Defined Networking.* In **Proceedings of 11th IOE Graduate Conference,** 228–236. 2022.

23. S. Brief, *SDN Security Considerations in the Data Center.* 2013.

24. C. Ezechi, M. O. Akinsolu, A. O. Sangodoyin, F. T. Akinsolu & W. Sakpere. *Software-Defined Networking in Cyber-Physical Systems*. In **Cyber Physical System 2.0: Communication and Computational Technologies**, 2024, 44.

25. A. Voellmy, J. Kim, S. Feamster & N. Feamster. *Procera: A Language for High-Level Reactive Network Control.* In **Proceedings of the First Workshop on Hot Topics in Software Defined Networking**, 2012, 43–48.

26. N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown & S. Shenker, *NOX: Towards an Operating System for Networks.* **ACM SIGCOMM Computer Communication Review,** 38(3), 2008, 105–110.

27. O. Coker & S. Azodolmolky, *Software-Defined Networking with OpenFlow: Deliver Innovative Business Solutions.* **Birmingham, UK: Packt Publishing Ltd**, 2017.

28. N. Feamster, J. Rexford & E. Zegura. *The Road to SDN: An Intellectual History of Programmable Networks.* **ACM SIGCOMM Computer Communication Review,** 44(2), 2014, 87–98.

29. M. B. Al-Somaidai & E. B. Yahya, *Survey of Software Components to Emulate OpenFlow Protocol as an SDN Implementation.* **American Journal of Software Engineering and Applications** 3(6), 2014, 74–82.

30. K. Benton, L. J. Camp & C. Small, *OpenFlow Vulnerability Assessment.* In **Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking**, 2013, 151–152.

31. S. Zerkane, D. Espes, P. Le Parc & F. Cuppens, *Vulnerability Analysis of Software Defined Networking*. In **International Symposium on Foundations and Practice of Security**, 2016, 97–116.

32. N. Dayal, P. Maity, S. Srivastava & R. Khondoker, *Research Trends in Security and DDoS in SDN*. **Security and Communication Networks,** 9(18), 2016, 6386–6411.

Lead City University DO NOT COPY

33. G. Eason, R. Kloti, V. Kotronis & P. Smith, *OpenFlow: A Security Analysis*. In **Proceedings of the IEEE International Conference on Network Protocols (ICNP)**, 2013.

34. P. Manso, J. Moura & C. Serrão, *SDN-based Intrusion Detection System for Early Detection and Mitigation of DDoS Attacks.* **Information***,* 10(3), 2019, 106.

35. J. Zheng, Q. Li, G. Gu, J. Cao, D. K. Yau & J. Wu, *Realtime DDoS Defense Using COTS SDN Switches via Adaptive Correlation Analysis.* **IEEE Transactions on Information Forensics and Security,** 13(7), 2018, 1838–1853.

36. Y. Xu & Y. Liu, *DDoS Attack Detection Under SDN Context*. In **Proceedings of IEEE INFOCOM 2016 – The 35th Annual IEEE International Conference on Computer Communications**, 2016, 1–9.

37. S. Hayward, G. O'Callaghan & S. Sezer, *SDN Security: A Survey.* In **2013 IEEE SDN for Future Networks and Services (SDN4FNS)**, 2013,1–7.

38. Z. Liu, R. H. Campbell & M. Mickunas, *Active Security Support for Active Networks*. **IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews** 33(4), 2003, 432–445.

39. S. Shin, P. Porras, V. Yegneswaran & G. Gu, *A Framework for Integrating Security Services into Software-Defined Networks.* In **Open Networking Summit**, 2013.

40. X. Wen, Y. Chen, C. Hu, C. Shi & Y. Wang, *Towards a Secure Controller Platform for OpenFlow Applications.* In **Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking,** 2013, 171–172.

41. S. Hartman, M. Wasserman & D. Zhang, *Security Requirements in the Software-Defined Networking Model.* **IETF Internet-Draft draft-hartman-sdnsec-requirements-01**, 2013.

42. H. Xie, T. Tsou, D. Lopez, H. Yin & V. Gurbani, *Use Cases for ALTO with Software-Defined Networks*. **IETF Internet-Draft draft-xie-alto-sdn-extension-use-cases-01,** 2012.

43. I. Ahmad, S. Namal, M. Ylianttila, & A. Gurtov, *Security in Software Defined Networks: A Survey.* **IEEE Communications Surveys & Tutorials** 17(4), 2015, 2317–2346.

44. J. Naous, D. Erickson, G. A. Covington, G. Appenzeller & N. McKeown, *Implementing an OpenFlow Switch on the NetFPGA Platform.* **In Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems,** 2008, 1–9.

45. S. Shin, V. Yegneswaran, P. Porras & G. Gu, *Avant-Guard: Scalable and Vigilant Switch Flow Management in Software-Defined Networks.* In **Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security**, 2013, 413–424.

Lead City University DO NOT COPY

46. S. Shin & G. Gu, *Attacking Software-Defined Networks: A First Feasibility Study.* In **Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking**, 2013, 165–166.

47. P. Fonseca, R. Bennesby, E. Mota & A. Passito, *A Replication Component for Resilient OpenFlow-Based Networking.* In **2012 IEEE Network Operations and Management Symposium**, 2012, 933–939.

48. G. Yao, J. Bi & L. Guo, *On the Cascading Failures of Multi-Controllers in Software Defined Networks*. In **2013 21st IEEE International Conference on Network Protocols (ICNP),** 2013, 1–2.

49. International Telecommunication Union – Telecommunication Standardization Sector (ITU-T), *Security Architecture for Systems Providing End-to-End Communications.* **ITU-T Recommendation X.805***,* 2003.

50. I. A. Ahmad, S. Namal, M. Ylianttila & A. Gurtov, *Security in Software Defined Networks: A Survey.* **IEEE Communications Surveys & Tutorials** 17(4), 2015, 2317–2346.

51. P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson & G. Gu, *A Security Enforcement Kernel for OpenFlow Networks.* In **Proceedings of the First Workshop on Hot Topics in Software-Defined Networks,** 2012, 121–126.

52. T. Hinrichs, N. Gude, M. Casado, J. Mitchell & S. Shenker, *Expressing and Enforcing Flow-Based Network Security Policies*. **University of Chicago Technical Report,** 9, 2008.

53. S. Namal, I. A. Ahmad, A. Gurtov & M. Ylianttila, *Enabling Secure Mobility with OpenFlow*. In **2013 IEEE SDN for Future Networks and Services (SDN4FNS)**, 2013, 1–5.

54. N. Gutz, A. Story, C. Schlesinger & N. Foster, *Splendid Isolation: A Slice Abstraction for Software-Defined Networks*. In **Proceedings of the First Workshop on Hot Topics in Software-Defined Networks,** 2012, 79–84.

55. A. Khurshid, X. Zou, W. Zhou, M. Caesar & P. B. Godfrey, *VeriFlow: Verifying Network-Wide Invariants in Real Time.* In **Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI '13),** 2013, 15–27.

56. C.-Y. Huang, M.-C. Tang, Y.-T. Chen, Y.-C. Cheng & Y.-R. Chang, A Novel Design for Future On-Demand Service and Security. In **2010 IEEE 12th International Conference on Communication Technology,** 2010, 385–388.

57. J. H. Jafarian, E. Al-Shaer & Q. Duan, *OpenFlow Random Host Mutation: Transparent Moving Target Defense Using Software-Defined Networking*. In **Proceedings of the First Workshop on Hot Topics in Software-Defined Networks,** 2012, 127–132.

58. M. Santos, B. de Oliveira, C. Margi, B. N. Astuto, T. Turletti & K. Obraczka. *Software-Defined Networking Based Capacity Sharing in Hybrid Networks*. 2013.

Leadcity University DO NOT COPY

59. K. Phemius, M. Bouet & J. Leguay, *Disco: Distributed Multi-Domain SDN Controllers*. In **2014 IEEE Network Operations and Management Symposium (NOMS),** 2014, 1–4.

60. A. Voellmy & J. Wang, *Scalable Software Defined Network Controllers.* **ACM SIGCOMM Computer Communication Review,** 42(4), 2012, 289–290.

61. E. Ng, Z. Cai & A. Cox, *Maestro: A System for Scalable OpenFlow Control.* **Rice University Technical Report TR10-08**, 2010.

62. S. W. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu & M. Tyson, *FRESCO: Modular Composable Security Services for Software-Defined Networks.* In **20th Annual Network and Distributed System Security Symposium (NDSS)**, 2013.

63. A. K. Nayak, A. Reimers, N. Feamster & R. Clark, *Resonance: Dynamic Access Control for Enterprise Networks*. In **Proceedings of the 1st ACM Workshop on Research on Enterprise Networking,** 2009, 11–18.

64. J. Naous, R. Stutsman, D. Mazières, N. McKeown & N. Zeldovich, *Delegating Network Security with More Information*. In **Proceedings of the 1st ACM Workshop on Research on Enterprise Networking,** 2009, 19–26.

65. T. Dierks. *The Transport Layer Security (TLS) Protocol Version 1.2.* 2008.

66. A. A. Alashhab, M. S. M. Zahid, M. A. Azim, M. Y. Daha, B. Isyaku & S. Ali, *A Survey of Low Rate DDoS Detection Techniques Based on Machine Learning in Software-Defined Networks.* **Symmetry** 14(1563), 2022, 1–30. Available online: https://doi.org/10.3390/sym14081563

67. J. Wang & L. Wang, "SDN-Defend: *A Lightweight Online Attack Detection and Mitigation System for DDoS Attacks in SDN*. **Sensors,** 22(21), 2022, 8287. Available online: https://doi.org/10.3390/s22218287

68. D. Karki & B. R. Dawadi, *Machine Learning based DDoS Detection System in Software-Defined Networking*. In **Proceedings of the 11th IOE Graduate Conference,** 2022, 228–236.

69. L. Girish et al., *DadGAN: DDoS Anomaly Detection using Generative Adversarial Network.* In **IEEE Symposium on Security and Privacy Workshops,** 2021, 1–7.

70. M. Goyal & Q. H. Mahmoud, *A Systematic Review of Synthetic Data Generation Techniques Using Generative AI.* **Electronics,** 13(17), 2024, 3509. Available online: https://doi.org/10.3390/electronics13173509

71. E. Ayanoglu, K. Davaslioglu & Y. E. Sagduyu, *Machine Learning in NextG Networks via Generative Adversarial Networks.* **IEEE Transactions on Cognitive Communications and Networking,** 8(2), 2022, 480–502.

Lead City University DO NOT COPY

72. M. Sabuhi, M. Zhou, , C. P. Bezemer & P. Musilek, *Applications of Generative Adversarial Networks in Anomaly Detection: A Systematic Literature Review.* **IEEE Access,** 9, 2023, 161003-161029.

73. A. O. Sangodoyin, M. O. Akinsolu, P. Pillai & V. Grout, *Detection and classification of DDoS flooding attacks on software-defined networks: A case study for the application of machine learning*. **IEEE Access**, 9, 2021, 122495-122508.

74. N. S. Musa, N. M. Mirza, S. H. Rafique, A. M. Abdallah & T. Murugan, M*achine Learning and Deep Learning Techniques for Distributed Denial of Service Anomaly Detection in Software-Defined Networks—Current Research Solutions.* **IEEE Access**, 12, 2024, 17982-18011.

75. I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville & Y. Bengio, *Generative Adversarial Nets.* In **Advances in Neural Information Processing Systems,** 27, 2014, 2672–2680.

76. K. N. Qureshi et al., *A Distributed Software Defined Networking Model to Improve the Scalability and Quality of Services for Flexible Green Energy Internet for Smart Grid Systems*. **Computers & Electrical Engineering,** 84, 2020, 106634.

77. T. G. Trucano et al., *Calibration, Validation, and Sensitivity Analysis: What's What*. **Reliability Engineering & System Safety,** 91, 2006, 1331–1357.

78. B. Iooss & P. Lemaître, *A Review on Global Sensitivity Analysis Methods*. In **Handbook of Uncertainty Quantification**, 2014, 101–122.

79. A. Saltelli, *Sensitivity Analysis for Importance Assessment*. **Risk Analysis,** 22, 2002, 579–590.

80. A. Saltelli, S. Tarantola, F. Campolongo & M. Ratto, *Why So Many Published Sensitivity Analyses Are False. A Systematic Review of Sensitivity Analysis Practices*. **Environmental Modelling & Software,** 114, 2019, 29–39.

81. M. Ionescu-Bujor, & D. G. Cacuci. *A Comparative Review of Sensitivity and Uncertainty Analysis of Large-Scale Systems - I: Deterministic Methods.* **Nuclear Science and Engineering,** 147, 2004, 189–203.

82. D. G. Cacuci & M. Ionescu-Bujor, *A Comparative Review of Sensitivity and Uncertainty Analysis of Large-Scale Systems - II: Statistical Methods.* **Nuclear Science and Engineering,** 147, 2004, 204–217.

83. B. Iooss & P. Lemaître, *A Review on Global Sensitivity Analysis Methods*. In **Uncertainty Management in Simulation-Optimization of Complex Systems**, 2015, 101–122.

84. A. Saltelli et al., *Global Sensitivity Analysis: The Primer*. **West Sussex, UK: John Wiley & Sons**, 2008.

Peace City University DO NOT COPY

85. T. Santner, B. Williams & W. Notz, *The Design and Analysis of Computer Experiments*. **New York, NY, USA: Springer**, 2003. Available online: https://doi.org/10.1007/978-1-4757-3799-8

86. U. Umoga et al., *Exploring the Potential of AI-Driven Optimization in Enhancing Network Performance and Efficiency.* **Magna Scientia Advanced Research and Reviews,** 10, 2024, 368–378.

87. S. Razavi et al., *The Future of Sensitivity Analysis: An Essential Discipline for Systems Modeling and Policy Support*. **Environmental Modelling & Software,** 137, 2020, 104954.

88. A. Saltelli et al., *Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models*. **West Sussex, UK: Wiley**, 2004.

89. Z. Pang, Z. O'Neill, Y. Li, & F. Niu, *The Role of Sensitivity Analysis in the Building Performance Analysis: A Critical Review.* **Energy and Buildings**, 209, 2020, 109659.

90. J. M. Zurada, A. Malinowski & I. Cloete, *Sensitivity Analysis for Minimization of Input Data Dimension for Feedforward Neural Network.* In **Proceedings of IEEE International Symposium on Circuits and Systems-ISCAS94**, 6, 1994, 447–450.

91. C. Rojas et al., *A Critical Look at Efficient Parameter Estimation Methodologies of Electrochemical Models for Lithium-Ion Cells*. **Journal of Energy Storage,** 80, 2024, 110384.

92. A. O. Sangodoyin et al., *A Deductive Approach for the Sensitivity Analysis of Software-Defined Network Parameters.* **Simulation Modelling Practice and Theory,** 103, 2020, 102099.

93. D. G. Cacuci & M. Ionescu-Bujor, *Sensitivity and Uncertainty Analysis, Data Assimilation, and Predictive Best-Estimate Model Calibration.* In **Handbook of Nuclear Engineering**, 2010, 1913-2051

94. S. Razavi & H. V. Gupta, *What Do We Mean by Sensitivity Analysis? The Need for Comprehensive Characterization of Global Sensitivity in Earth and Environmental Systems Models.* **Water Resources Research,** 51, 2015, 3070–3092.

95. A. Hunter, L. Kennedy, J. Henry & I. Ferguson, *Application of Neural Networks and Sensitivity Analysis to Improved Prediction of Trauma Survival.* **Computer Methods and Programs in Biomedicine,** 62(1), 2000, 11–19.

96. J. Kirch et al., *The Effect of Model Rescaling and Normalization on Sensitivity Analysis on an Example of a MAPK Pathway Model.* **EPJ Nonlinear Biomedical Physics,** 4, 2016. Available online: https://doi.org/10.1140/epjnbp/s40366-016-0030-z

97. J. M. Zurada, A. Malinowski & S. Usui, *Perturbation Method for Deleting Redundant Inputs of Perceptron Networks*. **Neurocomputing,** 14(2), 1997, 177–193.

Lead City University DO NOT COPY

98. P. A. Kowalski & M. Kusy, *Sensitivity Analysis for Probabilistic Neural Network Structure Reduction.* **IEEE Transactions on Neural Networks and Learning Systems,** 29(5), 2017, 1919–1932.

99. A. Saltelli & P. Annoni, *How to Avoid a Perfunctory Sensitivity Analysis.* **Environmental Modelling & Software,** 25(12), 2010, 1508–1517.

100. I. M. Sobol, *Global Sensitivity Analysis for Nonlinear Mathematical Models and Their Monte Carlo Estimates.* **Mathematics and Computers in Simulation,** 55(1–3), 2001, 271–280.

101. R. Cukier, H. Levine & K. Shuler, *Nonlinear Sensitivity Analysis of Multiparameter Model Systems.* **Journal of Computational Physics** 26 (1978): 1–42.

102. J. C. Helton, *Uncertainty and Sensitivity Analysis Techniques for Use in Performance Assessment for Radioactive Waste Disposal.* **Reliability Engineering & System Safety,** 42, 1993, 327–367.

103. M. D. Morris, *Factorial Sampling Plans for Preliminary Computational Experiments.* **Technometrics,** 33(2), 1991, 161–174.

104. K. G. Link et al., *A Local and Global Sensitivity Analysis of a Mathematical Model of Coagulation and Platelet Deposition Underflow*. **PLoS ONE** 13(7), 2018, e0200917.

105. C. Qin et al., *Comparative Study of Global Sensitivity Analysis and Local Sensitivity Analysis in Power System Parameter Identification.* **Energies,** 16, 2023, 5915.

106. J. H. Friedman., *Greedy Function Approximation: A Gradient Boosting Machine.* **Annals of Statistics,** 2001, 1189–1232.

107. I. Goodfellow, Y. Bengio & A. Courville, *Deep Learning*. **MIT Press**, 2016.

108. M. T. Hagan, & M. B. Menhaj. *Training Feedforward Networks with the Marquardt Algorithm.* **IEEE Transactions on Neural Networks** 5, 6 (1994): 989–993.

109. H. Yu & B. M. Wilamowski, *Levenberg-Marquardt Training.* In **Industrial Electronics Handbook,** 5(12), 2011, 1–16.

110. Y. Liu et al., *Machine Learning for 6G Enhanced Ultra-Reliable and Low-Latency Services.* **IEEE Wireless Communications** 30(2), 2023, 48–54. Available no: https://doi.org/10.1109/MWC.006.2200407

111. I. Salah et al., *Comparative Study of Efficiency Enhancement Technologies in 5G Networks - A Survey*. **Procedia Computer Science,** 182, 2021, 150–158.

112. C. Cortes & V. Vapnik, Support-Vector Networks, **Machine Learning** 20(3), 1995, 273–297.

Lead City University DO NOT COPY

113. L. Breiman, *Random Forests.* **Machine Learning** 45(1), 2001, 5–32.

114. I. M. Sobol, *Sensitivity Analysis for Nonlinear Mathematical Models*. **Mathematical Modeling and Computational Experiment,** 1(4), 1993, 407–414.

115. C. E. Rasmussen & C. K. I. Williams, *Gaussian Processes for Machine Learning*. **MIT Press**, 2006.

116. T. J. Santner, B. J. Williams & W. I. Notz, *The Design and Analysis of Computer Experiments*. **Springer**, 2018.

117. M. O. Akinsolu et al., *Behavioral Study of Software-Defined Network Parameters Using Exploratory Data Analysis and Regression-Based Sensitivity Analysis.* **Mathematics** 10, 2022, 2536.

118. N. Khan., *5G Network: Techniques to Increase Quality of Service and Quality of Experience.* **International Journal of Computer Networks and Applications,** 9, 2022, 476.

119. X. You et al., *Towards 6G Wireless Communication Networks: Vision, Enabling Technologies, and New Paradigm Shifts.* **Science China Information Sciences,** 64, 2021, 110301. Available online: https://doi.org/10.1007/s11432-021-3330-1

120. S. Y. Shin, S. Nejati, M. Sabetzadeh, L. C. Briand, C. Arora & F. Zimmer, *Dynamic Adaptation of Software-Defined Networks for IoT Systems: A Search-Based Approach.* In **Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS),** 2020, 137–148.

121. Z. Qin, G. Denker, C. Giannelli, P. Bellavista & N. Venkatasubramanian, *A Software-Defined Networking Architecture for the Internet of Things.* In **Proceedings of the 2014 IEEE Network Operations and Management Symposium (NOMS)**, 2014, 1–9.

122. N. Bizanis & F. A. Kuipers, *SDN and Virtualization Solutions for the Internet of Things: A Survey.* **IEEE Access,** 4, 2016, 5591–5606.

123. J. Liu, Y. Li, M. Chen, W. Dong & D. Jin, *Software-Defined Internet of Things for Smart Urban Sensing*. **IEEE Communications Magazine,** 53(9), 2015, 55–63.

124. S. M. M. Gilani, M. Usman, S. Daud, A. Kabir, Q. Nawaz & O. Judit, *SDN-Based Multi-Level Framework for Smart Home Services*. **Multimedia Tools and Applications,** 83, 2024, 327–347.

125. R. Chaudhary, G. S. Aujla, N. Kumar & P. K. Chouhan, *A Comprehensive Survey on Software-Defined Networking for Smart Communities*. **International Journal of Communication Systems,** 38, 2025, e5296.

Leading University DO NOT COPY

126. A. Biswas, M. A. Al Nejrum, A. Imran, A. T. Sejuty, F. Fairooz, S. Puppala & S. Talukder, *Generative Adversarial Networks for Data Augmentation.* In ***Data-Driven Approaches on Medical Imaging***, 2023, 159–177.

127. M. Ring, D. Schlör, D. Landes & A. Hotho, *Flow-Based Network Traffic Generation Using Generative Adversarial Networks*. **Computers & Security,** 82, 2019, 156–172.

128. R. S. Shukla & A. M. Alatawi, *Enhancing Internet Traffic Forecasting in MEC Environments with 5GT-Trans: Leveraging Synthetic Data and Transformer-Based Models.* **IEEE Access,** 13, 2025, 83607–83618. Available online: https://doi.org/10.1109/ACCESS.2025.3569389

129. B. Vaz & Á. Figueira, *GANs in the Panorama of Synthetic Data Generation Methods.* **ACM Transactions on Multimedia Computing, Communications and Applications,** 21, 2024, 1–28.

130. A. Saltelli, *Global Sensitivity Analysis: The Primer*. **Chichester, UK: John Wiley & Sons**, 2008.

131. J. Fox, G. Ökten & B. Uzunoğlu, *Global Sensitivity Analysis for Power Systems via Quasi-Monte Carlo Methods.* In Proceedings of the 2019 4th International Conference on System Reliability and Safety (ICSRS), 2019, 446–451. Available online: https://doi.org/10.1109/ICSRS48664.2019.8987666

132. J. Zhang, *Modern Monte Carlo Methods for Efficient Uncertainty Quantification and Propagation: A Survey.* **Wiley Interdisciplinary Reviews: Computational Statistics,** 13, 2021, e1539.

133. Z. Ma, M. Xiao, Y. Xiao, Z. Pang, H. V. Poor & B. Vucetic, *High-Reliability and Low-Latency Wireless Communication for Internet of Things: Challenges, Fundamentals, and Enabling Technologies.* **IEEE Internet of Things Journal,** 6, 2019, 7946–7970. Available no: https://doi.org/10.1109/JIOT.2019.2907245

134. A. O. Sangodoyin, M. O. Akinsolu & I. Awan, *A Deductive Approach for the Sensitivity Analysis of Software-Defined Network Parameters.* **Simulation Modelling Practice and Theory,** 103, 2020, 102099.

135. I. M. Sobol, *Global Sensitivity Indices for Nonlinear Mathematical Models and Their Monte Carlo Estimates*. **Mathematics and Computers in Simulation,** 55, 2001, 271–280.

136. F. K. Zadeh, J. Nossent, F. Sarrazin, F. Pianosi, A. Van Griensven, T. Wagener & W. Bauwens, *Comparison of Variance-Based and Moment-Independent Global Sensitivity Analysis Approaches by Application to the SWAT Model.* **Environmental Modelling & Software,** 91, 2017, 210–222.

137. S. L. Piano, F. Ferretti, A. Puy, D. Albrecht & A. Saltelli, *Variance-Based Sensitivity Analysis: The Quest for Better Estimators and Designs between Explorativity and Economy*. **Reliability Engineering & System Safety,** 206, 2021, 107300.

Leeds City University DO NOT COPY

138. P. Phromphan, J. Suvisuthikasame, M. Kaewmongkol, W. Chanpichitwanich & S. Sleesongsom, *A New Latin Hypercube Sampling with Maximum Diversity Factor for Reliability-Based Design Optimization of HLM.* **Symmetry,** 16, 2024, 901.

139. P. Borisut & A. Nuchitprasittichai. *Adaptive Latin Hypercube Sampling for a Surrogate-Based Optimization with Artificial Neural Network*. **Processes,** 11, 2023, 3232.

140. C. Pandey, V. Tiwari, J. J. P. C. Rodrigues & D. S. Roy. *5GT-GAN-NET: Internet Traffic Data Forecasting with Supervised-Loss-Based Synthetic Data over 5G.* **IEEE Transactions on Mobile Computing,** 23, 2024, 10694–10705. Available no: https://doi.org/10.1109/TMC.2024.3364655

141. Á. Figueira & B. Vaz, *Survey on Synthetic Data Generation, Evaluation Methods and GANs.* **Mathematics,** 10(19), 2022, 2733.

142. B. Alhijawi, T. Mismar, M. A. Alsmirat & M. Al-Ayyoub, *A Comprehensive Survey on DDoS Attacks and Defense Mechanisms: Classifications, Challenges, and Future Directions.* **Computer Science Review,** 37, 2020, 100279. Available no: https://doi.org/10.1016/j.cosrev.2020.100279

143. A. K. Jain, H. Shukla & D. Goel, *A Comprehensive Survey on DDoS Detection, Mitigation, and Defense Strategies in Software-Defined Networks*. **Cluster Computing,** 2024. Available no: https://doi.org/10.1007/s10586-024-04596-z

144. W. Huang, L. Zhou, & Z. Li, *DDoS Attack Detection Based on Machine Learning and Deep Learning Techniques: A Comprehensive Review.* **Journal of Network and Computer Applications,** 238, 2024, 1256. Available no: https://doi.org/10.1016/S2772-6711(24)00125-6

Lead City University DO NOT COPY

Chapter Three

Methodology

3.1. Introduction

The SDN architecture and the IoT network topology used in this work are introduced in this section, providing a detailed overview of the foundational network design. It also describes the traffic scenario simulation carried out on the SDN-reliant IoT network model, expanding on techniques developed in the authors' previous publications to make the study more independent[1,2,3]. The subsections cover the architecture and topology setup, network traffic simulation techniques, and the incorporation of knowledge from earlier studies to support the SDN-reliant IoT model.

3.2. SDN Architecture

Lead City University DO NOT COPY

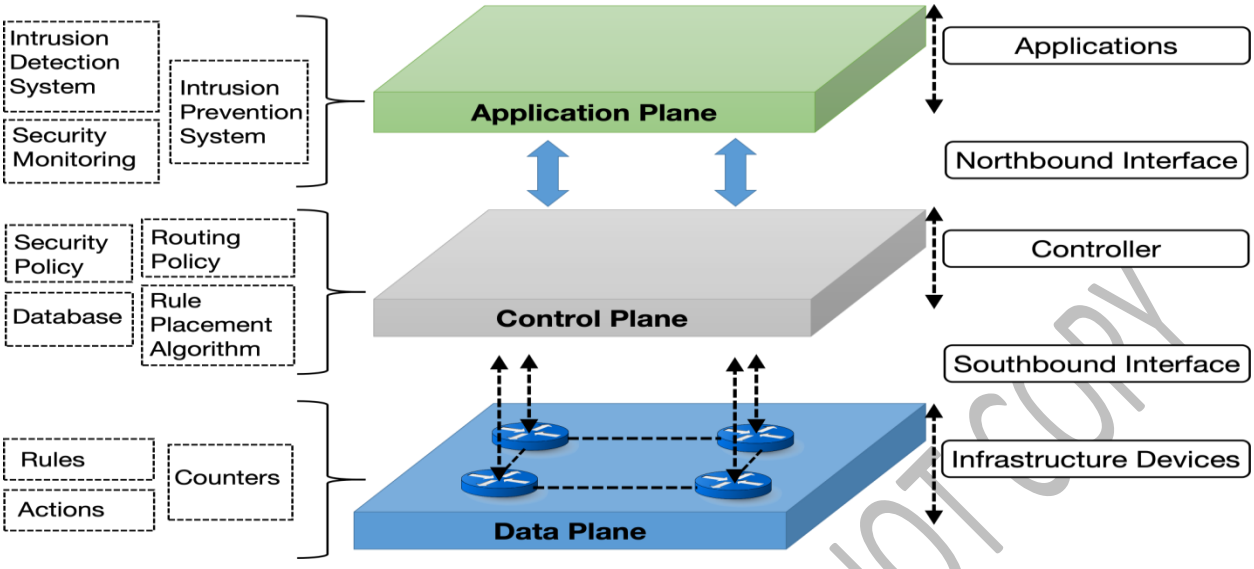


Figure 3.1. Typical SDN architecture

Source: Author's Computation 2026

The controller functions as the centralised decision-making unit in the SDN architecture depicted in Figure 3.1, connecting the data and application planes[4]. The controller decides what should be done in the data plane by using inputs from applications in the application plane[5]. Hardware or software-based forwarding devices in the data plane carry out these directives by managing packet routing and forwarding in accordance with pre-established forwarding policies[5]. This configuration, which is shown in Figure 3.1, enables dynamic network adaptation and effective policy implementation. The presented SDN architecture serves as the study's fundamental framework, allowing investigation of its potential to improve the security and operational effectiveness of IoT networks.

3.3. IoT Network Topology

Lead City University DO NOT COPY

IoT networks are characterized by their diverse topologies, inherited from traditional data and communication networks, enabling flexible arrangements and interconnection of devices[6]. Among the common topologies employed are mesh, star, tree, and point- to-point configurations, each offering specific advantages depending on the application context[7]. Within the IoT ecosystem, the data center (DC) serves as the central hub, managing substantial traffic and ensuring efficient data processing.

For DC architectures, the Fat-Tree and BCube topologies are widely utilized[8]. The Fat-Tree topology shown in Figure 2, in particular, is favored in this study due to its advantages in scalability, consistent bandwidth distribution, and cost-effective design[9]. IoT networks that need effective and scalable data management can benefit greatly from Fat-Tree networks' ability to support predictable performance and enable line-speed transmission when packets are equally distributed by using switches with uniform capacity. IoT networks rely heavily on data centres, which are also a component of the network architecture, and are typically multilayered[11]. When considering the data processing, storage, and management (including traffic management) levels in IoT networks, this is particularly true[12]. Because of this, the network topology employed in this work is quite similar to a conventional IoT network as previously defined[3]. In earlier research, similar IoT systems with multi-layer network designs, tree topologies, and hierarchical layouts have also been documented[13,14,15,16]. Building on this foundation, In order to enable wide-area network adaptability and scalability, the unique SDN architecture in this study uses an IoT network with a tree topology (see Figure 3.1).

Lead City University DO NOT COPY

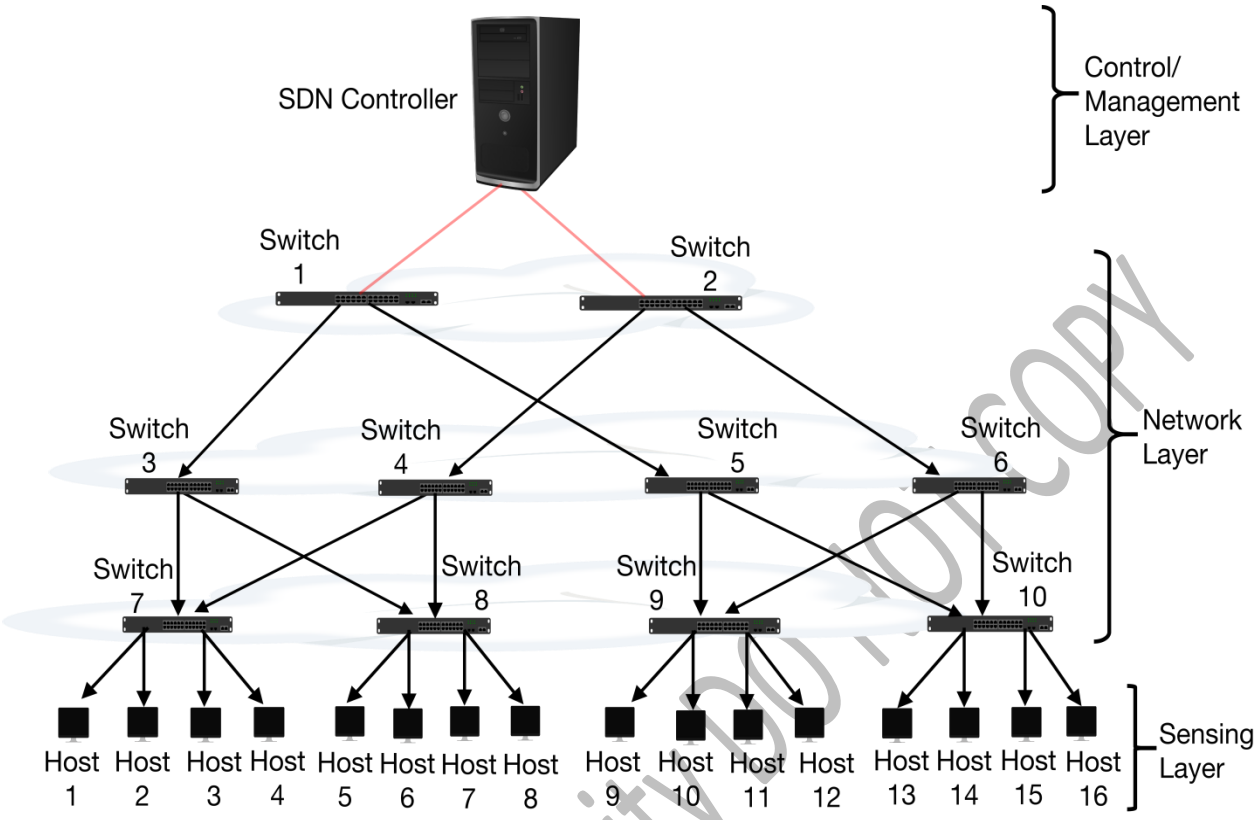


Figure 3.2. Illustration of the modeled SDN-reliant IoT network architecture (a typification)

Source: Author's Computation 2026

In network management, it is standard practice to examine network behaviour and characteristics prior to deployment using emulators or simulation techniques[13,17,18]. Mininet is one of these techniques that is well known for its efficacy in SDN research[3,13,17,18]. Networks modelled in the Mininet environment accurately reflect real-world SDN settings, as numerous studies have shown[13,17,18]. The Mininet emulator was used to implement the SDN-reliant IoT network model used in this work on a server with an Intel Xeon E3-1220 CPU, 32 GB of RAM, and Kali Linux. The Mininet environment ran Ubuntu 16.10 LTS3, and the Floodlight controller was housed in VirtualBox running Ubuntu 18.10 LTS. Ten OpenFlow switches and sixteen interconnected hosts connected by 100 Mbps links made up the experimental configuration. Despite its small size, this

Lead City University DO NOT COPY

network design is representative of real-world situations[1,2,3], like campus or enterprise networks. This study examines the sensitivity and weaknesses of IoT networks under DDoS flooding attacks by combining the advanced characteristics of SDN with the simplicity of a tree topology. It emphasises the versatility of tree and/or Fat-Tree techniques in creating robust network infrastructures. It should be mentioned that the authors' earlier works[1,2,3] have also used the network topology employed in this work.

3.4. Simulation of Traffic Scenarios on the SDN-reliant IoT Network Model

The Low Orbit Ion Cannon (LOIC) was used to carry out DDoS flooding attacks, while the tools *iperf* and *ping* were used to create authentic network connectivity between nodes in order to mimic malicious and non-malicious traffic in the SDN environment. Each experimental round lasted 45 minutes, with six compromised hosts (hosts 6 to 8 and 10 to 12; see Figure 2) launching a series of 15-minute HTTP (hypertext transfer protocol), TCP (transmission control protocol) and UDP (user datagram protocol) DDoS flooding attacks against the network server. In order to have 900 observations per DDoS flooding attack, essential system parameters (throughput (T_p), jitter (J_t), and response time (R_t)) were captured every second during these times. Key performance indicators are represented by these properties: J_t calculates the variation in packet transmission delays, R_t shows the latency between task initiation and completion[2], and T_p quantifies the actual data transfer rate within the network[2]. Comprehensive insights into network performance fluctuations were provided by the data, which was gathered under normal traffic conditions (900 observations) and DDoS flooding attack scenarios (2,700 observations).

3.5. Performance Metrics of the SDN-reliant IoT Network Model

Lead City University DO NOT COPY

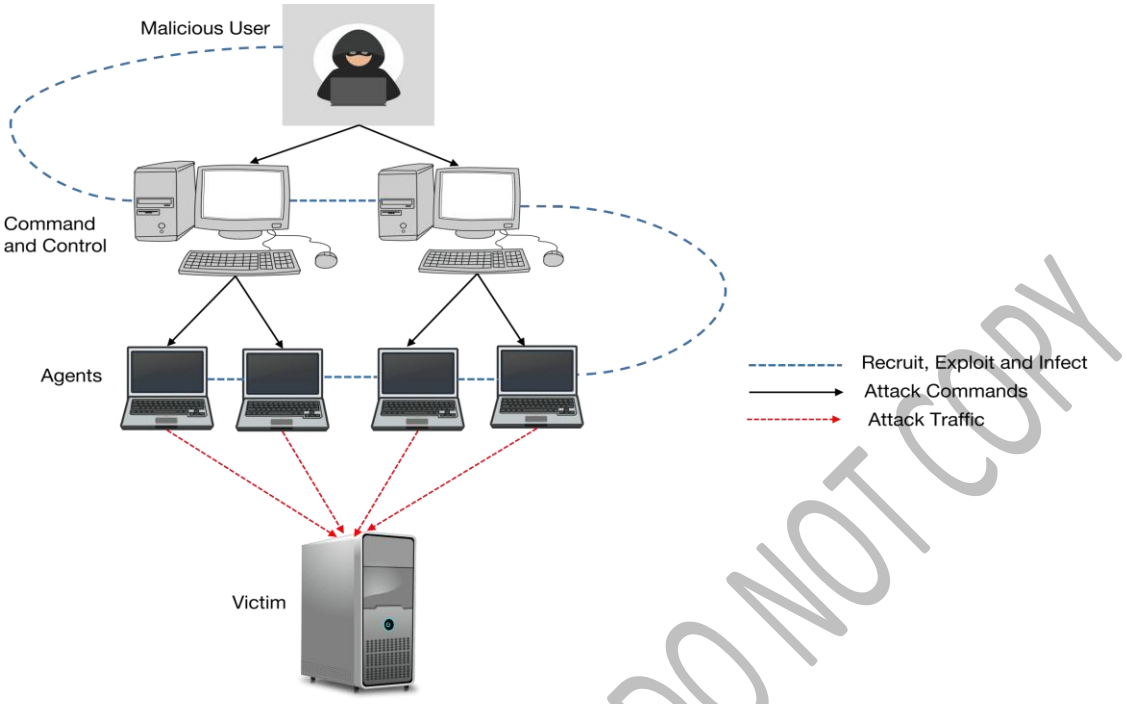


Figure 3.3. Typification of DDoS attack strategy

Source: Author's Computation, 2026

A structured process was used to build the performance metrics data set for the SDN-reliant IoT network architecture. The first step was to use the pingall command to confirm connectivity. In order to assess baseline T_p, J_t, and R_t over a 15-minute period, hosts were able to send traffic to TCP and UDP servers that were created on separate ports using iperf. LOIC was then used from compromised hosts to emulate a DDoS flooding attack technique by launching a series of HTTP, UDP, and TCP DDoS flooding attacks (Figure 3.3). During each attack, the server's performance metrics were recorded, and port numbers were dynamically adjusted to ensure no interference between scenarios. The data, initially captured in .txt format, was processed using Konstanz Information Miner (KNIME) to extract the target metrics and eliminate duplicates, ensuring a clean and structured dataset[2]. This dataset not only included normal and anomalous traffic patterns but also addressed limitations of existing SDN benchmarks by incorporating modern attack

Lead City University DO NOT COPY

footprints and traffic variations reflective of the IoT era[1,2,3,19]. The resulting data array formed the basis for ML-based ADC and SA, carried out in this study, enabling effective behavioral analysis of SDN-reliant IoT networks and the mitigation of modern DDoS flooding threats using the proposed DOE-GAN-SA framework. To guarantee that there was no interference across scenarios, port numbers were dynamically changed and the server's performance metrics were tracked during each attack. To ensure a clean and organised dataset[2], the data, which was first recorded in.txt format, was processed using Konstanz Information Miner (KNIME) to extract the desired metrics and remove duplicates. In addition to normal and abnormal traffic patterns, this dataset addressed the shortcomings of previous SDN benchmarks by integrating contemporary attack footprints and traffic changes that are indicative of the Internet of Things era[1,2,3,19]. The resultant data array served as the foundation for the ML-based ADC and SA used in this study, which allowed for the mitigation of contemporary DDoS flooding threats utilising the suggested DOE-GAN-SA framework and efficient behavioural analysis of SDN-reliant IoT networks. Notably, the authors have used similar simulation settings in their earlier works[1,2,3]. Section 4 presents the statistical distributions of the generated data for each of the IoT network traffic scenarios that rely on SDN.

3.6. Basic Techniques

This work explores and adopts a number of fundamental strategies to facilitate the construction of the DOE-GAN-SA framework. Although these strategies are well-known and established in the literature, the main goal of this work is to make it self-contained and detailed for understanding, thus it discusses them here with a particular emphasis on how they have been implemented precisely. Thus, the following is a quick discussion of various methods.

3.6.1. Design of Experiments (DOE)

Lead City University DO NOT COPY

A common approach to creating a surrogate model is to sample the design space using a well-fitted DOE method, and then estimate the computationally expensive model data. Accordingly, any approach that guides sample allocation in the design space to optimise the quantity of information gathered is commonly known as a DOE method[20,21]. At these sampling points, the computationally expensive models are evaluated in order to generate the training data required to build the surrogate model[20]. In order to effectively capture global trends in the design space, an efficient DOE technique ensures that the samples are often as widely spread as possible due to the inherent trade-off between the number of sample points and the amount of information accessible from these points[20,21]. DOE techniques include LHS, Hammersley sampling, Quasi-Monte Carlo sampling, and classical factorial designs[22,23].

3.6.2. Latin Hypercube Sampling (LHS)

LHS is perhaps the most widely used DOE technique[22] for uniform sampling distributions, and it has been adopted by many disciplines in mathematics, engineering, and computational science[20,24,25]. Compared to other DOE approaches[20,25,26], LHS has been shown to provide superior qualities that support effective filtering and a notable reduction in variance for numerous applications. To assign n samples, LHS divides the range of each sample into d bins, resulting in a total of *dn* bins in the design space for *n* samples. Various variations of LHS have been developed and/or proposed to reduce the likelihood of non-uniform distributions[27], enhance space-filling[28], optimise projective properties[29], reduce spurious correlations[30], minimise least square error, and maximise entropy[31]. LHS effectively samples input variables as a statistical technique, guaranteeing that every input parameter's whole range is represented. As demonstrated in the proposed DOE-GAN-SA framework, LHS may be used to provide representative samples of

Lead City University DO NOT COPY

network metrics, such as T_p, J_t, and R_t, which are indicative of bandwidth, latency, and packet loss in the context of SDN-reliant IoT networks. These samples are then used for SA. LHS minimises the number of simulations needed to evaluate the impact of input parameters on network performance[32] by effectively covering the input space. Due to the fact that LHS is a well-known approach[22,33,34], its complete algorithmic specifics are not given. However, in the context of our study, the following are the crucial steps in the implementation of LHS:

Step 1: Suppose there are n input parameters $X_1, X_2, \ldots, X_n$, each with a specified range $[a_i, b_i]$ where $i = 1, 2, \ldots, n$. In this work, this will be a set of any of the SDN performance metrics being considered, i.e., any of T_p, J_t, and R_t and their ranges for a specific network context. LHS then works by creating k sample points, with each input parameter broken down into k equally likely intervals. It should be noted that k is normally specified by the user, as explained later.

Step 2: For each input parameter X_i, the range is divided into k equally spaced intervals. For example, if the range of X_1 is $[a_1, b_1]$, it is divided into intervals $[a_1 + (j-1)\Delta, a_1 + j\Delta]$, where $j = 1, \ldots, k$ and $\Delta = \frac{b_1 - a_1}{k}$. This step is very essential because it ensures that the samples are distributed uniformly across their respective ranges.

Step 3: Each input parameter, X_i, has a value chosen at random from each interval. This guarantees that the parameter space is represented in its entirety. After that, a Latin hypercube structure is created using the sample points that were obtained for every parameter. In contrast to traditional random sampling, the Latin hybercube structure enables a more uniform and representative sampling. Hence, the name LHS.

Lead City University DO NOT COPY

Step 4: Each parameter's sampled values are shuffled to form k unique sample points, each of which contains a single value from the range of each parameter.

It should be mentioned that the sample size influences the outcomes of methods like LHS, which aim to distribute samples evenly across the range of feature values[34], irrespective of the value of k that the user specifies. As stated by[33], the sample space has been partitioned into bins in this work, and new samples have been drawn at a predefined bin fraction 30 bins, to be exact. This process is used to maximise insight into parameter trends by ensuring that LHS efficiently covers the space of network performance indicators across different scenarios within the SDN-reliant IoT network. This method's major goal is to gradually increase the sample size while preserving steady marginal distributions of the network parameters[33].

3.6.3. Machine Learning (ML)

Advanced analytical techniques are now required to characterise and improve the performance of IoT networks under various scenarios due to their rapid evolution. With its data-driven techniques for automated analysis and decision-making, machine learning (ML) has become a major enabler in this field[35,36]. ML often gives systems the capacity to autonomously learn from experience and improve without specifically programming it. ML is necessary to analyse data intelligently and robustly and to create the relevant real-world data-driven paradigms. ML approaches can be broadly divided into four categories: semi-supervised, supervised, unsupervised, and reinforcement learning approaches[37]. The capacity to use labelled data to model intricate correlations between input properties and desired results makes supervised learning techniques stand out among other techniques for machine learning. Given the typical SDN-reliant IoT network in this study under a variety of operational conditions[37], they are especially well suited

Lead City University DO NOT COPY

for the DOE-GAN-SA framework, which necessitates accurate prediction, to enable ADC and SA. The classification and regression tree (CART) and neural networks (ANN and GAN specifically) are the primary supervised learning methods that support the DOE-GAN-SA framework. The following is a discussion of these supervised learning techniques.

3.6.3.1 Neural Networks

Inspired by the architecture of the human brain, where interconnected nodes, or "neurones," work together to receive inputs and produce outputs, neural networks (NNs) are computational frameworks created to mimic learning and data processing[38]. NNs, also known as universal "approximators," are essential for characterising complex systems, including Internet of Things networks, because they allow for the study and optimisation of system responses using knowledge and patterns that have been learned[39]. They promote advances in network analysis and system optimisation by modelling and processing information in a manner similar to that of biological brain systems. Certain applications and architectures, such as ANNs for structured data processing and GANs for creating and improving data representations, are covered in more detail below.

3.6.3.2. Artificial Neural Network (ANN)

A typical artificial neural network (ANN) is made up of interconnected neurones that communicate with one another through synapses. It usually consists of three layers: input (which works with data), hidden (which does calculations), and output (which generates results)[40]. One can combine or modify any of the learning algorithms to create a strong ANN model that suits the provided data. Numerous real-world issues have been addressed with ANNs[41,42,43]. ANNs can be utilised for pattern recognition, classification, prediction, and optimisation, including traffic differentiation in intrusion detection and prevention systems (IDPSs)[44]. In general, data representation,

Lead City University DO NOT COPY

feedback[45], prior knowledge, and data improvement all affect how effective and efficient ANNs are. An ANN's performance can be optimised to make it more suitable for a variety of applications[46] by modifying the connection weights using training algorithms such as backpropagation. There are several ways in which ANNs can learn[47]. ANNs can be trained using a variety of batch training procedures, each with varying memory requirements, speed, and precision.

This work uses the Levenberg-Marquardt training algorithm, which is considered to be one of the most effective for ANNs[48]. By integrating the Gauss-Newton method and gradient descent, the Levenberg-Marquardt training algorithm overcomes their respective drawbacks[48,49]. Due to the inversion of the Hessian matrix and the storage needs for the Jacobian matrix, which are contingent on the quantity of patterns, outputs, and weights[49], the Levenberg-Marquardt training algorithm's main disadvantage is its computational expense. This can get too costly for networks that are too big. The Levenberg-Marquardt training algorithm, however, provided a balance of speed, stable convergence, and manageable memory consumption for all the ANN models built, according to our dataset of a few thousand instances or observations, with three inputs and one output, under 250 epochs on average (see Chapter Four). Since ANN is a well-known technique[1,38], its full algorithmic details are not provided, but the following are the essential steps in the ANN implementation in the proposed DOE-GAN-SA framework:

Step 1: Initialize the weights $W^{(l)}$ and biases $b^{(l)}$ randomly or using a predefined scheme for all layers l, where the activation or the neuron's output is A fed to the next layer.

Step 2: For each layer l, compute the weighted sum of inputs:

Lead City University DO NOT COPY

$$Z^{(l)} = W^{(l)} A^{(l-1)} + b^{(l)}$$

Apply an activation function (σ) to get:

$$A^{(l)} = \sigma(Z^{(l)})$$

Step 3: Calculate the loss function $\mathcal{L}$ to measure the error between predicted output and actual target Y:

$$\mathcal{L} = \frac{1}{m} \sum_{i=1}^{m} \ell(\hat{Y}^{(i)}, Y^{(i)})$$

where ℓ represents a chosen loss function such as mean squared error (MSE) or cross-entropy loss.

Step 4: Compute gradients of the loss for parameters using the chain rule:

$$\frac{\partial \mathcal{L}}{\partial W^{(l)}} = \delta^{(l)} A^{(l-1)T}, \quad \frac{\partial \mathcal{L}}{\partial b^{(l)}} = \delta^{(l)}$$

where $\delta^{(l)}$ is the error term:

$$\delta^{(l)} = \left(W^{(l+1)T} \delta^{(l+1)}\right) \odot \sigma'(Z^{(l)})$$

Step 5: Update the weights and biases using the Levenberg-Marquardt algorithm:

$$W^{(l)} = W^{(l)} - \alpha \frac{\partial \mathcal{L}}{\partial W^{(l)}}, \quad b^{(l)} = b^{(l)} - \alpha \frac{\partial \mathcal{L}}{\partial b^{(l)}}$$

Lead City University DO NOT COPY

where α is the learning rate.

3.6.3.3. Generative Adversarial Networks (GANs)

Given their demonstrated capacity to produce high-fidelity synthetic data by identifying complex patterns and structures in a variety of data sources, such as text, images, and time series, GANs are particularly pertinent to this work[50,51]. More precisely, they are flexible tools for applications that need synthetic data, like data augmentation, anomaly detection, and testing in data-scarce situations, because of their cross-domain adaptability[52,53]. For data production, classification, and modelling in networks and communications, a number of deep learning architectures have been investigated. While they are quite good at extracting visual information, convolutional neural networks (CNNs) perform poorly when dealing with sequential network traffic[54]. Time-series data[55] is better suitable for recurrent neural networks (RNNs) and related variations, such as gated recurrent unit (GRU) networks and long short-term memory (LSTM) networks. But when it comes to producing high-quality synthetic data56, GANs perform better than them. GANs are very good at creating realistic network traffic because of their adversarial training process[50].

In contrast to traditional synthetic data generation techniques, GANs perform exceptionally well in situations with difficult data landscapes, offering a reliable mechanism for modelling and producing realistic datasets[50]. These features make them a strong and flexible method for tackling the difficulties in generating data and analysis[53], which supports their use for data augmentation in DOE-GAN-SA framework. GANs can be used to create synthetic network traffic data that closely resembles the properties of actual traffic in SDN-reliant IoT networks. Without requiring lengthy and computationally costly real-world experiments, this data can then be utilised to investigate how various input factors impact the network's performance. The produced data can be

Lead City University DO NOT COPY

used as input for SA, offering a more economical and scalable method of parameter space[57] research.

This method enables more flexible exploration of the parameter space[58] and eliminates the need for expensive real-world experiments. This is accomplished using the generator *G(z)*, which creates synthetic data *G(z)* that mimics the actual network traffic data by sampling a random noise vector z from a noise distribution $p_z(z)$. The probability that the input is actual data rather than synthetic data is then output by the discriminator *D(x)*, which takes either real data x from the distribution $p_{data}(x)$ or synthetic data *G(z)*. Due to the fact that GAN is a well-known technique, its complete algorithmic details are not given. Nonetheless, the following is a brief description of the key GAN architecture steps included in the suggested DOE-GAN-SA framework:

Step 1: Initialize the parameters θ_g of G (z) and θ_d of D (x).

Step 2: For each training step, sample real data $x \sim p_{data}(x)$ and noise $z \sim p_z$ (z).

Update θ_d by maximizing the likelihood that it correctly classifies real and synthetic data:

$$\max_{\theta_d} \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

Step 3: Update θ_g to minimize the ability of D (x) to distinguish real from synthetic data:

$$\min_{\theta_g} \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

Lee City University DO NOT COPY

Step 4: Alternate between training G (z) and D (x) until G (z) produces data indistinguishable from real data.

The min-max game between the G (z) and D (x) can be summarized using the objective function:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

3.6.3.4. Classification and Regression Tree (CART)

CART's robust and interpretable decision-making skills make it an essential supervised learning technique used in the proposed DOE-GAN-SA paradigm. Unlike other classification techniques, CART is well known for being straightforward, effective, and capable of handling both continuous and categorical data, which makes it an excellent choice for simulating intricate SDN-reliant network scenarios[3]. Its ranking among the Top 10 Data Mining Algorithms[60] and its proven dominance in prediction tasks in SDN-reliant network scenarios[3] further emphasise its applicability. By repeatedly dividing the input feature space, CART builds decision trees (DTs) with the goal of maximising information gain at each split. This method is more applicable to a variety of datasets since it is non-parametric, which means it does not assume anything about the distribution of the data. For a set of input attributes $X = \{x_1, x_2, \ldots, x_d\}$, By assessing the impurity reduction using a metric like Gini's diversity index[61], the CART algorithm determines the best split at each node. The following crucial actions are then taken in order to implement CART in the suggested DOE-GAN-SA architecture (complete algorithmic specifics are not included because CART is a well-known approach[3,62]):

Step 1: Determine the impurity of each node I by applying Gini's diversity index:

Lead City University DO NOT COPY

$$gdi = 1 - \sum_{n=1}^{N} (p_n)^2$$

where the number of classes is N, and p_n is the proportion of observations in class n.

Step 2: Evaluate all possible splits and select the one that maximizes impurity gain:

$$I_G = p(T) - \Big(p(T_L) + p(T_R)\Big)$$

where T is the set of observations at the current node, and T_L and T_R are the subsets

resulting from the split.

Step 3: Continue splitting recursively until a stopping criterion is met, such as reaching a maximum depth or a minimum number of samples per leaf.

Step 4: Prune the tree (if enabled) to reduce overfitting and improve generalization.

It is important to note that once the hyperparameters of ML approaches are appropriately optimised, they often perform better[3]. This is due to the fact that the quality of the data and training techniques are intrinsically linked to the dependability of machine learning models, and it is frequently inevitable that spurious correlations will be discernible by human analysts after evaluation[3]. Since it is rarely possible to attain optimality in both metrics at the same time, a compromise between high accuracy (positive predictive value, or PPV) and high recall (sensitivity, or true positive rate, or TPR) is usually inevitable[63]. Hyperparameter tuning is one method for

Lead City University DO NOT COPY

optimising (i.e., maximising) either recall or precision on the validation set, which is the set of observations used for hyperparameter optimization[63].

Nevertheless, this procedure is not within the scope of the present research; hyperparameter optimisation is not examined in this work. Instead, the default hyperparameter settings from MATLAB (Matrix Laboratory) and the Scikit-learn library[64,65] are used for all machine learning algorithms used in the DOE-GAN-SA framework. The default settings in MATLAB and the Scikit-learn package are assumed to be unlikely to be altered by most network engineers, especially those who lack significant ML knowledge. Table 3.1 lists the default values for a few important hyperparameters.

Table 3.1. Table of Hyperparameters.

Method	Default Hyperparameter Setting
CART	Maximum Number of Splits = 100
GAN	Learning Rate = 2 ×10−4
ANN	Number of Hidden Layers = 10

Lead City University DO NOT COPY

3.7. The DOE-GAN-SA Framework

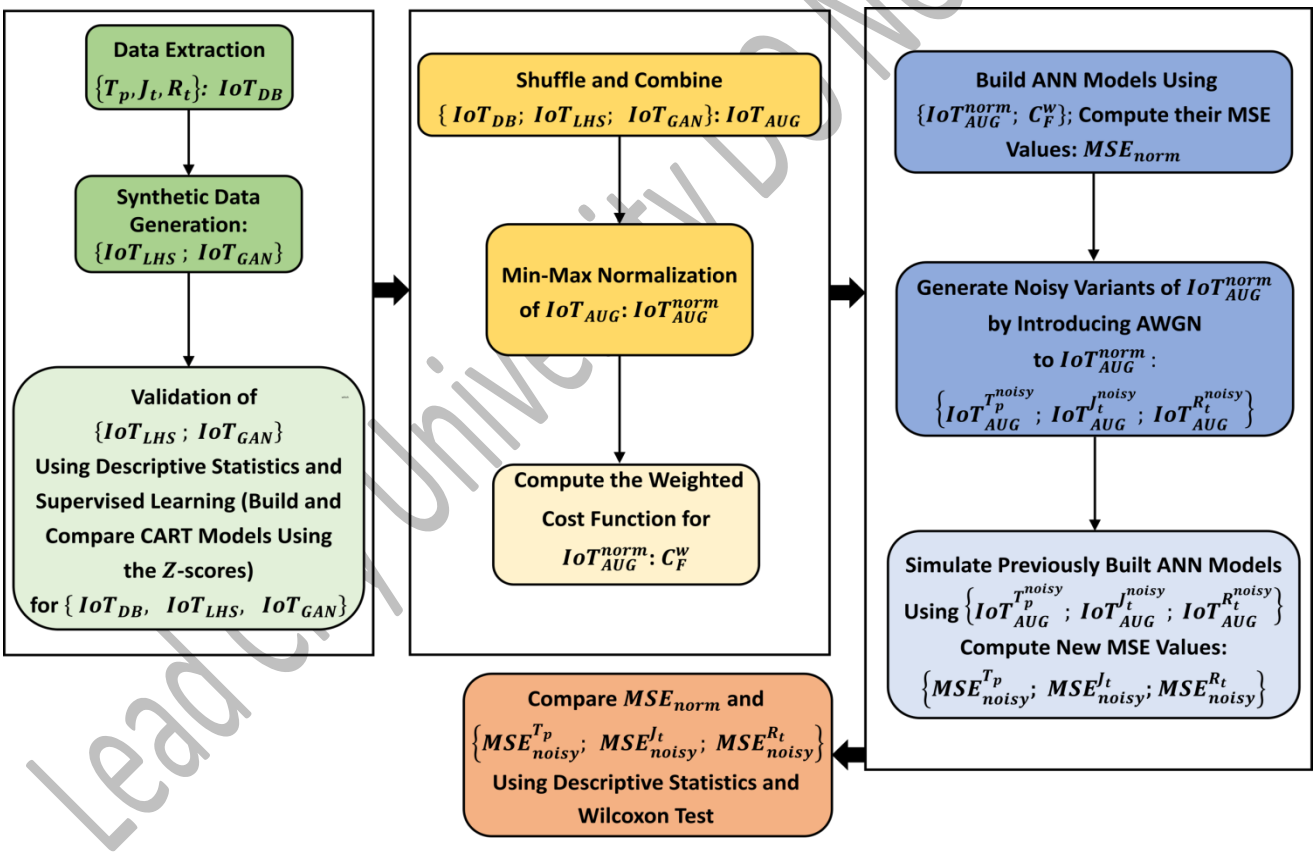


Figure 3.4 Flow diagram of the DOE-GAN-SA framework

Source: Author's Computation 2026

Lead City University DO NOT COPY

The flow diagram for the proposed DOE-GAN-SA framework is shown in Figure 3.1, and the principal steps in the framework are discussed as follows:

Step 1: Collect the simulated metrics data describing the behavior and/or performance of the SDN-reliant IoT network (see Section 3). In this case, the dataset (IoT_{DB}) formed from collecting T_p , J_t , and R_t metrics for all the scenarios on the SDN-reliant IoT network can be described as follows:

$$IoT_{DB} = \{T_p, J_t, R_t\} \ \forall \text{ IoT Network Scenarios}$$

Step 2: Use the collected data from Step 1 (i.e., IoT_{DB}) to implement LHS-based DOE (see Section 4.1.1) and GAN-based synthetic data generation (see Section 4.2.1), concurrently to enlarge the parameter space by $N \in [1, \ldots, \mathbb{R}]$ samples for each network scenario to mimic on-the-fly T_p, J_t, and R_t metrics for the SDN-reliant IoT network. The new datasets from the LHS-based DOE and GAN-based synthetic data generation (i.e. IoT_{LHS} and IoT_{GAN}, respectively) can be described as follows:

$IoT_{LHS} = \{T_p^{LHS}, J_t^{LHS}, R_T^{LHS}\}$ ∀ IoT Network Scenarios

$IoT_{GAN} = \{T_p^{GAN}, J_t^{GAN}, R_T^{GAN}\}$ ∀ IoT Network Scenarios

In this Step, N is set to 900 to match the number of observations per scenario for all the SDN-reliant IoT network scenarios considered (see Section 3.3).

Step 3: Validate the new LHS and GAN metrics (i.e., T_p^{LHS} , J_t^{LHS}, R_T^{LHS}, and T_p^{GAN} , J_t^{GAN}, R_T^{GAN}) from Step 2 for each of the considered SDN-reliant IoT network scenario (i.e., normal, HTTP DDoS flooding attack, TCP DDoS flooding attack and UDP DDoS flooding attack operating conditions) using

Lead City University DO NOT COPY

descriptive statistics, and by building and comparing classification models using the standard scores (Z-scores) for [T_p, J_t, R_t], [T_p^{LHS} , J_t^{LHS}, R_T^{LHS}] and [T_p^{GAN} , J_t^{GAN}, R_T^{GAN}], respectively, and a suitable supervised learning technique (CART in this particular instance - see Section 4.2.2). The standardization procedure in this Step is carried out as suggested in [38,70,126], and it can be described as follows:

$$Z_i = \frac{(X_i - \mu_i)}{\sigma_i} \quad \forall \text{ IoT Network Scenarios}$$

where Z_i is the Z-score of the *ith* observation for any metric, $X_i \in \{IoT_{DB}, IoT_{LHS}, \text{ and } IoT_{GAN}\}$ having a mean and standard deviation of μ_i and σ_i, respectively.

Step 4: Shuffle the validated LHS and GAN metrics from Step 3, and combine them with the simulated data of metrics from Step 1 (i.e., [T_p; T_p^{LHS}; T_p^{GAN}], [J_t; J_t^{LHS}; J_t^{GAN}], and [R_t; R_T^{LHS}; R_T^{GAN}]) for each of the SDN-reliant IoT network scenario considered (i.e., normal, HTTP DDoS flooding attack, TCP DDoS flooding attack and UDP DDoS flooding attack operating conditions) to have an augmented dataset for the network metrics, described as follows:

$$IoT_{AUG} = \{T_p^{AUG} , J_t^{AUG}, R_T^{AUG} \} \quad \forall \text{ IoT Network Scenarios}$$

Step 5: Normalize the combined metrics from Step 4 using min-max normalization. The min-max or linear transformation normalization in this Step is carried out as suggested in[1,66,67], and it can be described as follows:

$$IoT_{AUG^i}^{norm} = \frac{IoT_{AUG^i} - IoT_{AUG^i}^{min}}{IoT_{AUG^i}^{max} - IoT_{AUG^i}^{min}} \quad \forall \text{ IoT Network Scenarios}$$

Lead City University DO NOT COPY

where IoT_{AUG^i} is the ith observation in the augmented SDN-reliant IoT network dataset (IoT_{AUG}) whose minimum and maximum values are $IoT_{AUG^i}^{min}$ and $IoT_{AUG^i}^{max}$, respectively, and $IoT_{AUG^i}^{norm}$ is the normalized value defined as $IoT_{AUG^i}^{norm} \in [0, 1]$.

In this Step, min-max normalization is preferred to ensure that each metric in [T_p^{AUG}, J_t^{AUG}, R_T^{AUG}] contributes a similar relative numerical weight, reducing data redundancy and ensuring that all target input values have an amicable metric scale prior to the implementation of ML-driven SA. The formed normalized augmented dataset can be described mathematically as follows:

$$IoT_{AUG}^{norm} = \{T_{p_{norm}}^{AUG}, J_{t_{norm}}^{AUG}, R_{t_{norm}}^{AUG}\} \; \forall \text{ IoT Network Scenarios}$$

Step 6: Generate response variables for every observation in IoT_{AUG}^{norm} by deriving a weighted cost function (C_F^w) as recommended in[1]. Specifically, $C_{F_i}^w$ for the *ith* the observation in IoT_{AUG}^{norm} can be derived as follows:

$$C_{F_i}^w = \quad [(abs(T_{p_{norm}}^{AUG_i} - J_{t_{norm}}^{AUG_i}) \times R_{t_{norm}}^{AUG_i})] \times w$$

w is set to 1, 2, 3, and 4, for when the SDN-reliant IoT network is operating normally, under TCP DDoS flooding attack, under UDP DDoS flooding attack, and HTTP DDoS flooding attack, respectively, as recommended in[1].

Step 7: Build ANN models (see Section 4.2.1) using IoT_{AUG}^{norm} as the explanatory variables and C_F^w as the corresponding response variables ∀ IoT network scenarios. Then compute the MSE values for the respective ANN models and store them in MSE_{norm}.

Lead City University DO NOT COPY

Step 8: Generate noisy variants of IoT_{AUG}^{norm} by introducing an Additive White Gaussian Noise (AWGN) severally to ${T_p}_{norm}^{AUG}$, ${J_t}_{norm}^{AUG}$, and ${R_t}_{norm}^{AUG}$ as carried out in[1]. The new noisy datasets can be described mathematically as follows:

$$IoT_{AUG}^{T_p^{noisy}} = \{{T_p}_{noisy}^{AUG}, {J_t}_{norm}^{AUG}, {R_t}_{norm}^{AUG}, C_F^W \} \forall \text{ IoT NetworkScenarios}$$

$$IoT_{AUG}^{J_t^{noisy}} = \{{T_p}_{norm}^{AUG}, {J_t}_{noisy}^{AUG}, {R_t}_{noisy}^{AUG}, C_F^W \} \forall \text{ IoT Network Scenarios}$$

$$IoT_{AUG}^{R_t^{noisy}} = \{{T_p}_{norm}^{AUG}, {J_t}_{norm}^{AUG}, {R_t}_{noisy}^{AUG}, C_F^W \} \forall \text{ IoT Network Scenarios}$$

Step 9: Simulate the same ANN models in Step 7 using the datasets in Equations in step 8 respectively. Compute the MSE values and store them in $MSE_{noisy}^{T_p}$, $MSE_{noisy}^{J_t}$, and $MSE_{noisy}^{R_t}$, respectively.

Step 10: Compare the MSE values from Steps 7 and 9 using descriptive statistics and hypothesis testing (specifically, Wilcoxon test[68]) to ascertain and statistically validate the sensitivities of T_p, J_t, and R_t in $[T_p^{AUG}, J_t^{AUG}, R_T^{AUG}]$, respectively. The comparisons are mathematically described as follows:

$$SA_I \begin{cases} MSE_{norm} \quad and \; MSE_{noisy}^{T_p}; \; \forall \text{ IoT Network Scenarios} \\ MSE_{norm} \quad and \; MSE_{noisy}^{J_t}; \; \forall \text{ IoT Network Scenarios} \\ MSE_{norm} \quad and \; MSE_{noisy}^{R_t}; \; \forall \text{ IoT Network Scenarios} \end{cases}$$

where SA_I denotes the SA inferences drawn from the comparisons of Equation. Standardisation (Z-score normalisation) is the recommended approach in Step 3 as opposed to min-max

Lead City University DO NOT COPY

normalisation. This is due to the assumption that classifiers used to solve classification problems have a normal or nearly normal distribution [69]. Due to its popularity and proven capacity to outperform other well-known classifiers in predicting SDN-reliant network scenarios[3,60], the CART technique was chosen to implement Step 3 as the classifier in Section 4.2.2. A cross-validation approach has been used to evaluate the predictive accuracy of the CART models that were fitted in Step 3 and reduce the possibility of overfitting. The dataset size, which consists of about 11,000 observations, influences the choice of this validation technique. According to standard procedures for handling classification problems, all classifiers and experiments have consistently used a five-fold nested cross-validation framework[3,70]. This approach is consistent with well-established methodologies for improving model reliability and generalization[3,70].

In Steps 7 and 9, ANN is preferred because of its robustness and popularity as a universal approximation function[71], and AWGN is used to alter the states of $T_{p_{norm}}^{AUG}$, $J_{t_{norm}}^{AUG}$, and $R_{t_{norm}}^{AUG}$ severally in Step 8 as recommended in[1] for the ML-driven SA of SDN parameters. It should be mentioned that the training of the ANN models constructed in Step 7 is greatly influenced by the number of processing elements in each layer as well as the total number of layers. To put it another way, too many processing elements can lead to overfitting to the training dataset[43,72], while too few can impede the learning process. The dataset was divided for this investigation using the methods recommended by earlier studies[1,73]. In particular, 7,560 data samples were set aside for training, which accounted for 70% of the entire dataset. Of the remaining data, 1,620 data samples (15%) were set aside for testing, and another 1,620 data samples (15%) were set aside for validation. This partitioning strategy guarantees a fair approach to the development and assessment of the model.

Lead City University DO NOT COPY

The DOE-GAN-SA framework described above was implemented in this work using a workstation that had an Intel 6-core i7-8700 3.20 GHz CPU and 32.0 GB of RAM, unless otherwise specified. To obtain a large enough sample size (in this case 50) to utilise a z-statistic to ascertain the probability values for all hypothesis tests conducted, all independent experimental runs are repeated 50 times[3,67]. For the synthetic data generated, i.e., IoT_{LHS} and IoT_{GAN}, To minimise the possible impact of extreme values and account for the stochasticity of the LHS and GAN procedures[22,50], the median of the 50 instances of generating them is used.

Leading University DO NOT COPY

Endnotes

1. A. O. Sangodoyin, M. O. Akinsolu & I. Awan, *A Deductive Approach for the Sensitivity Analysis of Software-Defined Network Parameters.* **Simulation Modelling Practice and Theory**, 2020, 102099.

2. M. O. Akinsolu, A. O. Sangodoyin & U. Uyoata, *Behavioral Study of Software-Defined Network Parameters Using Exploratory Data Analysis and Regression-Based Sensitivity Analysis.* **Mathematics**, 2022, 2536. Available online: https://doi.org/10.3390/math10142536

3. A. O. Sangodoyin, M. O. Akinsolu, P. Pillai & V. Grout, *Detection and Classification of DDoS Flooding Attacks on Software-Defined Networks: A Case Study for the Application of Machine Learning.* **IEEE Access**, 2021, 122495–122508. Available online: https://doi.org/10.1109/ACCESS.2021.3109490

4. D. Kreutz, F. Ramos P. Veríssimo, C. E. Rothenberg, S. Azodolmolky & S. Uhlig, *Software-Defined Networking: A Comprehensive Survey.* **arXiv**, 2014. Available online: https://doi.org/10.11063109JPROC.2014.2371999

5. D. Kreutz, F. Ramos & P. Veríssimo, *Towards Secure and Dependable Software-Defined Networks.* In **Proceedings of the ACM SIGCOMM Workshop on Hot Topics in Software-Defined Networking**, 2013. Available online: https://doi.org/10.1145/2491185.2491199

6. M. R. Palattella, M. Dohler, A. Grieco, G. Rizzo, J. Torsner, T. Engel & L. Ladid, *Internet of Things in the 5G Era: Enablers, Architecture, and Business Models.* **IEEE Journal on Selected Areas in Communications,** 2016, 510–527. Available online: https://doi.org/10.1109/JSAC.2016.2525418

7. Z. Nurlan, T. Zhukabayeva, M. Othman, A. Adamova & N. Zhakiyev, *Wireless Sensor Network as a Mesh: Vision and Challenges.* **IEEE Access**, 2021, 46–67.

8. B. Neilson & N. Rossiter, *Automating Labour and the Spatial Politics of Data Centre Technologies.* In **Topologies of Digital Work**, 2022, 77–101.

9. J. Escudero-Sahuquillo, P. J. Garcia, F. J. Quiles, S. A. Reinemo, T. Skeie, O. Lysne & J. Duato, *A New Proposal to Deal with Congestion in InfiniBand-Based Fat-Trees.* **Journal of Parallel and Distributed Computing**, 2014, 1802–1819.

10. P.-J. Lu, M.-C. Lai & J.-S. Chang, *A Survey of High-Performance Interconnection Networks in High-Performance Computer Systems.* **Electronics**, 2022, 1369.

11. M. Lombardi, F. Pascale & D. Santaniello, *Internet of Things: A General Overview between Architectures, Protocols and Applications.* **Information**, 2021, 87.

12. L. Bittencourt, R. Immich, R. Sakellariou, N. Fonseca, E. Madeira, M. Curado, L. Villas, L. A. Da Silva, C. Lee & O. Rana, *The Internet of Things, Fog and Cloud Continuum: Integration and Challenges.* **Internet of Things**, 2018, 134–155.

13. S. M. M. Gilani, M. Usman, S. Daud, A. Kabir, Q. Nawaz & O. Judit, *SDN-Based Multi-Level Framework for Smart Home Services.* **Multimedia Tools and Applications**, 2024, 327–347.

14. R. Chaudhary, G. S. Aujla, N. Kumar & P. K. Chouhan, *A Comprehensive Survey on Software-Defined Networking for Smart Communities.* **International Journal of Communication Systems**, 2025, e5296.

15. N. Bizanis & F. A. Kuipers, *SDN and Virtualization Solutions for the Internet of Things: A Survey.* **IEEE Access**, 2016, 5591–5606.

16. M. Burhan, R. A. Rehman, B. Khan & B. S. Kim, *IoT Elements, Layered Architectures and Security Issues: A Comprehensive Survey.* **Sensors**, 2018, 2796.

17. R. Dos Reis Fontes, C. Campolo, C. E. Rothenberg & A. Molinaro, *From Theory to Experimental Evaluation: Resource Management in Software-Defined Vehicular Networks.* **IEEE Access**, 2017, 3069–3076. Available online: https://doi.org/10.1109/ACCESS.2017.2671030

18. D. Muelas, J. Ramos & J. E. L. de Vergara, *Assessing the Limits of Mininet-Based Environments for Network Experimentation.* **IEEE Network**, 2018, 168–176. Available online: https://doi.org/10.1109/MNET.2018.1700277

19. R. Sommer & V. Paxson, *Outside the Closed World: On Using Machine Learning for Network Intrusion Detection.* In **Proceedings of the IEEE Symposium on Security and Privacy**, 2010, 305–316.

20. T. J. Santner, B. J. Williams & W. I. Notz, *The Design and Analysis of Computer Experiments.* **Springer**, 2003. Available online: https://doi.org/10.1007/978-1-4757-3799-8

Leadcity University DO NOT COPY

21. S. Koziel & X. S. Yang, *Computational Optimization: Methods and Algorithms.* **Springer**, 2011.

22. M. Stein, *Large Sample Properties of Simulations Using Latin Hypercube Sampling.* **Technometrics**, 1987, 143–151.

23. A. Giunta, S. Wojtkiewicz & M. Eldred, *Overview of Modern Design of Experiments Methods for Computational Simulations.* In **Proceedings of the 41st Aerospace Sciences Meeting**, 2003.

24. M. O. Akinsolu, B. Liu, V. Grout, P. I. Lazaridis, M. E. Mognaschi & P. Di Barba, *A Parallel Surrogate-Model-Assisted Evolutionary Algorithm for Electromagnetic Design Optimization.* **IEEE Transactions on Emerging Topics in Computational Intelligence**, 2019, 93–105.

25. M. D. Shields & J. Zhang, *The Generalization of Latin Hypercube Sampling.* **Reliability Engineering & System Safety**, 2016, 96–108.

26. M. O. Akinsolu, *Efficient Surrogate-Model-Assisted Evolutionary Algorithm for Electromagnetic Design Automation with Applications.* 2019.

27. K. Palmer & K. L. Tsui, *A Minimum-Bias Latin Hypercube Design.* **IEEE Transactions on Reliability**, 2001, 793–808.

28. K. Dalbey & G. Karystinos, *Fast Generation of Space-Filling Latin Hypercube Sample Designs.* In **Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference**, 2010.

29. M. Liefvendahl & R. Stocki, *A Study on Algorithms for Optimization of Latin Hypercubes.* **Journal of Statistical Planning and Inference**, 2006, 3231–3247.

30. M. Vořechovský & D. Novák, *Correlation Control in Small-Sample Monte Carlo Simulations I.* **Probabilistic Engineering Mechanics**, 2009, 452–462.

31. J.S. Park, *Optimal Latin-Hypercube Designs for Computer Experiments.* **Journal of Statistical Planning and Inference**, 1994, 95–111.

32. J. C. Helton & F. J. Davis, *Latin Hypercube Sampling and the Propagation of Uncertainty.* **Reliability Engineering & System Safety**, 2003, 23–69.

33. J. Pleming & R. Manteufel, *Replicated Latin Hypercube Sampling.* In **Proceedings of the 46th AIAA Conference**, 2005.

34. D. D. Saurette, A. Biswas & A. W. Gillespie, *Determining Minimum Sample Size for the Conditioned Latin Hypercube Sampling Algorithm.* **Pedosphere**, 2024, 530–539.

Lead City University DO NOT COPY

35. I. H. Sarker, M. H. Furhad & R. Nowrozy, *AI-Driven Cybersecurity: An Overview.* **SN Computer Science,** 2021, 173.

36. I. H. Sarker, *Machine Learning: Algorithms, Real-World Applications and Research Directions.* **SN Computer Science**, 2021, 160.

37. M. Mohammed, M. B. Khan & E. B. M. Bashier, *Machine Learning: Algorithms and Applications.* **CRC Press**, 2016. Available no: https://doi.org/10.1201/9781315371658

38. M. Chen, U. Challita, W. Saad, C. Yin & M. Debbah, *Artificial Neural-Network-Based Machine Learning for Wireless Networks.* **IEEE Communications Surveys & Tutorials**, 2019, 3039–3071.

39. M. van Gerven & S. Bohte, *Artificial Neural Networks as Models of Neural Information Processing.* **Frontiers in Computational Neuroscience**, 2017. Available no: https://doi.org/10.3389/fncom.2017.00114

40. K. Kumar & G. S. M. Thakur**,** *Advanced Applications of Neural Networks and Artificial Intelligence: A Review.* **International Journal of Information Technology and Computer Science**, 2012, 57–68.

41. M. J. Kang & J. W. Kang, *Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security.* **PLOS ONE**, 2016, e0155781.

42. M. S. ElSayed, N. A. Le-Khac, M. A. Albahar & A. Jurcut, *A Novel Hybrid Model for Intrusion Detection Systems in SDNs.* **Journal of Network and Computer Applications**, 2021, 103160.

43. N. S. Shaji, T. Jain, R. Muthalagu & P. M. Pawar, *Deep-Discovery: Anomaly Discovery in SDNs Using ANN.* **Computers & Security**, 2023, 103320.

44. E. Hodo, X. Bellekens, A. Hamilton, P. L. Dubouilh, E. Iorkyase, C. Tachtatzis & R. Atkinson, *Threat Analysis of IoT Networks Using ANN Intrusion Detection.* In **ISNCC**, 2016, 1–6. Available no: https://doi.org/10.1109/ISNCC.2016.7746067

45. S. J. Russell & P. Norvig, *Artificial Intelligence: A Modern Approach.* **Pearson**, 2016.

46. Y. LeCun, Y. Bengio & G. Hinton, *Deep Learning.* **Nature**, 2015, 436–444.

47. I. Goodfellow, Y. Bengio & A. Courville, *Deep Learning.* **MIT Press**, 2016.

48. M. T. Hagan & M. B. Menhaj, *Training Feedforward Networks with the Marquardt Algorithm.* **IEEE Transactions on Neural Networks**, 1994, 989–993.

49. M. Li, H. Wu, Y. Wang, H. Handroos & G. Carbone, *Modified Levenberg–Marquardt Algorithm Backpropagation Neural Network Training in Dynamic Model Identification of*

Lead City University DO NOT COPY

Mechanical Systems. **Journal of Dynamic Systems, Measurement & Control**, 139(3), 2017, 031012.

50. I. Goodfellow et al., *Generative Adversarial Nets.* In **NIPS** 27, 2014, 2672–2680.

51. A. Figueira & B. Vaz, *Survey on Synthetic Data Generation and GANs.* **Mathematics**, 2022, 2733.

52. A. Radford, L. Metz & S. Chintala, *Unsupervised Representation Learning with DCGAN.* **arXiv**, 2015.

53. A. Creswell et al., *Generative Adversarial Networks: An Overview.* **IEEE Signal Processing Magazine**, 2018, 53–65.

54. A. Krizhevsky, I. Sutskever & G. E. Hinton, *ImageNet Classification with Deep CNNs.* **Communications of the ACM**, 2017, 84–90.

55. S. Hochreiter & J. Schmidhuber, *Long Short-Term Memory.* **Neural Computation**, 1997, 1735–1780.

56. C. Esteban, S. L. Hyland & G. Rätsch, *Real-Valued Medical Time-Series Generation with RCGANs.* **arXiv**, 2017.

57. W. Lim, K. Chek, B. Lau & C. Lin**.** *Future of GAN for Anomaly Detection in Network Security.* **Computers & Security**, 2024, 103733.

58. A. Dunmore, J. Jang-Jaccard, F. Sabrina & J. Kwak, *A Comprehensive Survey of GANs in Cybersecurity Intrusion Detection.* **IEEE Access**, 2023.

59. I. Goodfellow, *NIPS Tutorial: Generative Adversarial Networks.* **arXiv**, 2016.

60. X. Wu et al**.,** *Top 10 Algorithms in Data Mining.* **Knowledge and Information Systems**, 2008, 1–37.

61. L. Breiman, J. H. Friedman, R. A. Olshen & C. J. Stone, *Classification and Regression Trees.* **Wadsworth**, 1984.

62. W. Y. Loh, *Classification and Regression Trees.* **Wiley Interdisciplinary Reviews**, 2011, 14–23.

63. A. Burkov, *The Hundred-Page Machine Learning Book.* **Burkov Publishing**, 2019.

64. The MathWorks Inc, *Statistics and Machine Learning Toolbox.* **MathWorks**, 2025.

65. Scikit-learn, *Machine Learning in Python.* 2025. Available no: https://scikit-learn.org/stable

Lead City University DO NOT COPY

66. A. O. Sangodoyin, *Design and Analysis of Anomaly Detection and Mitigation Schemes.* **PhD Dissertation**, 2019.

67. M. O. Akinsolu & K. Zribi, *A Generalized Framework for Adopting Regression-Based Predictive Modeling in Manufacturing.* **Inventions**, 8(1), 2023, 32.

68. F. Wilcoxon, *Individual Comparisons by Ranking Methods.* In **Breakthroughs in Statistics**, 1992, 196–202.

69. C. Forbes, M. Evans, N. Hastings & B. Peacock, *Statistical Distributions.* **Wiley**, 2011.

70. P. Memar & F. Faradji, *A Novel Multi-Class EEG-Based Sleep Stage Classification System.* **IEEE Transactions on Neural Systems and Rehabilitation Engineering**, 2018, 84–95.

71. M. T. Augustine, *A Survey on Universal Approximation Theorems.* **arXiv**, 2024.

72. I. Brown & C. Mues**.** *Comparison of Classification Algorithms for Imbalanced Credit Scoring.* **Expert Systems with Applications**, 2012, 3446–3453.

73. S. Bhardwaj & M. Dave, *Enhanced Neural-Network-Based Attack Investigation Framework for Network Forensics.* **Computers & Security**, 2023, 103521.

Lead City University DO NOT COPY.

Chapter Four

Result and Discussion of Findings

To determine their representativeness of the modelled SDN-reliant IoT network with regard to the performance metrics and the network scenarios taken into consideration, IoT_{LHS} and IoT_{GAN} are validated before being used for the ML-driven SA conducted in this study. According to the DOE-GAN-SA framework, the validation process and the ML-driven SA implementation that followed are described in detail below.

4.1. Validation of Generated Synthetic Data

To evaluate the reliability of the generated synthetic datasets (IoT_{LHS} and IoT_{GAN}), a systematic comparison was undertaken against the originally simulated dataset (IoT_{DB}) using both descriptive statistical measures and supervised learning techniques. The statistical properties of IoT_{DB}, IoT_{LHS} and IoT_{GAN} are summarised in Tables 4.1–4.4, while their distributions are illustrated in Figures 4.1–4.4. Examination of these tables and figures indicates that both synthetic datasets (i.e., IoT_{LHS} and IoT_{GAN}) are capable of producing instances of throughput, jitter, and response time (T_p^{LHS} , J_t^{LHS}, R_t^{LHS}, T_p^{GAN} , J_t^{GAN}, and R_t^{GAN} respectively) that closely reflect the characteristics of the

Lead City University DO NOT COPY

original dataset. These synthetic instances can therefore be applied to augment the baseline metrics, enhancing the emulation of real-time traffic scenarios in the modelled SDN-enabled IoT environment. To strengthen the robustness of the comparison, the analysis was performed using the median values derived from 50 independent generations of IoT_{LHS} and IoT_{GAN}, rather than single-sample outputs, as outlined previously. Despite this conservative approach, both synthetic datasets exhibit close correspondence with IoT_{DB} in terms of their statistical distributions and observed patterns. Across all network scenarios assessed, the descriptive statistics and visual distributions of IoT_{LHS}, IoT_{GAN}, and IoT_{DB} demonstrate a consistent alignment, thereby confirming the reliability of the synthetic datasets for subsequent performance analysis (Tables 4.1–4.4; Figures 4.1–4.4).

Table 4.1. Descriptive statistics of metrics over 900 observations (normal network instance).

Metric	Minimum	Maximum	Mean	Median	Standard Deviation
$\mathbf{T_p}$	95.1000	95.9000	95.6332	95.6000	0.1402
$\mathbf{T_p^{LHS}}$	95.5206	95.7434	95.6332	95.6338	0.0270
$\mathbf{T_p^{GAN}}$	95.5960	95.7011	95.6253	95.6066	0.0347
$\mathbf{J_t}$	0.0040	0.4930	0.2271	0.1940	0.0943
$\mathbf{J_t^{LHS}}$	0.1528	0.2888	0.2264	0.2256	0.0212
$\mathbf{J_t^{GAN}}$	0.1604	0.2715	0.1898	0.2237	0.0153

Lead City University DO NOT COPY

$\mathbf{R_t}$	0.0320	2.1200	0.2114	0.1980	0.1286
$\mathbf{R_t^{LHS}}$	0.1165	0.6057	0.2116	0.1918	0.0650
$\mathbf{R_t^{GAN}}$	0.1252	0.2497	0.1905	0.1900	0.0189

Source: Author's Computation 2026

Table 4.2. Descriptive statistics of metrics over 900 observations (TCP DDoS attack network instance).

Metric	Minimum	Maximum	Mean	Median	Standard Deviation
$\mathbf{T_p}$	0.0000	95.9000	0.5441	0.0238	7.0999
$\mathbf{T_p^{LHS}}$	0.0000	27.5431	0.5332	0.0150	3.0105
$\mathbf{T_p^{GAN}}$	-0.0822	0.0884	0.0028	0.0019	0.0252
$\mathbf{J_t}$	0.0040	0.4930	0.2271	0.1940	0.0943
$\mathbf{J_t^{LHS}}$	0.1604	0.2878	0.2274	0.2264	0.0207
$\mathbf{J_t^{GAN}}$	0.1579	0.2750	0.1963	0.1946	0.0164
$\mathbf{R_t}$	0.2650	678.0000	302.2676	299.0000	110.6598
$\mathbf{R_t^{LHS}}$	2.9487	413.3743	302.6604	308.0122	69.2561

Lead City University DO NOT COPY

$\mathbf{R_t^{GAN}}$	275.1999	364.6439	304.1860	303.4058	11.935

Source: Author's Computation 2026

Table 4.3. Descriptive statistics of metrics over 900 observations (UDP DDoS attack network instance).

Metric	Minimum	Maximum	Mean	Median	Standard Deviation
$\mathbf{T_p}$	95.1000	95.9000	95.6332	95.6000	0.1402
$\mathbf{T_p^{LHS}}$	95.5138	95.7334	95.6331	95.6341	0.0258
$\mathbf{T_p^{GAN}}$	95.5955	95.7007	95.6118	95.6033	0.0249
$\mathbf{J_t}$	9.1610	18.4280	10.5100	10.1725	1.0496
$\mathbf{J_t^{LHS}}$	9.9098	12.2078	10.5090	10.4302	0.3390
$\mathbf{J_t^{GAN}}$	10.0758	10.2915	10.1737	10.1733	0.0305
$\mathbf{R_t}$	0.1980	82.1000	26.3097	24.8000	7.3245
$\mathbf{R_t^{LHS}}$	23.9558	62.7044	26.3115	24.7773	5.5788
$\mathbf{R_t^{GAN}}$	24.3073	25.5330	24.8590	24.8620	0.2098

Source: Author's Computation 2026

Lead City University DO NOT COPY

Table 4.4. Descriptive statistics of metrics over 900 observations (HTTP DDoS attack network instance).

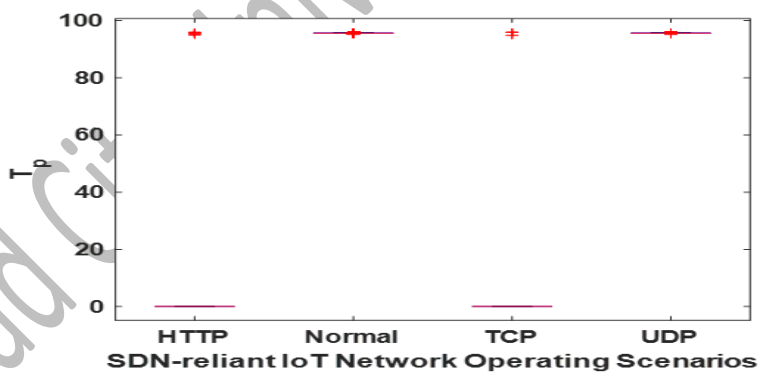
Metric	Minimum	Maximum	Mean	Median	Standard Deviation
$\mathbf{T_p}$	0.0000	95.9000	0.7429	0.0000	8.3955
$\mathbf{T_p^{LHS}}$	0.0000	35.7642	0.8434	2.685×10−5	4.6874
$\mathbf{T_p^{GAN}}$	-0.1042	0.0704	-0.0087	-0.0078	0.0283
$\mathbf{J_t}$	0.0040	0.4930	0.2271	0.1940	0.0943
$\mathbf{J_t^{LHS}}$	0.1617	0.3030	0.2259	0.2253	0.0214
$\mathbf{J_t^{GAN}}$	0.1621	0.2572	0.1957	0.1931	0.0169
$\mathbf{R_t}$	0.0200	1637.0000	49.1262	23.7000	90.9398
$\mathbf{R_t^{LHS}}$	0.0200	180.8001	48.8627	36.3924	42.0700
$\mathbf{R_t^{GAN}}$	10.2506	44.7136	24.0585	23.7436	5.0824

Source: Author's Computation 2026

Lead City University DO NOT COPY

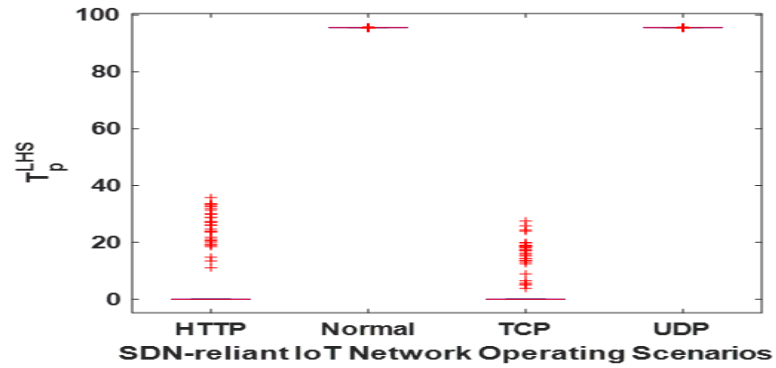
Drawing on the results reported in Tables 4.1–4.4 and illustrated in Figure 4.1, several observations can be made regarding the behaviour of T_p, T_p^{LHS}, and T_p^{GAN}. First, a DDoS flooding attack exerts a clear impact on T_p, T_p^{LHS}, and T_p^{GAN}, altering their normal patterns of behaviour. Second, when the network is subjected to a UDP DDoS attack, the distributions of tT_p, T_p^{LHS}, and T_p^{GAN} remain

largely unchanged, indicating limited disruption under this specific attack type. Third, in contrast, both TCP and HTTP-DDoS attacks produce marked shifts in the distributions of T_p, T_p^{LHS}, and T_p^{GAN}, with values predominantly clustering around 0. This outcome represents a significant departure from the normal operating range of approximately 95 to 96, which is preserved under baseline conditions and during UDP flooding. From a practical perspective, these findings suggest that T_p, T_p^{LHS}, and T_p^{GAN} are particularly vulnerable to TCP and HTTP attacks. This heightened susceptibility reflects the operational mechanisms of these attacks, which rely on overwhelming the target server with excessive TCP connection requests or large volumes of HTTP requests . Such actions deplete network and server resources, leading to service unavailability and denial-of-service for legitimate users [1,2,3].

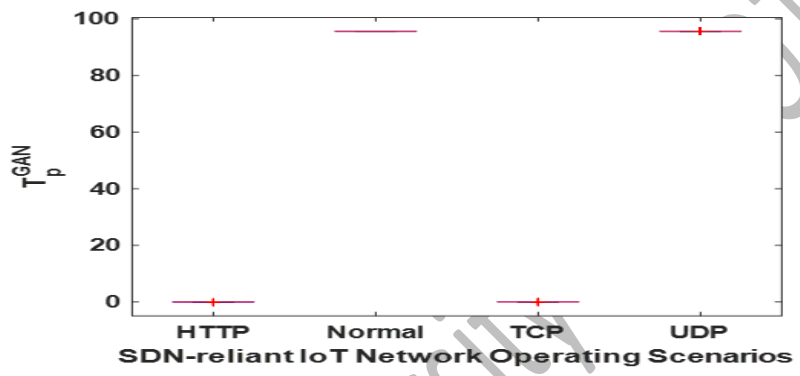


(a) T_p against normal and DDoS flooding attacks

Lead City University DO NOT COPY



(b) T_p^{LHS} against normal and DDoS flooding attack



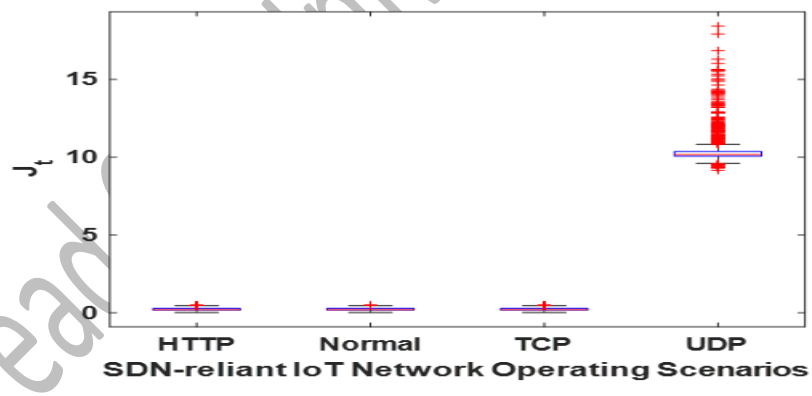
(c) T_p^{GAN} against normal and DDoS flooding attack scenarios.

Figure 4.1. Box plots illustrating the distributions of original and synthetic throughput metrics across all scenarios in the modelled SDN-reliant IoT network.

The results presented in Tables 4.1–4.4 and Figure 4.2 yield several important conclusions regarding the behaviour of J_t, J_t^{LHS}, and J_t^{GAN}. First, DDoS attacks demonstrably affect these indicators, leading to observable deviations from their baseline patterns. Second, the distributions of J_t, J_t^{LHS}, and J_t^{GAN} remain relatively stable when the network is subjected to TCP or HTTP-based DDoS attacks, suggesting limited disruption under these attack types. Third, in contrast, UDP-based DDoS attacks produces pronounced variations in the distributions, with the majority of values clustering around 10. This stands in sharp contrast to the normal operating range of

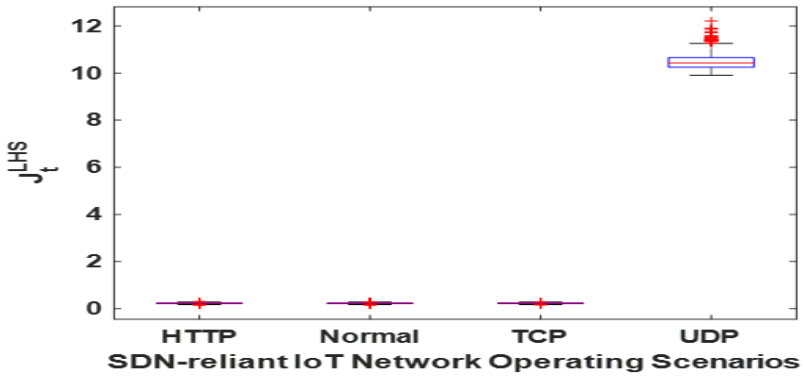
Lead City University ... T COPY

approximately 0 to 0.3, which is preserved under baseline, TCP, and HTTP flooding scenarios. The behaviour of J_t, in particular, is closely tied to the timing and sequence of packet arrivals. Elevated J_t values arise when packets are delivered out of order or arrive in bursts separated by irregular intervals. From a practical standpoint, the evidence in Figure 4.2 and Tables 4.1–4.4 suggests that J_t, J_t^{LHS}, and J_t^{GAN} are especially vulnerable to UDP DDoS attacks. This susceptibility reflects the operational characteristics of such attacks, which circumvent the connection-oriented handshake and overwhelm the target with unsolicited traffic, thereby consuming network resources without requiring prior consent from the server [1,3]..

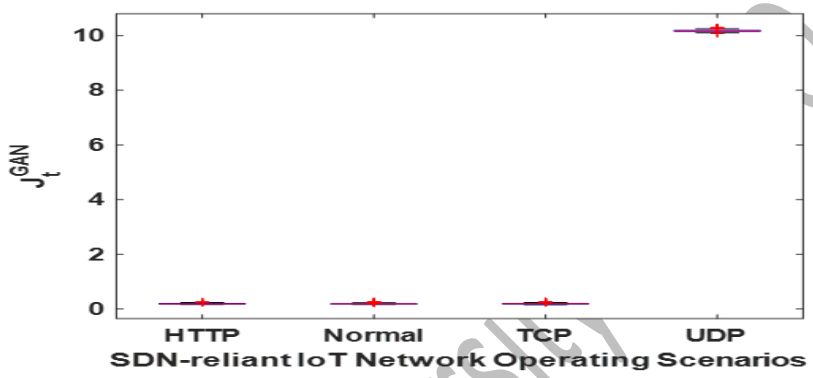


(a) J_t against normal and DDoS flooding attack scenarios.

Lead University DO NOT COPY



(b) J_t^{LHS} against normal and DDoS flooding attack scenarios.



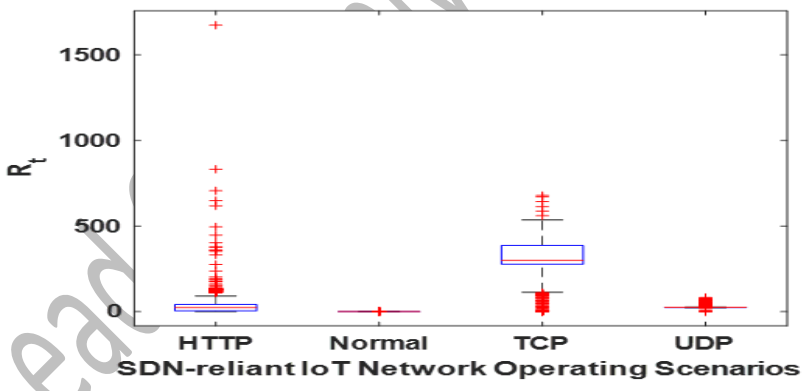
(c) J_t^{GAN} against normal and DDoS flooding attack scenarios.

Figure 4.2. Box plots illustrating the distributions of original and synthetic jitter metrics across all scenarios in the modelled SDN-reliant IoT network.

The results reported in Tables 4.1–4.4 and illustrated in Figure 4.3 provide several insights into the behaviour of R_t, R_t^{LHS}, and R_t^{GAN}. It is evident that DDoS attacks exert measurable effects on these metrics, with their distributions showing substantial deviations from normal operating behaviour. Under TCP DDoS attacks, values are concentrated within the range of approximately 0 to 500, while UDP DDoS attacks produces distributions between 10 to 50, and HTTP DDoS attacks results in values clustered around 0 to 200. These patterns contrast sharply with the narrow

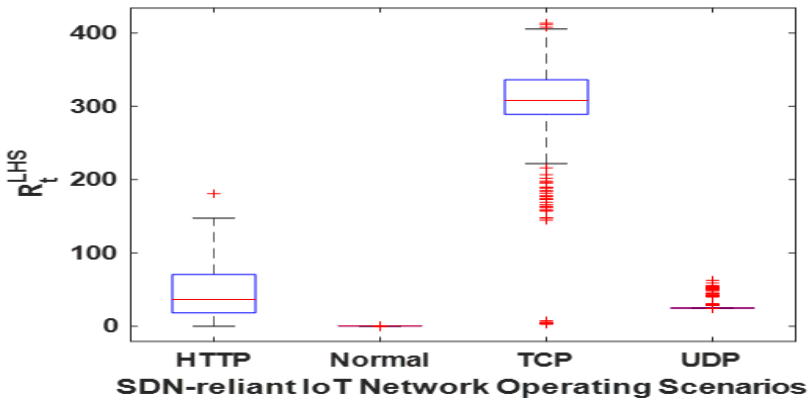
Lead City University COPY

range of approximately 0 to 0.5 observed under normal conditions. From a practical perspective, the evidence suggests that R_t, R_t^{LHS}, and R_t^{GAN} are vulnerable to all three forms of DDoS attacks. This vulnerability stems from the operational dynamics of the attacks, which disrupt normal packet handling and consume network resources [1,2,3,4]. Among these metrics, R_t emerges as particularly critical, as it can be severely degraded during DDoS events. This is especially consequential for applications reliant on acknowledgment-based communication prior to the transmission of subsequent packets. Under such conditions, unified communication systems operating within SDN-reliant IoT environments may experience considerable disruption, underscoring the importance of monitoring response time as a key indicator of service degradation.

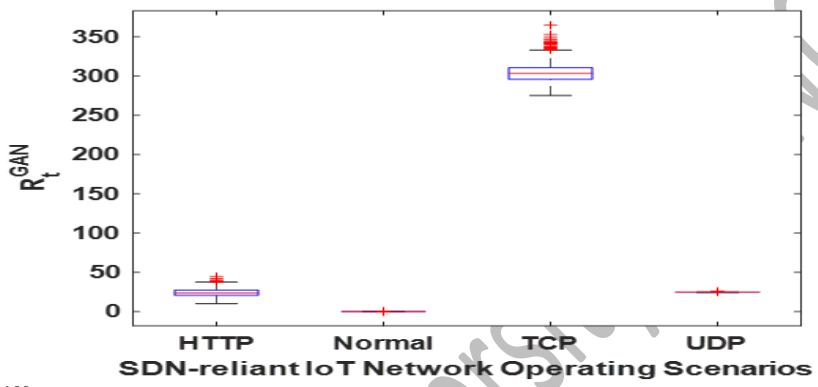


(a) R_t against normal and DDoS flooding attack scenarios.

Leading University DO NOT COPY



(b) R_t^{LHS} against normal and DDoS flooding attack scenarios.



(c) R_t^{GAN} against normal and DDoS flooding attack scenarios.

Figure 4.3. Box plots illustrating the distributions of original and synthetic response time metrics across all scenarios in the modelled SDN-reliant IoT network.

Closer inspection of Tables 4.1 to 4.4 indicates that the mean and median values of the corresponding metrics in IoT_{DB}, IoT_{LHS} and IoT_{GAN} remain largely consistent, with only minimal or acceptable deviations. An exception is observed under the HTTP DDoS attack scenario, where larger discrepancies occur. These deviations can be attributed to the inherent distributional properties of T_p and R_t during HTTP DDoS attack, where distinct outliers are present (see Figures 4.1 and 4.3). Figures 4.1 to 4.4 further demonstrate that, under normal network conditions, the distributions of IoT_{DB}, IoT_{LHS} and IoT_{GAN} across the three datasets show close alignment. Although DDoS attack scenarios introduce some variation, the overall distributions remain

Lead City University ... NOT COPY

reasonably consistent. Moreover, similar to throughput, both IoT_{DB}, IoT_{LHS} and IoT_{GAN} effectively differentiate between normal and DDoS attack conditions, as evidenced in Figures 4.1 to 4.4. This distinguishing capability is particularly beneficial for the implementation of ADC, which is examined in greater detail in Section 4.1.

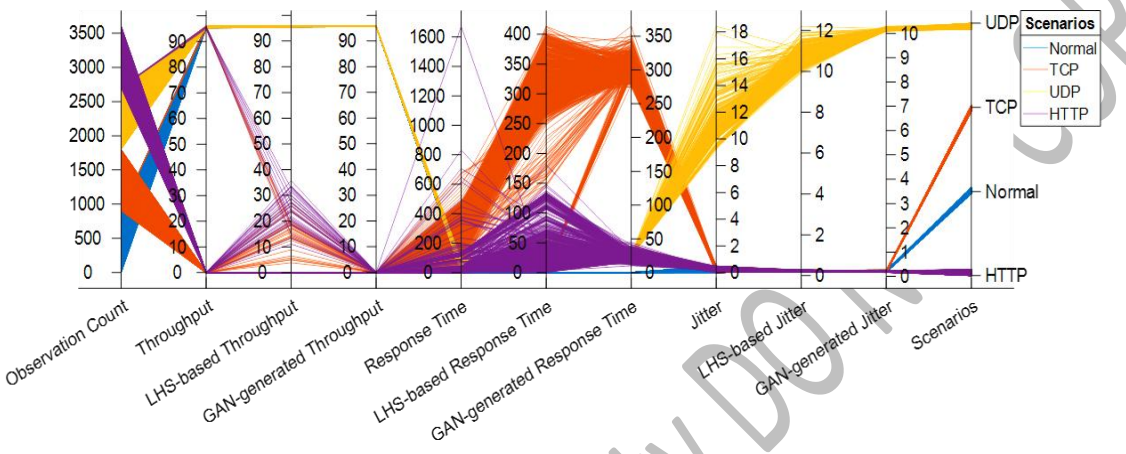


Figure 4.4. Parallel coordinates plots of simulated, LHS-based, and GAN-generated T_p, J_t and R_t metrics across all network scenarios.

Source: Author's Computation 2026

4.1.1. Hypothesis Testing and Comparisons for the Validation of Generated Datasets

In instances where discrepancies arise between the synthetic datasets (IoT_{LHS} and IoT_{GAN}) and the original dataset IoT_{DB}, it is instructive to examine their combined behaviour. Considered together, the synthetic datasets are expected to capture distributional patterns more effectively than either dataset alone, highlighting their complementary nature. This makes them particularly valuable for augmenting $IoT_{DB}$and enhancing its applicability in predictive modelling of diverse scenarios within SDN-reliant IoT networks. To evaluate this expectation, supervised learning was employed to assess the classification of SDN-reliant IoT network scenarios using IoT_{DB} and the combined

Lead City University DO NOT COPY

synthetic datasets (IoT_{LHS} and IoT_{GAN}). The chosen supervised learning method was CART, selected for the reasons outlined previously (see Section 3.6.3.4). The implementation framework and experimental settings also followed those described earlier in Section 3.6.3.4. For the comparison, 50 CART models were constructed using IoT_{DB} and another 50 using the combined (IoT_{LHS} and IoT_{GAN}) dataset. The descriptive statistics of predictive accuracy P_A and F1-score were then compared across the two sets of models, with particular emphasis on anomaly detection and classification (ADC). Predictive accuracy is defined as the ratio of correctly classified observations to the total number of classified observations. In this study, P_A values for the CART classifiers were derived from their respective confusion matrices. Figure 9 presents the confusion matrices for representative CART models out of the 50 trained on each dataset IoT_{DB} and (IoT_{LHS} and IoT_{GAN}). The P_A of a given CART classifier can be computed from the confusion matrix according to the following expression [5,6]:

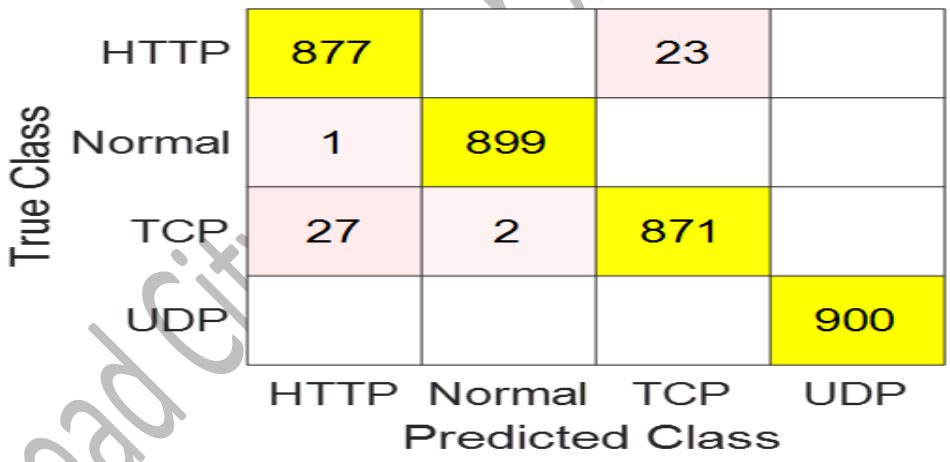
$$\mathrm{P_A} = \frac{\mathrm{TP + TN}}{\mathrm{TP + TN + FP + FN}}$$

Here, *TP* denotes a true positive, referring to observations correctly classified as belonging to class $O*$; *TN* represents a true negative, referring to observations correctly classified as not belonging to class $O*$; *FP* indicates a false positive, where observations are incorrectly classified as $O*$ when they do not belong to that class; and *FN* denotes a false negative, where observations are incorrectly classified as not $O*$ despite actually belonging to it.

The descriptive statistics of predictive accuracy P_A across 50 independent statistical runs, each involving the construction of a CART model using IoT_{DB} and the combined dataset IoT_{LHS} and

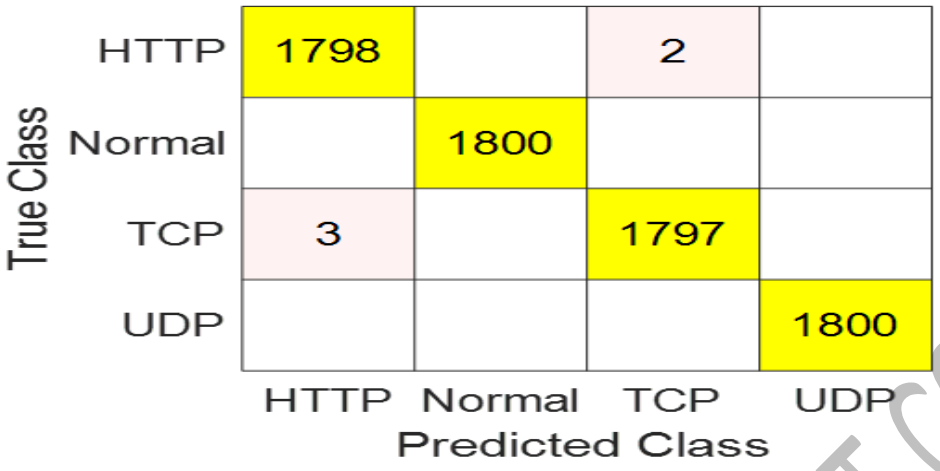
Lead City University DO NOT COPY

IoT_{GAN}, are summarised in Table 4.5. In line with the representative confusion matrices shown in Figure 4.5, the results demonstrate that P_A remains consistently high for both datasets, indicating their suitability for CART-based ADC within the modelled SDN-reliant IoT network, even under worst-case conditions. Notably, the P_A achieved using the combined synthetic dataset (IoT_{LHS} and IoT_{GAN}) is marginally higher than that obtained with IoT_{DB}. This suggests that the synthetic datasets may introduce additional variability or structural features that improve the capacity of the classifier to discriminate between scenarios, likely owing to the greater volume of observations available. Moreover, the very low standard deviations observed for both datasets reinforce their reliability and confirm their effectiveness in supporting CART-based ADC across the network scenarios under consideration.



(a) IoT_{DB}

Lead City University DO NOT COPY



(b) Combination of IoT_{LHS} and IoT_{GAN}

Figure 4.5. Confusion matrices for the CART models constructed.

Source: Author's Computation 2026

Table 4.5. Descriptive statistics of P_A over 50 independent runs.

Dataset	Worst	Best	Mean	Median	Standard Deviation
IoT_{DB}	0.9836	0.9897	0.9871	0.9872	0.0012
IoT_{LHS} and IoT_{GAN}	0.9983	0.9999	0.9993	0.9993	3.1348×10^{-4}

Source: Author's Computation 2026

The CART models achieved a classification accuracy of 99.93 % on the combined IoT_{LHS} and IoT_{GAN} dataset. This result shows stable performance across all 50 runs. The best accuracy reached

Lead City University DO NOT COPY

0.9999. The worst accuracy remained high at 0.9983. The standard deviation stayed low at 3.1348 × 10^{-4}. This shows consistent behaviour with limited variation. Since all CART models generate confidence scores (posterior probabilities) for their predictions, receiver-operating characteristic (ROC) curves and the corresponding areas under the curve (AUCs) can also be employed to analyse and summarise their P_A values [5,6] . Figure 4.6 presents the ROC curves for representative CART models selected from the 50 runs constructed using both $IoT_{DB}$and the combined dataset IoT_{LHS} and IoT_{GAN}. In this context, the confidence scores produced by each CART model are discretised to derive prediction thresholds. At each threshold, the sensitivity also referred to as recall or true positive rate (TPR) and the false positive rate (FPR) are calculated and combined to construct the ROC curve and the associated P_A summary. The observations within the dataset are then classified using these threshold values, thereby providing insight into model performance across varying decision boundaries. For a given ROC curve, the true positive rate (TPR), false positive rate (FPR), positive predictive value (PPV), and the resulting F1-score are estimated according to the following expressions [5,6]:

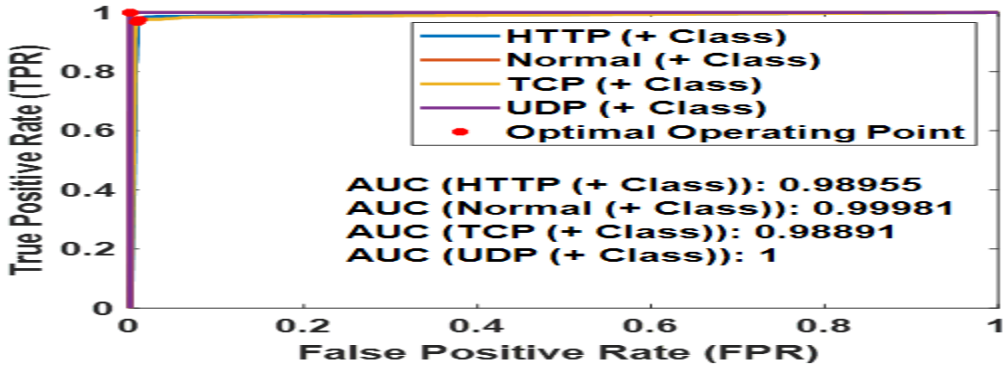
$$\text{TPR or Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

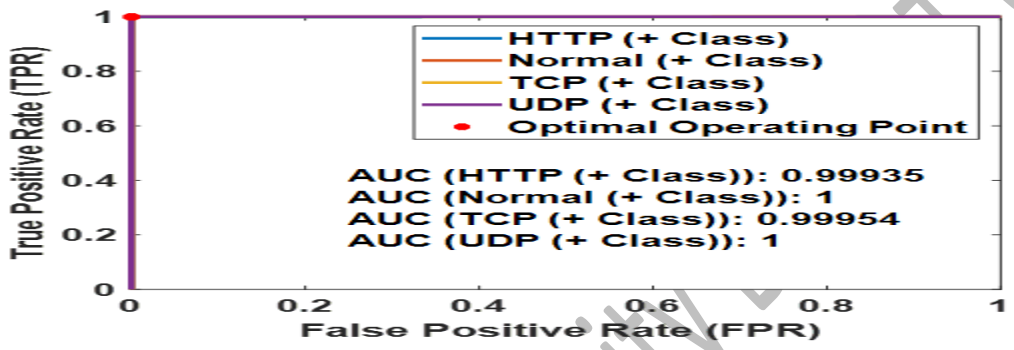
$$\text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{F1} = 2 \times \frac{\text{PPV X TPR}}{\text{PPV} + \text{TPR}}$$

Lead City University DO NOT COPY



(a) IoT_{DB}



(b) Combination of IoT_{LHS} and IoT_{GAN}

Figure 4.6. Typical ROC curves for the CART models constructed.

Source: Author's Computation 2026

The descriptive statistics of the F1-scores across 50 independent runs each involving the construction of a CART model using $IoT_{DB}$and the combined dataset IoT_{LHS} and IoT_{GAN}, are summarised in Table 4.6. The combined dataset achieved an F1-score of 1.000 across all 50 runs. Table 4.6 shows that the worst, best, mean and median values were identical, with a standard deviation of 0. This confirms perfect stability. The IoT_{DB} dataset also produced a stable F1-score of 0.9994. The combined dataset delivered the strongest performance. The synthetic observations increased data volume and improved class separation, which explains the perfect F1-score. In line

Lead City University COPY

with the representative ROC curves shown in Figure 4.6, the results demonstrate that F1-scores remain consistently high for both datasets. This finding indicates that each is suitable for CART-based ADC within the modelled SDN-reliant IoT network, even under worst-case conditions. Notably, the F1-score achieved using the combined synthetic dataset (IoT_{LHS} and IoT_{GAN}) is marginally higher than that obtained with IoT_{DB}. This suggests that the inclusion of synthetic observations may provide additional discriminative features that improve classification performance, likely due to the increased data volume. Moreover, the very low standard deviations observed across both datasets further reinforce their reliability and confirm their suitability for CART-based ADC in the network scenarios examined.

Leadcity University DO NOT COPY

Table 4.6. Descriptive statistics of F1 over 50 independent runs.

Dataset	Worst	Best	Mean	Median	Standard Deviation
$\mathbf{IoT_{DB}}$	0.9994	0.9994	0.9994	0.9994	0.0000
$\mathbf{IoT_{LHS}}$ **and** $\mathbf{IoT_{GAN}}$	1.0000	1.0000	1.0000	1.0000	0.0000

Source: Author's Computation 2026

To statistically verify whether the CART models constructed using the combined dataset IoT_{LHS} and IoT_{GAN} are more robust for ADC within the modelled SDN-reliant IoT network compared to

those built solely on the original dataset IoT_{DB}, a Wilcoxon rank-sum test [7] was conducted. For this test, the P_A and F1-scores obtained from 50 independent runs of CART models trained on each dataset were treated as data samples. The null hypothesis (H_0) stated that the median values of the two data samples are equal at the 5% significance level (i.e., 95% confidence level), while the alternative hypothesis (H_1) proposed that the medians differ. The null hypothesis was rejected if the two-sided probability value (p-value) of the test was less than or equal to 0.05, indicating sufficient evidence against median equality. As shown in Table 4.7, the observed improvements in P_A and F1-scores for CART models trained on the combined dataset (IoT_{LHS} and IoT_{GAN}), relative to those trained on IoT_{DB}, are statistically significant. Specifically, p-values of 6.0493 $\times 10^{-18}$ (for PA) and 2.9193 $\times 10^{-22}$ (for F1-score) confirm the superiority of the combined dataset in supporting robust classification. These findings suggest that synthetic data generated from IoT_{LHS} and IoT_{GAN} closely mirror the behavioural characteristics of the modelled SDN-reliant IoT network under both normal conditions and DDoS attack scenarios. Importantly, the combined use of synthetic and original data enhances dataset richness and improves the fidelity of network simulations. Building on these results, the next section presents a comprehensive Machine Learning (ML) Driven Sensitivity Analysis (SA) using the augmented dataset denoted as IoT_{DB} + (IoT_{LHS} and IoT_{GAN}), which also equals IoT_{AUG} .

Table 4.7. Hypothesis test: IoT_{DB} vs. the combined datasets of IoT_{LHS} and IoT_{GAN}.

Metric	***p*-Value**
P_A	6.0493 $\times 10^{-18}$
F1	2.9193 $\times 10^{-22}$

Source: Author's Computation 2026

Lead City University DO NOT COPY

4.2. ML-Driven SA

The ML-driven SA component of the DOE-GAN-SA framework was implemented as outlined in Section 3.7. Figure 4.7 depicts the trend of C_F^w across all scenarios in the modelled SDN-reliant IoT network, taking IoT_{AUG}^{norm} into consideration. Using C_F^w as the response or target variable for the ANN models to be constructed allows, comparatively, an extra layer of differentiation between various scenarios in the modelled SDN-reliant IoT network, as shown in Figure 4.7. In particular, C_F^w shows greater values when the network is exposed to DDoS flooding attacks, whereas it approaches or stays around nil when the network functions smoothly. All ANN models were constructed using the same architecture, consisting of three input nodes, ten hidden layers, and a single output node, as shown in Figure 4.8. Based on this configuration, a total of 50 ANN models were trained over 50 independent statistical runs, with IoT_{AUG}^{norm} as the target variable. The training details of a representative ANN model including the MSE trend, error histogram, gradient plot, and regression plots are presented in Figures 4.9, 4.10, 4.11, and 4.12, respectively.

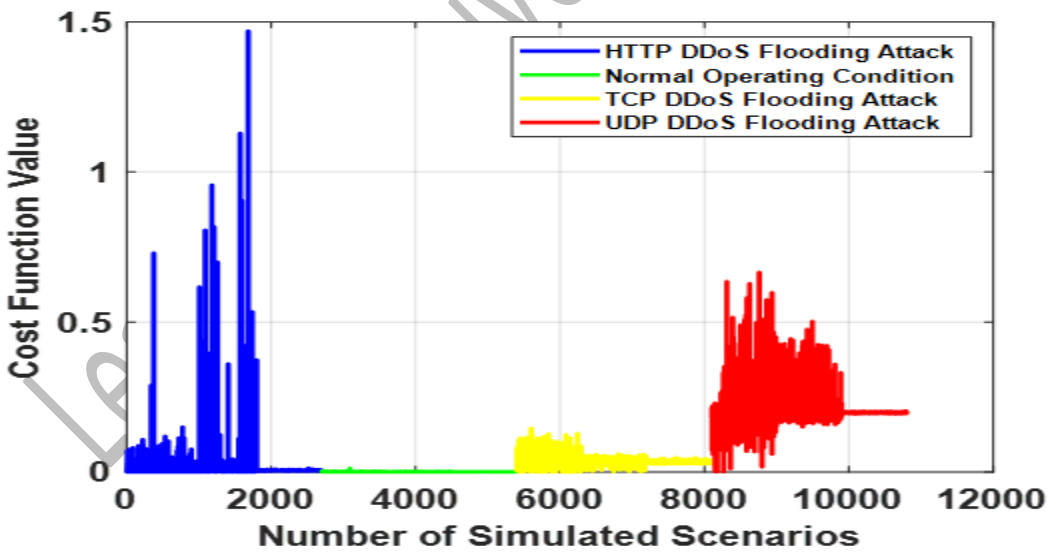


Figure 4.7. C_F^w plot against IoT network scenarios.

iversity DO NOT COPY

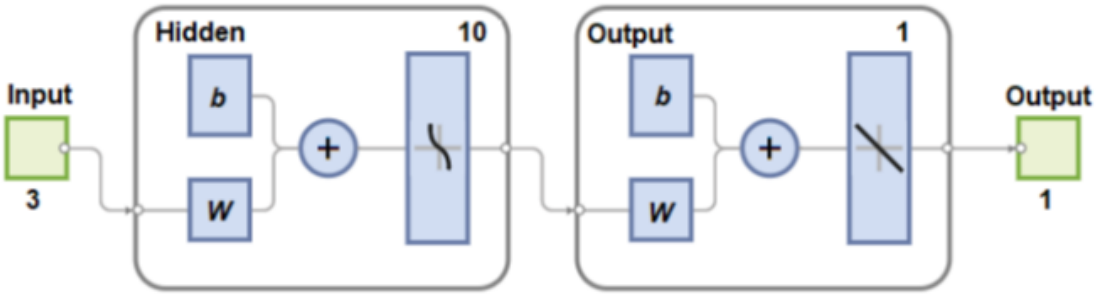


Figure 4.8. Layout of the ANN models that was built.

Source: Author's Computation 2026

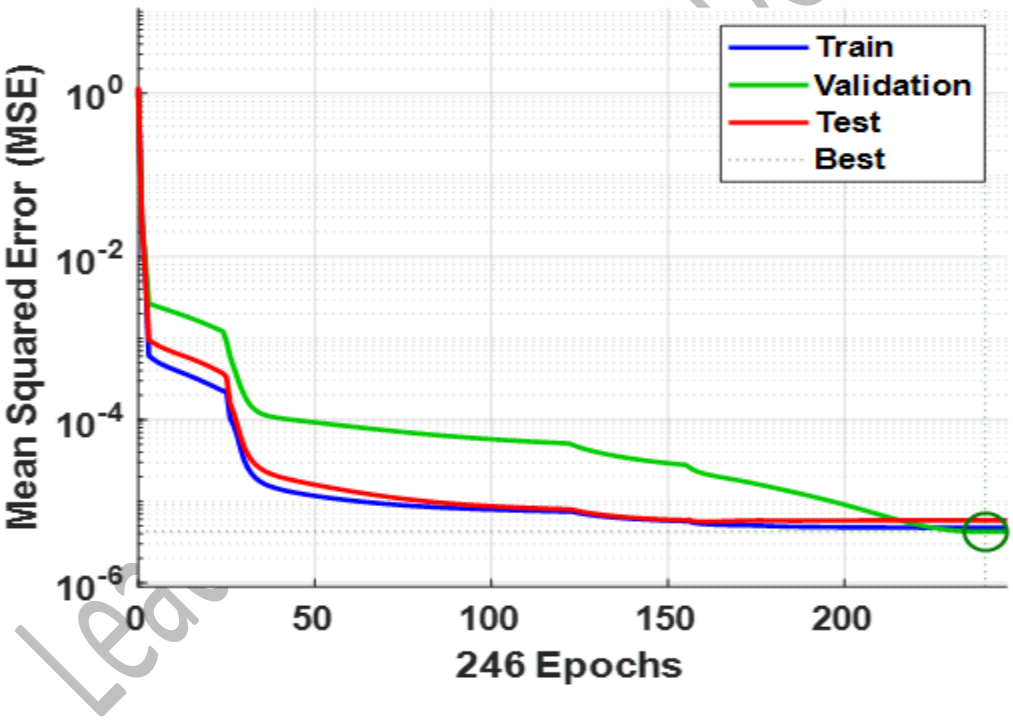


Figure 4.9. Convergence trend of the MSE values for the ANN models constructed using IoT_{AUG}^{norm} (the best validation performance is 4.2194×10^{-6} at epoch 240).

Source: Author's Computation 2026

DO NOT COPY

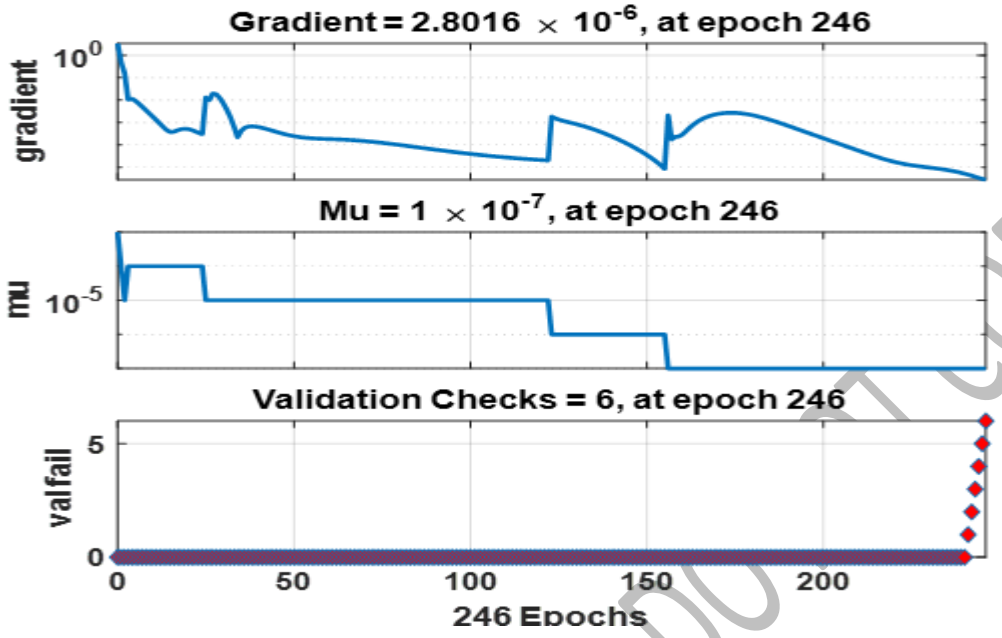


Figure 4.10 Gradient information trend for the ANN models constructed using IoT_{AUG}^{norm}.

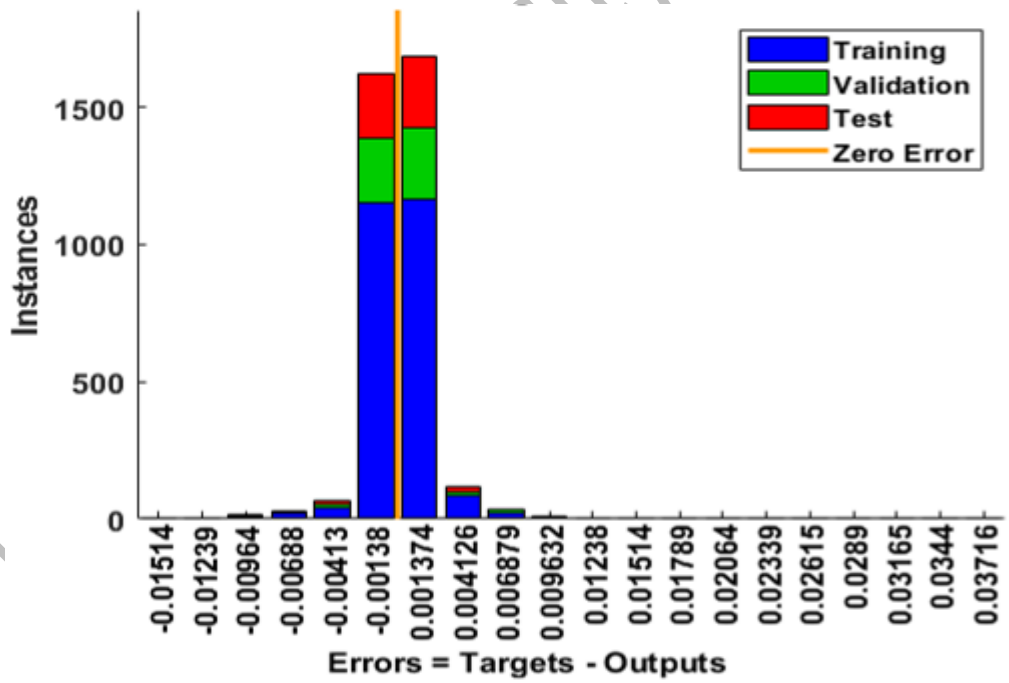


Figure 4.11. Error histogram (with 20 bins) for the ANN models constructed using IoT_{AUG}^{norm}.

Source: Author's Computation 2026

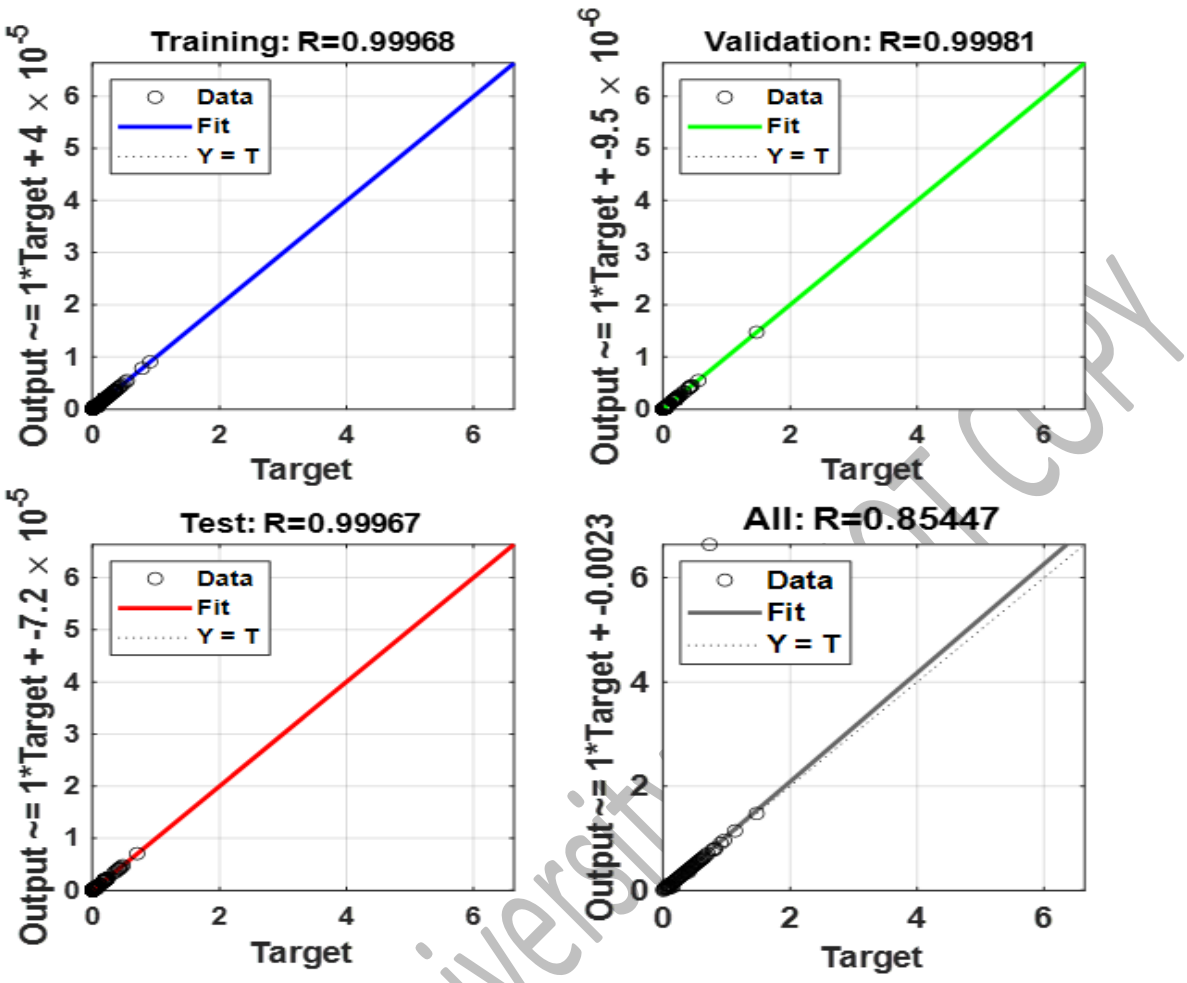


Figure 4.12. Plots of regression for ANN models constructed using IoT_{AUG}^{norm}.

Source: Author's Computation 2026

Figures 4.9 and 4.10 indicate that the training of the ANN models was computationally efficient, as convergence to low MSE values was generally achieved within fewer than 250 epochs. The overall performance of the models can be regarded as satisfactory, given their consistently low error rates and high correlation coefficients, as illustrated in Figures 4.11 and 4.12. As outlined in Section 3.7, the sensitivity of the performance metrics for the modelled SDN-reliant IoT network (T_p, J_t, and R_t) was evaluated by re-simulating the ANN models trained with IoT_{AUG}^{norm}. This process was repeated across 50 independent statistical runs using $IoT_{AUG}^{T_p^{noisy}}$, $IoT_{AUG}^{J_t^{noisy}}$, and

$IoT_{AUG}^{R_t^{noisy}}$ as input variables, respectively. The descriptive statistics of the resulting MSE values for all datasets employed in the ANN simulations are summarised in Table 4.8.

Lead City University DO NOT COPY

Table 4.8. Descriptive statistics of the MSE values of the ANN models built over 50 independent runs.

Dataset	Worst	Best	Mean	Median	Standard Deviation
IoT_{AUG}^{norm} (MSE_{norm})	0.0041	1.0274×10^{-5}	9.1407×10^{-4}	3.5181×10^{-4}	0.0012
$IoT_{AUG}^{T_p^{noisy}}$ $(MSE_{noisy}^{T_p})$	0.0302	0.0108	0.0200	0.0193	0.0520
$IoT_{AUG}^{J_t^{noisy}}$ $(MSE_{noisy}^{J_t})$	0.8255	0.0146	0.1211	0.0854	0.1326
$IoT_{AUG}^{R_t^{noisy}}$ $(MSE_{noisy}^{R_t})$	22.3078	0.0109	6.9898	6.0691	5.0983

Source: Author's Computation 2026

Table 4.8 provides several important insights. First, the MSE values of the ANN models demonstrate clear sensitivity to variations in T_p, J_t, and R_t when comparing MSE_{norm} with $MSE_{noisy}^{T_p}$, $MSE_{noisy}^{J_t}$, and $MSE_{noisy}^{R_t}$, respectively. Second, among the three parameters, R_t emerges as the most sensitive, with a mean MSE value of 6.9898 for $MSE_{noisy}^{R_t}$, while T_p is the least sensitive, with a mean MSE value of 0.0200 for $MSE_{noisy}^{T_p}$. Third, under best-case conditions, T_p induced the least change in MSE_{norm}, recording a minimum MSE of 0.0108 for $MSE_{noisy}^{T_p}$ a result closely aligned with R_t, which produced a minimum MSE of 0.0109 for $MSE_{noisy}^{R_t}$. By contrast, J_t produced the largest best-case deviation, with a minimum MSE of 0.0146 for

Lead City University DO NOT COPY

$MSE_{noisy}^{J_t}$. Fourth, under worst-case conditions, R_t caused the greatest disruption, yielding a maximum MSE of 22.3078 for $MSE_{noisy}^{R_t}$, while response time caused the least disruption to MSE_{norm}, with a maximum MSE of 0.0302 for $MSE_{noisy}^{T_p}$. Finally, in terms of robustness, $T_p$produced the most consistent changes in MSE_{norm}, with a standard deviation of 0.0520 for $MSE_{noisy}^{T_p}$, whereas R_t generated the most inconsistent outcomes, with a standard deviation of 5.0983 for $MSE_{noisy}^{R_t}$. Taken together, these results demonstrate that although T_p, J_t, and R_t are all sensitive to perturbations, R_t is markedly more sensitive than the other two metrics. This finding is in line with prior studies [1], thereby lending further validity to the methodology adopted in this research.

4.2.1. Hypothesis Testing for Sensitivity Analysis

In line with the hypothesis testing procedure described earlier in Section 4.1, a Wilcoxon rank-sum test[7] was carried out to statistically examine whether MSE_{norm} is influenced by variations in T_p, J_t, and R_t when their noisy counterparts ($IoT_{AUG}^{T_p^{noisy}}$, $IoT_{AUG}^{J_t^{noisy}}$, and $IoT_{AUG}^{R_t^{noisy}}$) are employed to evaluate the ANN models constructed with IoT_{AUG}^{norm}. For this analysis, the values of MSE_{norm}, $MSE_{noisy}^{T_p}$, $MSE_{noisy}^{J_t}$, and $MSE_{noisy}^{R_t}$ obtained over 50 independent statistical runs were treated as data samples. The null hypothesis stated that the median values of MSE_{norm} and those of $MSE_{noisy}^{T_p}$, $MSE_{noisy}^{J_t}$, and $MSE_{noisy}^{R_t}$ are equal at the 5% significance level (95% confidence). The alternative hypothesis stated that their medians differ. As previously noted, the null hypothesis is rejected if the two-sided probability value (p-value) from the Wilcoxon test is less than or equal to 0.05, thereby providing sufficient evidence against median equality.

Lead City University DO NOT COPY

The results in Table 4.9 show that $MSE_{noisy}^{T_p}$, $MSE_{noisy}^{J_t}$, and $MSE_{noisy}^{R_t}$ each yield a p-value of 7.0661×10^{-18}, indicating strong statistical evidence that they differ significantly fromMSE_{norm}. These findings reinforce the conclusion that T_p, J_t, and R_t represent sensitive performance metrics for the SDN-reliant IoT network under investigation.

Table 4.9. Hypothesis test: Other datasets vs IoT_{AUG}^{norm} (MSE_{norm})

Dataset	***p*-Value**
$\boldsymbol{IoT_{AUG}^{T_p^{noisy}}}$ **(** $\boldsymbol{MSE_{noisy}^{T_p}}$ **)**	7.0661×10^{-18}
$\boldsymbol{IoT_{AUG}^{J_t^{noisy}}}$ **(** $\boldsymbol{MSE_{noisy}^{J_t}}$ **)**	7.0661×10^{-18}
$\boldsymbol{IoT_{AUG}^{R_t^{noisy}}}$ **(** $\boldsymbol{MSE_{noisy}^{R_t}}$**)**	7.0661×10^{-18}

Source: Author's Computation 2026

4.3. Comparisons with Other Approaches

Given the scope of this research and the constraints of time, a full-scale comparative evaluation between the DOE-GAN-SA framework and other existing methods could not be undertaken. However, when weighed against the effectiveness of other approaches documented in the literature [8,9,10,11], the framework's successful application in this study, provides a useful basis for analysis. As outlined in Table 4.10, the DOE-GAN-SA framework demonstrates strong alignment with established methods while simultaneously offering distinctive contributions. In particular, its deliberate integration of Design of Experiments (DOE) with machine learning (ML) techniques enhances the practical relevance of combined techniques in modelling and analysing the behaviour of SDN-enabled IoT networks. In contrast to seeking to supplant existing methodologies, the DOE-

Lead City University DO NOT COPY

GAN-SA framework is positioned as a complementary strategy, addressing specific limitations and supporting a more resilient adoption of ML-driven solutions within this domain.

Table 4.10. Comparative assessment.

Method	Data Augmentation	Anomaly Detection and Classification	Sensitivity Analysis	Scalability	Efficiency	References
Monte Carlo	Not Applicable	Not Applicable	Yes	Yes	High	[9,11]
Variance -based	Not Applicable	Not Applicable	Yes	Yes	High	[8,10]
This work	Yes	Yes	Yes	Yes	High	Not Applicable

Source: Author's Computation 2026

4.4. Recommendations

Although the experiments presented in this study were conducted within controlled emulated environments, the DOE-GAN-SA framework was conceived with direct applicability to real-world SDN-enabled IoT networks. In practical deployments, a representative architectural setup could be implemented to demonstrate how critical performance indicators specifically response time, jitter, throughput can be continuously monitored and embedded within the DOE-GAN-SA workflow to support advanced behavioural modelling and performance evaluation. This direction resonates with earlier work, such as the cost-effective IoT-SDN testbed described in [12], where Raspberry Pi devices were configured as OpenFlow switches and managed by controllers such as Floodlight. This testbed illustrated the potential of dynamic flow management through the

Lead City University DO NOT COPY

separation of control and data planes, which is also a fundamental design principle underpinning DOE-GAN-SA. In the present research, a comparable approach was adopted using a Mininet-based tree topology integrated with the floodlight controller. Performance was examined under DDoS conditions using tools such as iperf and LOIC, allowing direct assessment of system robustness. Findings reported in [12] demonstrated that it is possible to collect metrics in real time, such as throughput, jitter, and bandwidth utilisation, which strengthens the efficacy of SDN-driven traffic management in Internet of Things contexts. Notably, the component-based structure of the testbed supported time-series data collection under diverse network conditions, with particular focus on throughput and jitter. Its hierarchical tree topology also bears resemblance to Fat-Tree design employed in DOE-GAN-SA, chosen for its predictable packet forwarding behaviour and scalability. These parallels highlight the conceptual and structural coherence between the framework developed in this study and contemporary engineering practices aimed at building resilient SDN-IoT systems.

Throughput, defined as the rate of successful data transmission across the network, emerged as a critical performance metric in both the testbed and DOE-GAN-SA. In [12], centralised flow control was shown to improve bandwidth utilisation under varied traffic patterns. Similarly, DOE-GAN-SA recorded throughput under normal and DDoS flooding attack conditions, with observed declines serving as key indicators of network stress and performance degradation. Jitter, reflecting the variability in packet inter-arrival times, was also carefully monitored in both approaches. Stable, low jitter values were associated with reliable quality of service (QoS), while increased variability signalled potential service deterioration. DOE-GAN-SA adopted an analogous strategy,

Lead City University DO NOT COPY

capturing jitter fluctuations under normal and attack scenarios to further demonstrate its capacity to identify anomalies in service delivery.

Both methodologies rely on programmable OpenFlow-enabled switches managed by centralised controllers, which enable holistic monitoring and adaptive routing across the network. However, the DOE-GAN-SA advances beyond empirical testbed studies by integrating data augmentation via Latin Hypercube Sampling (LHS) and Generative Adversarial Networks (GANs), thereby increasing dataset diversity and enhancing model robustness. By contrast, the testbed described in [12] relied primarily on live empirical data. Taken together, these findings underscore the relevance of throughput and jitter as fundamental indicators of SDN-IoT performance, while also illustrating the practical feasibility of implementing the DOE-GAN-SA framework. The complementary insights of empirical testbeds and hybrid data-driven methodologies point to a clear pathway for the practical realisation of resilient and high-performance SDN-IoT systems.

Lead City University DO NOT COPY

Endnotes

1. M. Akinsolu, A. Sangodoyin & U. Uyoata, *Behavioral Study of Software-Defined Network Parameters Using Exploratory Data Analysis and Regression-Based Sensitivity Analysis.* **Mathematics**, 10, 2022. Available online: https://doi.org/10.3390/math10142536.

2. A. Sahi, D. Lai, Y. Li & M. Diykh, *An Efficient DDoS TCP Flood Attack Detection and Prevention System in a Cloud Environment*. **IEEE Access**, 5, 2017, 6036–6048.

3. D. Nashat & S. Khairy, *Detecting HTTP Flooding Attacks Based on Uniform Model.* In **Proceedings of the 2021 International Conference on Networking and Network Applications (NaNA)**, 2021, 94–98.

4. Z. Y. Shen, M. W. Su, Y. Z. Cai & M. H. Tasi, *Mitigating SYN Flooding and UDP Flooding in P4-Based SDN*. **In Proceedings of the 2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS)**, 2021, 374–377.

5. A. O. Sangodoyin, M. O. Akinsolu, P. Pillai & V. Grout, *Detection and Classification of DDoS Flooding Attacks on Software-Defined Networks: A Case Study for the Application of Machine Learning*. **IEEE Access**, 9, 2021, 122495–122508. Available online: https://doi.org/10.1109/ACCESS.2021.3109490.

6. A. Burkov, *The Hundred-Page Machine Learning Book*. **Andriy Burkov**, Québec, QC, 2019.

7. F. Wilcoxon, *Individual Comparisons by Ranking Methods*. **In Breakthroughs in Statistics: Methodology and Distribution**, 1992, 196–202.

8. S. L. Piano, F. Ferretti, A. Puy, D. Albrecht & A. Saltelli, *Variance-Based Sensitivity Analysis: The Quest for Better Estimators and Designs between Explorativity and Economy*. **Reliability Engineering & System Safety**, 206, 2021, 107300.

Lead City University DO NOT COPY

9. A. Nouri, C. van Treeck & J. Frisch, *Sensitivity Assessment of Building Energy Performance Simulations Using MARS Meta-Modeling in Combination with Sobol' Method.* **Energies,** 17, 2024, 695.

10. Q. Liu, N. Tong, X. Wu, X. Han & C. Chen, *A Generalized Sensitivity Analysis Method Based on Variance and Covariance Decomposition of Summatory Functions for Multi-Input Multi-Output Systems.* **Computer Methods in Applied Mechanics and Engineering,** 385, 2021, 114009.

11. D. Biazar, H. Khaloozadeh & M. Siahi, *Sensitivity Analysis for Evaluation of the Effect of Sensors Error on the Wind Turbine Variables Using Monte Carlo Simulation.* **IET Renewable Power Generation,** 16, 2022, 1623–1635.

12. S. A. Arman, M. M. Rahman, S. F. Rahman, N. P. Urmi, P. P. Urmee, N. Muslim & S. Islam, *Developing an IoT Networks-Based Testbed for Software-Defined Networks.* In **Proceedings of the 2020 IEEE Region 10 Symposium (TENSYMP),** 2020, 1752–1755.

Lead City University DO NOT COPY

Chapter Five

Conclusion

5.1 Summary

This study introduced the DOE-GAN-SA framework, a novel methodology that integrates design of experiments (DOE) and machine learning (ML) techniques to advance the behavioral analysis and characterization of software-defined networking (SDN)-enabled Internet of Things (IoT) environments. The framework specifically addressed three core components: data augmentation through DOE and generative adversarial networks (GANs), anomaly detection and characterization (ADC), and sensitivity analysis (SA). By leveraging Latin Hypercube Sampling (LHS), GANs, Classification and Regression Trees (CART), and Artificial Neural Networks (ANNs), the DOE-GAN-SA framework provides a systematic and comprehensive exploration of SDN-IoT network dynamics under diverse operational conditions. Its structured workflow begins with network scenario simulation, followed by synthetic data generation through LHS and GANs, statistical validation, CART-driven anomaly characterization, and ANN-based sensitivity analysis.

This hybrid approach advances conventional SA techniques by consolidating DOE and ML methods within a unified framework, thereby enabling effective data augmentation, ADC, and SA. The reliability of DOE-GAN-SA was further reinforced through statistical validation of synthetic

Lead City University DO NOT COPY

data and supervised learning, confirming its suitability for robust behavioral analysis. Additionally, hypothesis testing was conducted to statistically substantiate key findings, ensuring the rigor and credibility of the analytical results.

Empirical findings demonstrated that the proposed framework improved anomaly detection performance and enhanced sensitivity analysis for the modeled SDN-IoT network.

5.2 Contribution to Knowledge

This research advances the state of knowledge in SDN-IoT anomaly detection and performance analysis by introducing a unified architectural framework, DOE-GAN-SA, that integrates data generation, anomaly characterisation, and sensitivity analysis within a single methodology. Existing studies typically examine these components in isolation, focusing either on anomaly detection, experimental evaluation, or statistical modelling without establishing a cohesive analytical workflow. The proposed framework addresses this gap by combining design of experiments (DOE), generative adversarial networks (GANs), anomaly detection and characterisation (ADC), and Artificial Neural Network (ANN)-based sensitivity analysis into a structured, end-to-end architecture. A key contribution lies in the integration of Latin Hypercube Sampling with GAN-based data augmentation to address limitations found in prior works, which often rely on small, imbalanced datasets generated from constrained SDN testbeds. The framework ensures statistical robustness by validating the similarity of synthetic datasets to real network traffic, resulting in richer and more diverse training data for anomaly detection models. This improves generalisation and enhances the reliability of performance evaluations under both normal and attack conditions.

Lead City University DO NOT COPY

Another significant contribution is the architectural coupling of CART-based anomaly characterisation with ANN-driven sensitivity analysis. Whereas existing SDN anomaly detection studies primarily report classification accuracy without explaining which system parameters drive degradation, this framework identifies the relative influence of throughput, jitter, and response time on service performance. The sensitivity analysis provides deeper interpretability and offers practitioners actionable insights for optimising network configurations and resource allocation. The introduction of a weighted cost function further advances analytical capabilities by modelling the varying severity of different DDoS attack vectors. Existing approaches often treat all anomalies uniformly, limiting their ability to reflect real-world operational priorities. This framework captures differential attack impact, supporting more realistic and fine-grained resilience assessment.

Finally, the framework introduces methodological rigour through repeated experimental trials and statistical validation using hypothesis testing. The strong performance achieved, including 99.93 % accuracy, an F1 score of 1.000, and low mean squared error during sensitivity analysis, is supported by statistically robust evidence rather than single-run observations. Overall, DOE-GAN-SA provides a comprehensive, validated, and operationally practical approach for anomaly detection and performance optimisation in SDN-IoT environments, establishing a solid foundation for future research and real-time deployment

Lead City University DO NOT COPY

Feature/Method	DOE-GAN-SA	How It Improves on Previous Work
Framework Innovation	Introduced a hybrid sensitivity analysis framework (DOE-GAN-SA) combining Latin Hypercube Sampling (LHS), Generative Adversarial Networks (GAN), and Artificial Neural Networks (ANN) for anomaly detection + sensitivity analysis in SDN-IoT networks.	Earlier works such as treated sensitivity analysis (SA) and anomaly detection (ADC) separately, using regression or variance-based models without synthetic augmentation or unified hybridization.
Synthetic Data Generation	Used GANs to generate realistic synthetic traffic that preserves statistical similarity to real SDN data (validated via descriptive stats and hypothesis tests).	Prior models lacked robust synthetic data; data scarcity limited analysis. GANs improved dataset diversity and balance, enhancing model generalization.
DOE (LHS) Sampling Efficiency	LHS provided uniform parameter coverage, ensuring representative sampling of throughput (T_p), jitter (J_t), and response time (R_t) with fewer simulations.	Earlier approaches used Monte Carlo or variance-based sampling, requiring more runs and higher computational cost. DOE-GAN-SA achieves high efficiency and

Lead City University DO NOT COPY

		scalability with reduced sampling overhead
Anomaly Detection Accuracy (ADC)	CART classifier achieved 99.9% predictive accuracy and F1 = 1.000 when trained on the augmented dataset	Earlier ML-based SDN anomaly detection reached ~98–99% accuracy without synthetic augmentation. DOE-GAN-SA yields statistically higher accuracy ($p < 10^{-18}$).
Sensitivity Analysis	ANN-driven SA identified Response Time (Rt) as the most sensitive parameter (mean MSE ≈ 6.99), followed by Jitter (J_t) and Throughput (T_p). All differences were statistically significant ($p \approx 7 \times 10^{-18}$).	Prior SA using only regression or Monte Carlo methods lacked statistical testing and cross-validated ANN models; DOE-GAN-SA added hypothesis validation and error-based ranking of parameters.
Statistical Validation	Wilcoxon rank-sum tests confirmed all improvements were significant at $p < 0.05$.	Earlier works reported descriptive statistics only, without formal hypothesis testing.
Scalability and Robustness	Achieved low computational cost (< 250 epochs per ANN run) and high reproducibility (50 independent runs). Framework validated on normal and DDoS (TCP, UDP, HTTP) scenarios.	Previous models (e.g., regression-based) analyzed fewer attack types and lacked replication; DOE-GAN-SA scales efficiently across multiple IoT-SDN scenarios.

Lead City University DO NOT COPY

Complement to Prior Frameworks	Demonstrated a unified ADC + SA system for SDN-IoT, paving way for automated, data-driven network security analysis.	Earlier models were modular or isolated, addressing either anomaly detection or SA, not both simultaneously.

5.3 Area of Further Research

Future research will extend the framework to evaluate its applicability in the context of distributed denial-of-service (DDoS) flooding attacks and other emerging network anomalies. Comparative studies against alternative methods will also be pursued to further validate the robustness and generalizability of DOE-GAN-SA for SDN-enabled IoT networks.

Lead City University DO NOT COPY

Bibliography

Books

Andres, A. S. M. R. T., Gatelli, F. C. J. C. D., & Tarantola, M. S. S., Global sensitivity analysis: the primer. *Ist ed., John Wiley & Sons, The Atrium, Southern Gate, Chichester, England*, 2008.

Burkov, A. *The Hundred-Page Machine Learning Book.* Québec: Burkov Publishing, 2019.

Cacuci, D. G., & Ionescu-Bujor, M. *Sensitivity & Uncertainty Analysis, Data Assimilation, & Predictive Model Calibration.* In Handbook of Nuclear Engineering, Springer, 2010: 1913–2051.

Coker, O., & Azodolmolky, S. *Software-Defined Networking with OpenFlow: Deliver Innovative Business Solutions.* Birmingham: Packt Publishing Limited, 2017.

Forbes, C., Evans, M., Hastings, N., & Peacock, B. *Statistical Distributions.* Wiley, 2011.

Goodfellow, I., Bengio, Y., & Courville, A. *Deep Learning.* MIT Press, 2016.

Mohammed, M., Khan, M. B., & Bashier, E. B. M. *Machine Learning: Algorithms & Applications.* CRC Press, 2016.

Rasmussen, C. E., & Williams, C. K. I. *Gaussian Processes for Machine Learning.* MIT Press**,** 2006.

Santner, T. J., Williams, B., & Notz, W. *The Design & Analysis of Computer Experiments.* Springer**,** 2018.

Conferences

Lead City University DO NOT COPY

Bhushan, K., Gupta, B., & Misra, S. *Software Defined Networking: Review, Challenges & Research Directions.* International Conference on Communication Systems & Networks (COMSNETS)**,** 2015, 1–10.

Carter, C., & Vapnik, V. *Estimation of Dependences Based on Empirical Data.* Conference on Machine Learning, 1999.

Eason, G., Kloti, R., Kotronis, V., & Smith, P. *OpenFlow: A Security Analysis.* IEEE International Conference on Network Protocols (ICNP), 2013.

Fox, J., Ökten, G. & Uzunoğlu, B. *Global Sensitivity Analysis for Power Systems via Quasi Monte Carlo Methods.* International Conference on System Reliability & Safety**,** 2019, 446–451.

Hagan, M. T., & Menhaj, M. *Training Feedforward Networks with the Marquardt Algorithm.* IEEE International Conference on Neural Networks (ICNN)**,** 1994, 1–6.

Karki, D., & Dawadi, B. R. *Machine Learning-based DDoS Detection System in SDN.* IOE Graduate Conference, 2022, 228–236.

Kowalski, P. A., & Kusy, M. *Global Sensitivity Analysis for Probabilistic Neural Network Reduction.* International Joint Conference on Neural Networks (IJCNN), 2016.

Levine, A., Cukier, R., & Shuler, K. *Sensitivity Analysis for Multi-Parameter Reactor Models.* ANS Conference on Reactor Modelling, 1978.

Namal, S., Ahmad, I. A., Ylianttila, M., & Gurtov, A. *Enabling Secure Mobility with OpenFlow.* IEEE SDN for Future Networks & Services Conference, 2013.

Porras, P., Shin, S., Yegneswaran, V., & Gu, G. *Avant-Guard: Scalable & Vigilant Switch Flow Management in Software Defined Networks.* ACM Conference on Computer & Communications Security (CCS)**,** 2013, 413–424.

Rojas, C., et al. *Efficient Parameter Estimation for Electrochemical Energy Models.* IEEE Energy Storage Systems Conference, 2023, 1–10.

Shin, S. Y., Nejati, S., Sabetzadeh, M., Briand, L. C., Arora, C., & Zimmer, F. *Dynamic Adaptation of SDNs for IoT Systems: A Search-Based Approach.* International Conference on Software Engineering (ICSE), SEAMS Workshop, 2020, 137–148.

Tang, M. C., Chen, Y. T., Cheng, Y. C., & Chang, Y. R. *Future On-Demand Service & Security Design.* IEEE Communications Technology Conference, 2010, 385–388.

Wang, L., Hu, T., Xie, J., Tang, G., & Lu, J. *A Safe-Guard Mechanism for SDN Controllers Under DDoS.* IEEE International Conference on Communications (ICC)**,** 2019, 1–6.

Journals

Lead City University DO NOT COPY

Ahmad, S. & Mir, A. H., *Scalability, Consistency, Reliability & Security in SDN Controllers: A Survey of Diverse SDN Controllers.* **Journal of Network & Systems Management,** 29(1), 2021, 1-59. Available online: https://doi.org/10.1007/s10922-020-09569-5.

Akinsolu, M. O. & Zribi, K., *A Generalised Framework for Adopting Regression-Based Predictive Modelling in Manufacturing.* **Inventions,** 8(1), 2023, 1-32. Available online: https://doi.org/10.3390/inventions8020032.

Akinsolu, M. O., Liu, B., Grout, V., Lazaridis, P., Mognaschi, M. E. & Di Barba, P., *A Parallel Surrogate Model Assisted Evolutionary Algorithm for Electromagnetic Design Optimization.* **IEEE Transactions on Emerging Topics in Computational Intelligence,** 3(2), 2019, 93–105. Available online: https://doi.org/10.1109/TETCI.2018.2877259.

Akinsolu, M. O., Sangodoyin, A. O. & Uyoata, U., *Behavioral Study of Software-Defined Network Parameters Using Exploratory Data Analysis & Regression-Based Sensitivity Analysis.* **Mathematics,** 10(14), 2022, 1-24. Available online: https://doi.org/10.3390/math10142536.

Aladaileh, M. A., Anbar, M., Hasbullah, I. H., Chong, Y. W. & Sanjalawe, Y. K., *Detection Techniques of Distributed Denial of Service Attacks on Software Defined Networking Controller: A Review.* **IEEE Access,** 8, 2020, 143985–143995. Available online: https://doi.org/10.1109/ACCESS.2020.3015331.

Alhijawi, B., Mismar, T., Alsmirat, M. A. & Al-Ayyoub, M., *A Comprehensive Survey on DDoS Attacks & Defence Mechanisms.* **Computer Science Review,** 37, 2020, 100279. Available online: https://doi.org/10.1016/j.cosrev.2020.100279.

Al-Somaidai, M. B., & Yahya, E. B. *Survey of Software Components to Emulate OpenFlow Protocol as an SDN Implementation.* **American Journal of Software Engineering & Applications** 3(6), 2014, 74–82.

Anerousis, N., Chemouil, P., Lazar, A. A., Mihai, N. & Weinstein, S. B., *The Origin & Evolution of Open Programmable Networks & SDN.* **IEEE Communications Surveys & Tutorials,** 23(3), 2021, 1956–1971. Available online: https://doi.org/10.1109/COMST.2021.3071858.

Ayanoglu, E., Davaslioglu, K. & Sagduyu, Y. E., *Machine Learning in NextG Networks via Generative Adversarial Networks.* **IEEE Transactions on Cognitive Communications & Networking,** 8(2), 2022, 480–502. Available online: https://doi.org/10.1109/TCCN.2022.3150197.

Ayanoglu, E., Davaslioglu, K., & Sagduyu, Y. E. *Machine Learning in NextG Networks via Generative Adversarial Networks.* **IEEE Transactions on Cognitive Communications & Networking** 8(2), 2022, 480–502.

Leadcity University DO NOT COPY

Badotra, S. & Chandramouli, B., *A DDoS Vulnerability Analysis System Against Distributed Denial of Service Attacks in Software Defined Networking.* **Electronics,** 11(19), 2022, 1-15. Available online: https://doi.org/10.3390/electronics11193120.

Bhuiyan, Z. A., Islam, S., Islam, M. M., Ullah, A. A., Naz, F. & Rahman, M. S., *On the Insecurity of the Control Plane of SDN Architecture: A Survey.* **IEEE Access,** 11, 2023, 91550–91582.

Biazar, D., Khaloozadeh, H., & Siahi, M. *Sensitivity Analysis for Evaluation of Sensor Error on Wind Turbine Variables Using Monte Carlo Simulation.* **IET Renewable Power Generation** 16(1), 2022, 1623–1635.

Bizanis, N., & Kuipers, F. A. *SDN & Virtualization Solutions for the Internet of Things: A Survey.* **IEEE Access** 4(1), 2016, 5591–5606.

Breiman, L., *Random Forests.* **Machine Learning** 45(1), 2001, 5–32. https://doi.org/10.1023/A:1010933404324.

Brett, G., & O'Callaghan, N. *Automating Labour & the Spatial Politics of Data Centre Technologies.* In **Topologies of Digital Work**, 2022(1), 2022, 77–101.

Brown, I., & Mues, C. *Comparison of Classification Algorithms for Imbalanced Credit Scoring.* **Expert Systems with Applications** 39(1), 2012, 3446–3453.

Burhan, M., Rehman, R., Khan, B. & Kim, B. S., *IoT Elements, Layered Architectures & Security Issues: A Comprehensive Survey.* **Sensors,** 18(9), 2018, 1-37. Available online: https://doi.org/10.3390/s18092796.

Cacuci, D. G. & Ionescu Bujor, M., *A Comparative Review of Sensitivity & Uncertainty Analysis of Large Scale Systems. I: Deterministic Methods.* **Nuclear Science & Engineering,** 147(3), 2004, 189–203.

Cacuci, D. G. & Ionescu Bujor, M., *Sensitivity & Uncertainty Analysis, Data Assimilation, & Predictive Best Estimate Model Calibration.* **Handbook of Nuclear Engineering,** 2010, 1913–2051.

Carter, C. & Vapnik, V., *Support Vector Networks.* **Machine Learning,** 20(3), 1995, 273–297. Available online: https://doi.org/10.1007/BF00994018.

Casado, M., et al. *OpenFlow: Enabling Innovation in Campus Networks.* **ACM SIGCOMM Computer Communication Review** 38(2), 2008, 69–74.

Chen, M., Challita, U., Saad, W., Yin, C., & Debbah, M. *Artificial Neural Network Based Machine Learning for Wireless Networks.* **IEEE Communications Surveys & Tutorials** 21(4), 2019, 3039–3071.

Lead City University DO NOT COPY

Cortes, C., & Vapnik, V. *Support-Vector Networks.* **Machine Learning** 20(3), 1995, 273–297.

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B. & Bharath, A., *Generative Adversarial Networks: An Overview.* **IEEE Signal Processing Magazine,** 35(1), 2018, 53–65. Available online: https://doi.org/10.1109/MSP.2017.2765202.

Cukier, R., Levine, H. & Shuler, K., *Nonlinear Sensitivity Analysis of Multiparameter Model Systems.* **Journal of Computational Physics,** 26(1), 1978, 1–42.

Cukier, R., Levine, H., & Shuler, K. *Nonlinear Sensitivity Analysis of Multiparameter Model Systems.* **Journal of Computational Physics** 26(1), 1978, 1–42.

Dagiuklas, T., Ghosh, S., Iqbal, M. & Wang, X., *A Cognitive Routing Framework for Reliable Communication in IoT for Industry 5.0.* **IEEE Transactions on Industrial Informatics,** 18(7), 2022, 5446–5457. Available online: https://doi.org/10.1109/TII.2022.3152646.

Darwish, T., Alhaj, T. A. & Elhaj, F. A., *Controller Placement in Software Defined Emerging Networks: A Review & Future Directions.* **Telecommunication Systems,** 88(1), 2025, 18. Available online: https://doi.org/10.1007/s11235-024-01048-2.

Dayal, N., Maity, P., Srivastava, S., & Khondoker, R. *Research Trends in Security & DDoS in SDN.* **Security & Communication Networks** 9(18), 2016, 6386–6411.

Dunmore, A., Jang-Jaccard, J., Sabrina, F., & Kwak, J. *A Comprehensive Survey of GANs in Cybersecurity Intrusion Detection.* **IEEE Access** 11(1), 2023, 76071–76094.

ElSayed, M. S., Le Khac, N. A., Albahar, M. A. & Jurcut, A., *A Novel Hybrid Model for Intrusion Detection Systems in SDNs.* **Journal of Network & Computer Applications,** 194, 2021, 1-18. Available online: https://doi.org/10.1016/j.jnca.2021.103160.

Escudero-Sahuquillo, J., Garcia, P. J., Quiles, F. J., Reinemo, S. A., Skeie, T., Lysne, O., & Duato, J. *Congestion Control in InfiniBand-Based Fat Trees.* **Journal of Parallel & Distributed Computing,** 74(1), 2014, 1802–1819.

Ezechi, C., Akinsolu, M. O., Sangodoyin, A. O., Akinsolu, F. T. & Sakpere, W., *Software-Defined Networking in Cyber-Physical Systems.* **Cyber Physical System 2.0,** 2024(1), 2024, 44-69.

Feamster, N., Rexford, J., & Zegura, E. *The Road to SDN: An Intellectual History of Programmable Networks.* **ACM SIGCOMM Computer Communication Review,** 44(2), 2014, 87–98.

Leo City University DO NOT COPY

Figueira, Á. & Vaz, B., *Survey on Synthetic Data Generation, Evaluation Methods & GANs.* **Mathematics,** 10(19), 2022, 1-41. Available online: https://doi.org/10.3390/math10152733.

Figueira, Á., & Vaz, B. *Survey on Synthetic Data Generation, Evaluation Methods & GANs.* **Mathematics,** 10(15), 2022, 1-10.

Forbes, C., Evans, M., Hastings, N. & Peacock, B., *Statistical Distributions.* **John Wiley & Sons,** 2011.

Gao, S., Peng, Z., Xiao, B., Hu, A., Song, Y. & Ren, K., *Detection & Mitigation of DoS Attacks in Software-Defined Networks.* **IEEE/ACM Transactions on Networking,** 28(3), 2020, 1419–1433. Available online: https://doi.org/10.1109/TNET.2020.2980981.

Gao, S., Peng, Z., Xiao, B., Hu, A., Song, Y., & Ren, K. *Detection & Mitigation of DoS Attacks in Software-Defined Networks.* **IEEE/ACM Transactions on Networking,** 28(3), 2020, 1419–1433.

Gilani, S. M. M., Usman, M., Daud, S., Kabir, A., Nawaz, Q. & Judit, O., *SDN-Based Multi-Level Framework for Smart Home Services.* **Multimedia Tools & Applications,** 83(1), 2024, 327–347. Available online: https://doi.org/10.1007/s11042-023-15156-1.

Gilani, S. M. M., Usman, M., Daud, S., Kabir, A., Nawaz, Q., & Judit, O. *SDN-Based Multi-Level Framework for Smart Home Services.* **Multimedia Tools & Applications,** 83(1), 2024, 327–347.

Girish, L., Akinsolu, M. O. & Sangodoyin, A. O., *DadGAN: DDoS Anomaly Detection Using Generative Adversarial Networks.* **IEEE Symposium on Security & Privacy Workshops,** 2021(1), 2021, 1–7.

Goyal, M. & Mahmoud, Q. H., *A Systematic Review of Synthetic Data Generation Techniques Using Generative AI.* **Electronics,** 13(17), 2024, 1-38. Available online: https://doi.org/10.3390/electronics13173509.

Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., & Shenker, S., *NOX: Towards an Operating System for Networks*, **ACM SIGCOMM Computer Communication Review**, 38(3), 2008, 105-110.

Helton, J. C., & Davis, F. J. *Latin Hypercube Sampling & the Propagation of Uncertainty.* **Reliability Engineering & System Safety,** 81(1), 2003, 23–69.

Helton, J. C., *Uncertainty & Sensitivity Analysis Techniques for Use in Performance Assessment for Radioactive Waste Disposal.* **Reliability Engineering & System Safety,** 42(2–3), 1993, 327–367.

Lead City University DO NOT COPY

Hodo, E., Bellekens, X., Hamilton, A., Dubouilh, P. L., Iorkyase, E., Tachtatzis, C. & Atkinson, R., *Threat Analysis of IoT Networks Using ANN Intrusion Detection.* **ISNCC,** 2016(1), 2016, 1-6

Huang, W., Zhou, L. & Li, Z., *DDoS Attack Detection Based on Machine Learning & Deep Learning Techniques: A Comprehensive Review.* **Journal of Network & Computer Applications,** 238, 2024, 1256. Available online: https://doi.org/10.1016/S2772-6711(24)00125-6.

Hung, N. H., Koh, S. P., Yaw, C. T., Tiong, S. K., Benedict, F., Yusaf, T. & Hong, T. C., *SDN-Based VANET Routing: A Comprehensive Survey on Architectures, Protocols, Analysis & Future Challenges.* **IEEE Access,** 13(1), 2024, 126801–126861. Available online: https://doi.org/10.1109/ACCESS.2024.3418600.

Hussain, M., Shah, N., Amin, R., Alshamrani, S. S., Alotaibi, A., & Raza, S. M. *Software-Defined Networking: Categories, Analysis & Future Directions.* **Sensors,** 22(15), 2022, 1-16.

Ionescu Bujor, M. & Cacuci, D. G., *Sensitivity & Uncertainty Analysis of Large Scale Systems.* **Nuclear Science & Engineering,** 147, 2004, 189–217.

Jain, A. K., Shukla, H. & Goel, D. A., *A Comprehensive Survey on DDoS Detection, Mitigation & Defense Strategies in Software Defined Networks.* **Cluster Computing,** 27(9), 2024, 13129–13164. Available online: https://doi.org/10.1007/s10586-024-04596-z.

Kirch, J., et al., *The Effect of Model Rescaling & Normalization on Sensitivity Analysis on an Example of a MAPK Pathway Model.* **EPJ Nonlinear Biomedical Physics,** 4(1), 2016, 1–19. Available online: https://doi.org/10.1140/epjnbp/s40366-016-0030-z.

Kowalski, P. A. & Kusy, M., *Sensitivity Analysis for Probabilistic Neural Network Structure Reduction.* **IEEE Transactions on Neural Networks & Learning Systems,** 29(5), 2017, 1919–1932. Available online: https://doi.org/10.1109/TNNLS.2017.2681760.

Kreutz, D., Ramos, F. M. V., Verissimo, P., Rothenberg, C. E., Azodolmolky, S. & Uhlig, S., *Software Defined Networking: A Comprehensive Survey.* **Proceedings of the IEEE,** 103(1), 2015, 14–76. Available online: https://doi.org/10.1109/JPROC.2014.2371999.

Lai, M. C., Lu, P. J. & Chang, J. S., *A Survey of High Performance Interconnection Networks in High Performance Computer Systems.* **Electronics,** 11(9), 2022, 1-23. Available online: https://doi.org/10.3390/electronics11091369.

Lecun, Y., Bengio, Y. & Hinton, G., *Deep Learning.* **Nature,** 521, 2015, 436–444.

LeCun, Y., Bengio, Y., & Hinton, G. *Deep Learning.* **Nature,** 521(7553), 2015, 436–444.

Leadcity University DO NOT COPY

Li, M., Wu, H., Wang, Y., Handroos, H. & Carbone, G., *Modified Levenberg–Marquardt Algorithm Backpropagaation Neural Network Training in Dynamic Model Identification of Mechanical Systems.* **Journal of Dynamic Systems, Measurement, & Control,** 139(3), 2017, 031012. Available online: https://doi.org/10.1115/1.4035010

Lim, W., Chek, K., Lau, B. & Lin, C., *Future of GAN for Anomaly Detection in Network Security.* **Computers & Security,** 136, 2024, 103733. Available online: https://doi.org/10.1016/j.cose.2023.103733.

Link, K. G., Stobb, M. T., Di Paola, J., Neeves, K. B., Fogelson, A. L., Sindi, S. S., & Leiderman, K. *A Local and Global Sensitivity Analysis of a Mathematical Model of Coagulation and Platelet Deposition Under Flow*, **PLoS One**, 13(7), 2018, 1-10.

Liu, J., Li, Y., Chen, M., Dong, W., & Jin, D, *Software-Defined Internet of Things for Smart Urban Sensing*, **IEEE Communications Magazine**, 53(9), 2015, 55-63.

Liu, Y., Poor, H. V., Xiao, M., Pang, Z. & Vucetic, B., *High Reliability & Low Latency Wireless Communication for IoT: Challenges & Fundamentals.* **IEEE Internet of Things Journal,** 6(5), 2019, 7946–7970. Available online: https://doi.org/10.1109/JIOT.2019.2907245.

Liu, Y., Tong, N., Wu, X., Han, X. & Chen, C., *A Generalised Sensitivity Analysis Method Based on Variance & Covariance Decomposition for Multi Input Multi Output Systems.* **Computer Methods in Applied Mechanics & Engineering,** 385(1), 2021, 114009. Available online: https://doi.org/10.1016/j.cma.2021.114009.

Lombardi, M., Pascale, F. & Santaniello, D., *Internet of Things: A General Overview Between Architectures, Protocols & Applications.* **Information,** 12(2), 2021, 1-20. Available online: https://doi.org/10.3390/info12020087.

Lu, P. J., Lai, M. C., & Chang, J. S. *High-Performance Interconnection Networks: A Survey.* **Electronics,** 11(9), 2022, 1-17.

Mahrarch, S. & Haqiq, A., *DDoS Flooding Attack Mitigation in Software Defined Networks.* **International Journal of Advanced Computer Science & Applications,** 11(1), 2020, 37–44. Available online: https://doi.org/10.14569/IJACSA.2020.0110106.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., & Turner, J. *OpenFlow: Enabling Innovation.* **ACM SIGCOMM Computer Communication Review** 38(2), 2008, 69–74.

Lead City University DO NOT COPY

Mishra, S. R., Shanmugam, B., Yeo, K. C. & Thennadil, S., *SDN Enabled IoT Security Frameworks: A Review of Existing Challenges.* **Technologies,** 13(3), 2025, 1-38. Available online: https://doi.org/10.3390/technologies13030121.

Musa, N. S., Mirza, N. M., Rafique, S. H., Abdallah, A. M. & Murugan, T., *Machine Learning & Deep Learning Techniques for Distributed Denial of Service Anomaly Detection in SDNs.* **IEEE Access,** 12(1), 2024, 17982–18011. Available online: https://doi.org/10.1109/ACCESS.2024.3360868.

Najafimehr, M., Zarifzadeh, S. & Mostafavi, S., *DDoS Attacks & Machine Learning Based Detection Methods: A Survey and Taxonomy.* **Engineering Reports,** 5(12), 2023, e12697. Available online: https://doi.org/10.1002/eng2.12697

Namal, S., Ahmad, I. A., Ylianttila, M. & Gurtov, A., *Security in Software Defined Networks: A Survey.* **IEEE Communications Surveys & Tutorials,** 17(4), 2015, 2317–2346. Available online: https://doi.org/10.1109/COMST.2015.2449531.

Nisar, K., Jimson, E. R., Hijazi, M. H. A., Welch, I., Hassan, R., Aman, A. H. M., Sodhro, A. H., Pirbhulal, S. & Khan, S., *A Survey on the Architecture, Application & Security of Software Defined Networking: Challenges & Open Issues.* **Internet of Things,** 12(100289), 2020, 1-15. Available online: https://doi.org/10.1016/j.iot.2020.100289.

Nossent, J., Zadeh, F. K., Sarrazin, F., Pianosi, F., Van Griensven, A., Wagener, T. & Bauwens, W., *Comparison of Variance Based & Moment Independent Global Sensitivity Analysis Approaches by Application to the SWAT Model.* **Environmental Modelling & Software,** 91(1), 2017, 210–222. Available online: https://doi.org/10.1016/j.envsoft.2017.02.001.

Palmer, K., & Tsui, K. *A Minimum-Bias Latin Hypercube Design.* **IEEE Transactions on Reliability,** 33(9), 2001, 793–808.

Pandey, C., Tiwari, V., Rodrigues, J. J., & Roy, D. S, 5GT-GAN-NET: *Internet Traffic Data Forecasting with Supervised Loss Based Synthetic Data Over 5G*, **IEEE Transactions on Mobile Computing**, 23(11), 2024, 10694-10705.

Pang, Z., O'Neill, Z., Li, Y. & Niu, F., *The Role of Sensitivity Analysis in the Building Performance Analysis: A Critical Review*. **Energy and Buildings**, 209(109659), 2020, 1-15.

Piano, S. L., Ferretti, F., Puy, A., Albrecht, D. & Saltelli, A., *Variance Based Sensitivity Analysis: The Quest for Better Estimators & Designs Between Explorativity & Economy.* **Reliability Engineering & System Safety,** 206(107300), 2021, 1-31. Available online: https://doi.org/10.1016/j.ress.2020.107300.

Leeds City University DO NOT COPY

Qin, C., Jin, Y., Tian, M., Ju, P., & Zhou, S., *Comparative Study of Global Sensitivity Analysis and Local Sensitivity Analysis in Power System Parameter Identification*, **Energies**, 16(16), 2023, 1-16.

Qureshi, K. N., et al., *A Distributed Software Defined Networking Model to Improve Scalability & Quality of Services for Green Energy Smart Grid Systems.* **Computers & Electrical Engineering,** 84(106634), 2020, 1-17. Available online: https://doi.org/10.1016/j.compeleceng.2020.106634.

Razavi, S. & Gupta, H. V., *What Do We Mean by Sensitivity Analysis?* **Water Resources Research,** 51(5), 2015, 3070–3092. Available online: https://doi.org/10.1002/2014WR016527.

Razavi, S., & Gupta, H. V. *What Do We Mean by Sensitivity Analysis?,* **Water Resources Research,** 51(5), 2015, 3070–3092.

Ring, M., Schlör, D., Landes, D. & Hotho, A., *Flow-Based Network Traffic Generation Using GANs.* **Computers & Security,** 82(1), 2019, 156–172. Available online: https://doi.org/10.1016/j.cose.2018.11.007.

Rojas, C., Oca, L., Lopetegi, I., Iraola, U., & Carrasco, J., *A Critical Look at Efficient Parameter Estimation Methodologies of Electrochemical Models for Lithium-Ion Cells*, **Journal of Energy Storage**, 80(110384), 2024, 1-16.

Rojas, C., Oca, L., Lopetegi, I., Iraola, U., & Carrasco, J., *A Critical Look at Efficient Parameter Estimation Methodologies of Electrochemical Models for Lithium-Ion Cells*, **Journal of Energy Storage**, 80(110384), 2024, 1-16.

Sahi, A., Lai, D., Li, Y., & Diykh, M. *Efficient DDoS TCP Flood Attack Detection in Cloud Environments.* **IEEE Access,** 5(1), 2017, 6036–6048.

Saltelli, A., Annoni, P., Azzini, I., Campolongo, F., Ratto, M. & Tarantola, S., *Why So Many Published Sensitivity Analyses Are False.* **Environmental Modelling & Software,** 114(1), 2019, 29–39. Available online: https://doi.org/10.1016/j.envsoft.2019.01.012.

Saltelli, A., *Sensitivity Analysis for Importance Assessment.* **Risk Analysis,** 22(3), 2002, 579–590. Available online: https://doi.org/10.1111/0272-4332.00040.

Sangodoyin, A. O., Akinsolu, M. O. & Awan, I., *A Deductive Approach for the Sensitivity Analysis of Software Defined Network Parameters.* **Simulation Modelling Practice & Theory,** 103(102099), 2020, 1-29. Available online: https://doi.org/10.1016/j.simpat.2019.102099.

Leeds City University DO NOT COPY

Sangodoyin, A. O., Akinsolu, M. O., Pillai, P. & Grout, V., *Detection & Classification of DDoS Flooding Attacks on Software Defined Networks.* **IEEE Access,** 9(1), 2021, 122495–122508. Available online: https://doi.org/10.1109/ACCESS.2021.3109490.

Sarkar, I. H., *Machine Learning: Algorithms, Real World Applications & Research Directions.* **SN Computer Science,** 2(3), 2021, 1-21. Available online: https://doi.org/10.1007/s42979-021-00592-x.

Sarker, I. H., Furhad, M. & Nowrozy, R., *AI Driven Cybersecurity: An Overview.* **SN Computer Science,** 2(3), 2021, 173-187. Available online: https://doi.org/10.1007/s42979-021-00626-2.

Selvi, M., Mayur, K. L. & SVN, S. K., *A Comprehensive Review on Authentication, Threats & Privacy Preserving Challenges for Securing Smart Transportation Infrastructure.* **Peer-to-Peer Networking & Applications,** 18(6), 2025, 1–32. Available online: https://doi.org/10.1007/s12083-025-01868-1.

Shaji, N. S., Jain, T., Muthalagu, R. & Pawar, P. M. *Deep Discovery: Anomaly Discovery in SDNs Using ANN.* **Computers & Security,** 132(103320), 2023, 1-19. Available online: https://doi.org/10.1016/j.cose.2021.103320.

Shields, M. D. & Zhang, J., *The Generalisation of Latin Hypercube Sampling.* **Reliability Engineering & System Safety,** 148(1), 2016, 96–108. Available online: https://doi.org/10.1016/j.ress.2015.12.002.

Shukla, R. S. & Alatawi, A. M., *Enhancing Internet Traffic Forecasting in MEC Environments with 5GT-Trans: Leveraging Synthetic Data & Transformer Based Models.* **IEEE Access,** 13(1), 2025, 83607–83618. Available online: https://doi.org/10.1109/ACCESS.2025.3569389.

Singh, C. & Jain, A. K., *A Comprehensive Survey on DDoS Attacks Detection & Mitigation in SDN IoT Networks.* **e-Prime: Advances in Electrical Engineering, Electronics & Energy,** 8(100543), 2024, 1-16. Available online: https://doi.org/10.1016/j.prime.2024.100168.

Singh, J. & Behal, S., *Detection & Mitigation of DDoS Attacks in SDN: A Comprehensive Review, Research Challenges & Future Directions.* **Computer Science Review,** 37(100279), 2020, 1-15. Available online: https://doi.org/10.1016/j.cosrev.2020.100279.

Sobol, I. M. *Global Sensitivity Analysis for Nonlinear Mathematical Models.* **Mathematics & Computers in Simulation,** 55(1-3), 2001, 271–280.

Sobol, I. M., *Global Sensitivity Analysis for Nonlinear Mathematical Models & Their Monte Carlo Estimates.* **Mathematics & Computers in Simulation,** 55, 2001, 271–280. Available online: https://doi.org/10.1016/S0378-4754(00)00270-0.

Leeds City University DO NOT COPY

Sobol, I. M., *Sensitivity Indices for Nonlinear Mathematical Models & Their Monte Carlo Estimates.* **Mathematical Modelling & Computational Experiment,** 1(4), 1993, 407–414.

Stein, M., *Large Sample Properties of Simulations Using Latin Hypercube Sampling.* **Technometrics,** 29(2), 1987, 143–151. Available online: https://doi.org/10.1080/00401706.1987.10488205.

Thanh, N. H., Tuan, N. N., Khoa, D. A., Tuan, L. C., Kien, N. T., Dung, N. X. & Wamser, F., *Profiling, Benchmarking & Behavioural Analysis of SDN Architecture Under DDoS Attacks.* **Journal of Network & Systems Management,** 31(2), 2023, 43-56. Available online: https://doi.org/10.1007/s10922-022-09748-3.

Trucano, T. G., et al., *Calibration, Validation & Sensitivity Analysis: What's What.* **Reliability Engineering & System Safety,** 91(10–11), 2006, 1331–1357. Available online: https://doi.org/10.1016/j.ress.2005.11.031.

Wang, J., Wang, L. & Wang, R., *A Method of DDoS Attack Detection & Mitigation for the Comprehensive Coordinated Protection of SDN Controllers.* **Entropy,** 25(8), 2023, 1-26. Available online: https://doi.org/10.3390/e25081210.

Wang, Y., Hu, T., Tang, G., Xie, J. & Lu, J., *SGS: Safe Guard Scheme Against DDoS Attacks in Software Defined Networking.* **IEEE Access,** 7(1), 2019, 34699–34710. Available online: https://doi.org/10.1109/ACCESS.2019.2895092.

Wazirali, R., Ahmad, R., & Alhiyari, S. *SDN–OpenFlow Topology Discovery: Performance Issues.* **Applied Sciences,** 11(15), 2021, 1-16.

Wilcoxon, F. *Individual Comparisons by Ranking Methods in Breakthroughs in Statistics*, **Methodology and Distribution,** 1992(1), 1992, 196–202.

You, X., et al., *Towards 6G Wireless Communication Networks: Vision, Enabling Technologies & New Paradigm Shifts.* **Science China Information Sciences,** 64(110301), 2021, 1-15. Available online: https://doi.org/10.1007/s11432-021-3330-1.

Zadeh, F. K., Nossent, J., Sarrazin, F., Pianosi, F., Van Griensven, A., Wagener, T. & Bauwens, W., *Comparison of Variance Based & Moment Independent Global Sensitivity Analysis Approaches by Application to the SWAT Model.* **Environmental Modelling & Software,** 91, 2017, 210–222. Available online: https://doi.org/10.1016/j.envsoft.2017.02.001.

Zhang, J., *Modern Monte Carlo Methods for Efficient Uncertainty Quantification & Propagation: A Survey.* **Wiley Interdisciplinary Reviews: Computational Statistics** 13(5), 2021, 1-44. Available online: https://doi.org/10.1002/wics.1539.

Leeds City University DO NOT COPY

Zheng, J., Li, Q., Gu, G., Cao, J., Yau, D. K. & Wu, J., *Realtime DDoS Defence Using COTS SDN Switches via Adaptive Correlation Analysis.* **IEEE Transactions on Information Forensics & Security,** 13(7), 2018, 1838–1853. Available online: https://doi.org/10.1109/TIFS.2018.2812141.

Zhu, Z., Wang, J. & Zhao, L., *Enhancing SDN Controller Performance Under High Traffic Loads Using Adaptive Flow Management.* **IEEE Access,** 10(1), 2022, 45012–45025.

Zurada, J. M., Malinowski, A. & Usui, S., *Perturbation Method for Deleting Redundant Inputs of Perceptron Networks.* **Neurocomputing,** 14(2), 1997, 177–193. Available online: https://doi.org/10.1016/S0925-2312(97)00056-5.

Zurada, J. M., Malinowski, A., & Usui, S. *Perturbation Method for Deleting Redundant Inputs in Perceptron Networks.* **Neurocomputing,** 14(2), 1997, 177–193.

Unpublished Thesis / Dissertation

Sangodoyin, A. O. *Design and Analysis of Anomaly Detection and Mitigation Schemes for Distributed Denial of Service Attacks in Software Defined Networks,* **PhD diss., University of Bradford,** 2019. https://bradscholars.brad.ac.uk/handle/10454/18777.

Workshop

Arulkumaran, K., Creswell, A., White, T., Dumoulin, V. & Bharath, A. *An Introduction to Generative Adversarial Networks,* NIPS Deep Learning Workshop, 2016.

Benton, K., Camp, L. J., & Small, C. *OpenFlow Vulnerability Assessment.* **ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking**, 2013: 151–152.

Girish, L., Akinsolu, M. O., & Sangodoyin, A. O. *DadGAN: DDoS Anomaly Detection Using Generative Adversarial Networks,* IEEE Symposium on Security & Privacy Workshops (SPW), 2021, 1–7.

Hodo, E., Bellekens, X., Hamilton, A., Dubouilh, P. L., Iorkyase, E., Tachtatzis, C., & Atkinson, R. *Threat Analysis of IoT Networks Using Artificial Neural Networks,* International Symposium on Networks, Computers & Communications (ISNCC), 2016, 1–6.

Jafarian, J. H., Al-Shaer, E., & Duan, Q. *OpenFlow Random Host Mutation: Transparent Moving Target Defence.* ACM HotSDN Workshop, 2012.

Kim, J., Voellmy, A., & Feamster, N. *Improving Network Security with Software Defined Networking,* ACM Hot Topics in Software Defined Networking (HotSDN), 2012, 17–22.

Leeds City University DO NOT COPY

Kirch, J., et al. *The Effect of Model Rescaling & Normalization on Sensitivity Analysis on a MAPK Pathway Model,* EPJ Nonlinear Biomedical Physics Workshop**,** 2016.

Kreutz, D., Ramos, F. M. V., Verissimo, P., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. *Software Defined Networking: A Comprehensive Survey,* IEEE SDN for Future Networks Workshop, 2014.

Kreutz, D., Ramos, F., & Veríssimo, P. *Towards Secure & Dependable Software-Defined Networks.* ACM HotSDN Workshop, 2013, 55–60.

Marquardt, D. *An Algorithm for Least-Squares Estimation of Nonlinear Parameters,* SIAM Symposium on Estimation**,** 1963.

Porras, P., Shin, S., Yegneswaran, V., Fong, M., Tyson, M., & Gu, G. *A Security Enforcement Kernel for OpenFlow Networks,* ACM HotSDN Workshop, 2012, 121–126.

Saltelli, A., Tarantola, S., & Campolongo, F. *Sensitivity Analysis in the Context of Model Validation,* European Symposium on Sensitivity Analysis**,** 2004.

Shen, Z. Y., Su, M. W., Cai, Y. Z., & Tasi, M. H. *Mitigating SYN Flooding & UDP Flooding in P4-Based SDN,* Asia-Pacific Network Operations & Management Symposium (APNOMS), 2021, 374–377.

Shin, S., & Gu, G. *Attacking Software-Defined Networks: A Feasibility Study.* ACM HotSDN Workshop, 2013.

Sommer, R., & Paxson, V. *Outside the Closed World: On Machine Learning for Intrusion Detection.* IEEE Symposium on Security & Privacy, 2010, 305–316.

Stein, M. *Large Sample Properties of Latin Hypercube Sampling,* ASA Symposium on Statistical Design, 1987.

Voellmy, A., Kim, J., Feamster, S., & Feamster, N. *Procera: A Language for High-Level Reactive Network Control,* ACM SIGCOMM HotSDN Workshop, 2012, 1–6.

Xu, Y., & Liu, Y. *DDoS Attack Detection in SDN Using Entropy Variation & Flow Features,* IEEE INFOCOM Workshops, 2016, 1–9.

Xu, Y., & Liu, Y. *DDoS Attack Detection Under SDN Context,* IEEE INFOCOM Workshops, 2016, 1–9.

Zurada, J. M., Malinowski, A., & Cloete, I. *Sensitivity Analysis for Minimising Input Dimensions in Neural Networks,* IEEE International Symposium on Circuits & Systems (ISCAS), 1994, 447–450.

Lean City University DO NOT COPY

Reports and Internet Sources

DataReportal. *Internet Use in 2024.* DataReportal**,** 2024. https://datareportal.com/reports/digital-2024-deep-dive-the-state-of-internet-adoption.

Dennis, M. A. *Denial of Service Attack.* Encyclopedia Britannica, 2025. https://www.britannica.com/technology/denial-of-service-attack.

Dierks, T. *The Transport Layer Security (TLS) Protocol Version 1.2.* RFC 5246, 2008.

Esteban, C., Hyland, S. L., & Rätsch, G. *Real-Valued Medical Time-Series Generation with RCGANs.* arXiv preprint, 2017.

Goodfellow, I., et al. *NIPS Tutorial: Generative Adversarial Networks.* arXiv**,** 2016.

Hinrichs, T., Gude, N., Casado, M., Mitchell, J., & Shenker, S. *Expressing & Enforcing Flow-Based Network Security Policies.* University of Chicago Technical Report, 2008.

International Telecommunication Union. *Security Architecture for Systems Providing End-to-End Communications.* ITU-T X.805, 2003.

Ng, E., Cai, Z., & Cox, A. *Maestro: A System for Scalable OpenFlow Control.* Rice University Technical Report TR10-08, 2010.

Open Networking Foundation. *Homepage.* ONF**,** 2025. https://www.opennetworking.org.

Open Networking Foundation. *OpenFlow Switch Specification Version 1.5.1.* ONF, 2025. https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf.

Radford, A., Metz, L., & Chintala, S. *Unsupervised Representation Learning with DCGAN.* arXiv**,** 2015.

Scikit-learn. *Machine Learning in Python.* scikit-learn.org, 2025. https://scikit-learn.org/stable.

The MathWorks Inc, *Statistics and Machine Learning Toolbox.* **MathWorks**, 2025.

Lead City University DO NOT COPY

Appendix 1 (MATLAB Code for Training CART)

```
% function [trainedClassifier, validationAccuracy] = trainClassifier_CART(trainingData)
% [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)
% Returns a trained classifier and its accuracy. This code recreates the
% classification model trained in Classification Learner app. Use the
% generated code to automate training the same model with new data, or to
% learn how to programmatically train models.
%
%  Input:
%      trainingData: A table containing the same predictor and response
%       columns as those imported into the app.
%
%  Output:
%      trainedClassifier: A struct containing the trained classifier. The
%       struct contains various fields with information about the trained
%       classifier.
%
%      trainedClassifier.predictFcn: A function to make predictions on new
%       data.
```

Lead City University DO NOT COPY

```
%
%      validationAccuracy: A double containing the accuracy in percent. In
%       the app, the History list displays this overall accuracy score for
%       each model.
%
% Use the code to train the model with new data. To retrain your
% classifier, call the function from the command line with your original
% data or new data as the input argument trainingData.
%
% For example, to retrain a classifier trained with the original data set
% T, enter:
%   [trainedClassifier, validationAccuracy] = trainClassifier(T)
%
% To make predictions with the returned 'trainedClassifier' on new data T2,
% use
%   yfit = trainedClassifier.predictFcn(T2)
%
% T2 must be a table containing at least the same predictor columns as used
% during training. For details, enter:
%   trainedClassifier.HowToPredict


% Auto-generated by MATLAB on 03-Jan-2021 09:16:30
```

Lead City University DO NOT COPY

```
%% Use as a Script
clear
clc
close all force

for run_no=1:50

    rng shuffle % use current CPU time to shuffle random seed generator

    load ('zscoreTable.mat'); % Input Table z-scores

    trainingData=zscoreTable;

    % Extract predictors and response
    % This code processes the data into the right shape for training the
    % model.
    inputTable = trainingData;
    predictorNames = {'Tp', 'Jt', 'Rt'};
    predictors = inputTable(:, predictorNames);
    response = inputTable.Class;
    isCategoricalPredictor = [false, false, false, false];

    % Train a classifier
    % This code specifies all the classifier options and trains the classifier.
```

Lead City University DO NOT COPY

```
% Introduce variants ('MaxNumSplits' = 10, 20, 50, 100) 'MinLeafSize' 'MinParentSize'  'NumBins'

tic

classificationTree = fitctree(...

    predictors, ...

    response, ...

    'SplitCriterion', 'gdi', ...

    'MaxNumSplits', 100, ...

    'Surrogate', 'off', ...

    'ClassNames', {'HTTP'; 'Normal'; 'TCP'; 'UDP'});

trainingTime_CART(run_no,:)=toc;


% Create the result struct with predict function

predictorExtractionFcn = @(t) t(:, predictorNames);

treePredictFcn = @(x) predict(classificationTree, x);

trainedClassifier.predictFcn = @(x) treePredictFcn(predictorExtractionFcn(x));


% Add additional fields to the result struct

trainedClassifier.RequiredVariables = {'Jt', 'Rt', 'Tp'};

trainedClassifier.ClassificationTree = classificationTree;

 trainedClassifier.About = 'This struct is a trained model exported from Classification Learner R2020a.';

 trainedClassifier.HowToPredict = sprintf('To make predictions on a new table, T, use: \n  yfit = c.predictFcn(T) \nreplacing ''c'' with the name of the variable that is this struct, e.g. ''trainedModel''. \n \nThe table, T, must contain the variables returned by: \n  c.RequiredVariables \nVariable formats (e.g. matrix/vector, datatype) must match the original training data. \nAdditional variables are ignored. \n \nFor more information, see <a href="matlab:helpview(fullfile(docroot, ''stats'', ''stats.map''), ''appclassification_exportmodeltoworkspace'')">How to predict using an exported model</a>.');
```

Lead City University DO NOT COPY

```
% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
predictorNames = {'Tp', 'Jt', 'Rt'};
predictors = inputTable(:, predictorNames);
response = inputTable.Class;
isCategoricalPredictor = [false, false, false, false];

% Perform cross-validation
 partitionedModel = crossval(trainedClassifier.ClassificationTree, 'KFold', 5); %% ====> Use 5 and
10

% Compute validation predictions
tic
[validationPredictions, validationScores,validationCosts] = kfoldPredict(partitionedModel);
predictionTime_CART(run_no,:)=toc;

    predictionSpeed_CART(run_no,:)=(3600/predictionTime_CART(run_no,:)); % number of
observations divided by total time taken to predict them.

% Compute validation accuracy
validationAccuracy_CART(run_no,:) = 1 - kfoldLoss(partitionedModel, 'LossFun', 'ClassifError');
```

Lead City University DO NOT COPY

```
    % Compute confusion matrix
    confMat = confusionmat(response,validationPredictions);

    % Extract TP, FP, FN, and TN
    TP = confMat(2,2); % True Positives
    FP = confMat(1,2); % False Positives
    FN = confMat(2,1); % False Negatives
    % TN = confMat(1,1); % True Negatives (not needed for F1-score)

    % Compute Precision and Recall
    precision = TP / (TP + FP);
    recall = TP / (TP + FN);

    F1(run_no,:) = 2 * (precision * recall) / (precision + recall);

end

% Descriptive Statistics (% Derive F1 Score)

% Validation Accuracy
accuracy_avg_CART=mean(validationAccuracy_CART);
accuracy_best_CART=max(validationAccuracy_CART);
accuracy_worst_CART=min(validationAccuracy_CART);
```

Lead City University DO NOT COPY

```
accuracy_median_CART=median(validationAccuracy_CART);

accuracy_std_CART=std(validationAccuracy_CART);


% F1

F1_avg_CART=mean(F1);

F1_best_CART=max(F1);

F1_worst_CART=min(F1);

F1_median_CART=median(F1);

F1_std_CART=std(F1);



% Confusion Matrix

figure(1)

cMatrix1=confusionchart(response,validationPredictions,'RowSummary','row-
normalized','ColumnSummary','column-normalized');

cMatrix1.FontSize = 20;

cMatrix1.DiagonalColor = 'yellow';

cMatrix1.OffDiagonalColor = 'red';

figure(2)

cMatrix2=confusionchart(response,validationPredictions);

cMatrix2.FontSize = 20;

cMatrix2.DiagonalColor = 'yellow';

cMatrix2.OffDiagonalColor = 'red';
```

Lead City University DO NOT COPY

```
% Generate ROC Curves
diff_score1=zeros;
for i=1:size(validationScores,1)
   temp=validationScores(i,:);
   diff_score1(i,:)=temp(1)-max([temp(2),temp(3),temp(4)]);
end
[X1,Y1,T1,AUC1,OPTROCPT1,suby1,subnames1] = perfcurve(response,diff_score1,'HTTP');

diff_score2=zeros;
for i=1:size(validationScores,1)
   temp=validationScores(i,:);
   diff_score2(i,:)=temp(2)-max([temp(1),temp(3),temp(4)]);
end
[X2,Y2,T2,AUC2,OPTROCPT2,suby2,subnames2] = perfcurve(response,diff_score2,'Normal');

diff_score3=zeros;
for i=1:size(validationScores,1)
   temp=validationScores(i,:);
   diff_score3(i,:)=temp(3)-max([temp(1),temp(2),temp(4)]);
end
[X3,Y3,T3,AUC3,OPTROCPT3,suby3,subnames3] = perfcurve(response,diff_score3,'TCP');

diff_score4=zeros;
```

Lead City University DO NOT COPY

```
for i=1:size(validationScores,1)
   temp=validationScores(i,:);
   diff_score4(i,:)=temp(4)-max([temp(1),temp(3),temp(2)]);
end
[X4,Y4,T4,AUC4,OPTROCPT4,suby4,subnames4] = perfcurve(response,diff_score4,'UDP');

figure(3)
plot(X1,Y1,'LineWidth',2.5)
hold on
plot(X2,Y2,'LineWidth',2.5)
plot(X3,Y3,'LineWidth',2.5)
plot(X4,Y4,'LineWidth',2.5)
plot(OPTROCPT1(1),OPTROCPT1(2),'r*','LineWidth',3)
plot(OPTROCPT2(1),OPTROCPT2(2),'r*','LineWidth',3)
plot(OPTROCPT3(1),OPTROCPT3(2),'r*','LineWidth',3)
plot(OPTROCPT4(1),OPTROCPT4(2),'r*','LineWidth',3)
xlabel('False Positive Rate (FPR)')
ylabel('True Positive Rate (TPR)')
legend1=legend('HTTP (+ Class)', 'Normal (+ Class)', 'TCP (+ Class)', 'UDP (+ Class)','Optimal Operating
Point');
set(legend1,'FontSize',15);
set(gca,'FontSize',15,'FontWeight','bold');
title('ROC Curves for CART Classifier')
hold off
```

Lead City University DO NOT COPY

```
txt = {strcat(['AUC (HTTP (+ Class)): ',num2str(AUC1)]),strcat(['AUC (Normal (+ Class)): ',num2str(AUC2)]),strcat(['AUC (TCP (+ Class)): ',num2str(AUC3)]),strcat(['AUC (UDP (+ Class)): ',num2str(AUC4)])};

text(0.25,0.3,txt,'Color','black','FontSize',15,'FontWeight','bold')
```

Lead City University DO NOT COPY

Appendix 2 (MATLAB Code for ANN)

```
clear

clc

close all force


simulation_data = xlsread('sorted_sdn_data.xlsx');


% Throughput mathematical/statistical evaluation and normalization

Tp_data = simulation_data(:,1);

Tp_data_norm = norm_func(Tp_data);


% Jitter mathematical/statistical evaluation and normalization
```

```
Jt_data = simulation_data(:,2);

Jt_data_norm = norm_func(Jt_data);


% Response mathematical/statistical evaluation and normalization

Rt_data = simulation_data(:,3);

Rt_data_norm = norm_func(Rt_data);


input = [Tp_data_norm, Jt_data_norm, Rt_data_norm]';


% Shuffle data indices

index = randperm(10800);


% Calculate Targets (Normalized Cost Function for SDN performance Evaluation)

Lw_data = simulation_data(:,4) * 10; % Scale Link Weight (Lw)


% Create a normalized dataset including link weight

norm_simulation_data = [Tp_data_norm, Jt_data_norm, Rt_data_norm, Lw_data];


% Initialize cost function array

CF = zeros(size(norm_simulation_data,1), 1);


for i = 1:size(norm_simulation_data,1)

    sample_data = norm_simulation_data(i,:);

    Tp = sample_data(1);
```

Lead City University DO NOT COPY

```
    Jt = sample_data(2);
    Rt = sample_data(3);
    Lw = sample_data(4);

    CF(i) = (abs(Tp - Jt) * Rt) * Lw;
end

% Normalize the target values
targets = norm_func(CF)';

%% Initialize Parameters
num_runs = 50; % Total number of independent ANN runs
performances = zeros(num_runs, 1); % Store MSE values for each run

%% Loop Over 50 Independent Runs
for run = 1:num_runs
    x = input(:, index);
    t = targets(index);

    % Data Splitting: 70% Training, 15% Validation, 15% Testing
    trainInd = index(1:7560); % Training set (70%)
    valInd = index(7561:9180); % Validation set (15%)
    testInd = index(9181:end); % Testing set (15%)
```

Lead City University DO NOT COPY

```
xTrain = x(:, trainInd);

xValid = x(:, valInd);

xTest = x(:, testInd);


tTrain = t(trainInd);

tValid = t(valInd);

tTest = t(testInd);


% Define and configure artificial neural network

trainFcn = 'trainlm'; % Levenberg-Marquardt training algorithm

hiddenLayerSize = 10; % Number of neurons in hidden layer

net = feedforwardnet(hiddenLayerSize, trainFcn); % Create ANN model


% Assign the manually split data

net.divideFcn = 'divideind';

net.divideParam.trainInd = trainInd; % Training indices

net.divideParam.valInd = valInd; % Validation indices

net.divideParam.testInd = testInd; % Testing indices

net.performFcn = 'mse'; % Use Mean Squared Error as performance metric


% Train the artificial neural network

[net, tr] = train(net, x, t);


% Predict output
```

Lead City University DO NOT COPY

```
    y = net(x);

    % Compute Mean Squared Error (MSE)
    performance(run) = mse(net, t, y);
end

% Plot Cost Function Trend
figure;
plot(1:2700, CF(1:2700), 'g-', 'LineWidth', 2);
hold on;
plot(2701:5400, CF(2701:5400), 'b-', 'LineWidth', 2);
hold on;
plot(5401:8100, CF(5401:8100), 'y-', 'LineWidth', 2);
hold on;
plot(8101:10800, CF(8101:10800), 'r-', 'LineWidth', 2);
hold off;
grid on;
set(gca, 'FontSize', 15, 'FontWeight', 'bold');
xlabel('Number of Simulated Scenarios', 'FontSize', 15, 'FontWeight', 'bold');
ylabel('Cost Function Value', 'FontSize', 15, 'FontWeight', 'bold');
legend({'Normal Operating Condition', 'TCP Flood Attack', 'UDP Flood Attack', 'HTTP Flood Attack'},
'FontSize', 10);

%% Noisy Tp, Jt, Rt for 50 Runs
```

Lead City University DO NOT COPY

```
for run = 1:num_runs
    Tp_data_noise = Tp_data + sqrt(1) * randn(size(Tp_data));
    Tp_data_norm_noise = norm_func(Tp_data_noise);
    input = [Tp_data_norm_noise, Jt_data_norm, Rt_data_norm]';
    index = randperm(10800);
    y_noise = net(input(:, index));
    Tp_performance(run) = mse(net, t, y_noise);

    Jt_data_noise = Jt_data + sqrt(1) * randn(size(Jt_data));
    Jt_data_norm_noise = norm_func(Jt_data_noise);
    input = [Tp_data_norm, Jt_data_norm_noise, Rt_data_norm]';
    index = randperm(10800);
    y_noise = net(input(:, index));
    Jt_performance(run) = mse(net, t, y_noise);

    Rt_data_noise = Rt_data + sqrt(1) * randn(size(Rt_data));
    Rt_data_norm_noise = norm_func(Rt_data_noise);
    input = [Tp_data_norm, Jt_data_norm, Rt_data_norm_noise]';
    index = randperm(10800);
    y_noise = net(input(:, index));
    Rt_performance(run) = mse(net, t, y_noise);
end
```

Lead City University DO NOT COPY

Lead City University DO NOT COPY

Biodata

A. Personal Data

Full Name: Chekwube Ezechi

Address: Aweni Apartments opposite Soka City Hospital, Ireakari Estate, Soka Ibadan.

Email address: zechirayjr@yahoo.com, chekwube.ezechi@lcu.edu.ng

Phone number: +234(0)8028116377

Date and Place of Birth: 26th December, 1991; Lagos State.

Nationality: Nigerian

Marital status: Single

Name of Next of Kin: Uchenna McOliver Ezechi

B. Educational Background

Primary Education

i) Cedec International Schools

First School Leaving Certificate. 1993-2001

Secondary Education

i) Nigerian Navy Secondary School, Navy Town, Ojo, Lagos State.

West African School Certificate. 2001-2007

Higher Education

i) Lead City University, Ibadan, Oyo State

PhD in Computer Science 2022 - Present

ii) Chukwuemeka Odumegwu Ojukwu University, Uli, Anambra State

M.Sc. in Computer Science. 2018-2021

iii) Chukwuemeka Odumegwu Ojukwu University, Uli, Anambra State

B.Sc. in Computer with Electronics. 2011-2015

C. 1. Work Experience

i) Nextworks Limited, Lagos State.

IT Support 2015 – 2016.

ii) Immanuel Model College, Gboko, Benue State.

Teaching Assistant (NYSC) 2017- 2018

iii) Zurichat (Remote)

Frontend Developer 2021 - 2022.

Lead City University DO NOT COPY

iv) Lead City University, Ibadan, Oyo State

Research Assistant 2022 - 2025.

D. Publications

Thesis/Dissertation

i) Design and Implementation of A Computerized Visa Processing Information System (BSc Project)

ii) Design and Implementation of An Integrated System for Continuous Assessment and Exam Management (MSc Dissertation)

iii) A Hybrid Anomaly Characterisation and Sensitivity Analysis Framework for Software-Defined Network-Aided Communication (PhD Thesis)

Published Papers

i) "A Hybrid Framework for the Sensitivity Analysis of Software-Defined Networking Performance Metrics Using Design of Experiments and Machine Learning Techniques". Chekwube Ezechi, Mobayode O. Akinsolu, Wilson Sakpere, Abimbola O. Sangodoyin, Uyoata E. Uyoata, Isaac Owusu-Nyarko and Folahanmi T. Akinsolu. (MDPI, Volume 16, Issue 9, September 2025)

Lead City University DO NOT COPY

ii) "Artificial Intelligence-Driven Sensitivity Analysis: Present-Day Approaches in Software-Defined Networking". Chekwube Ezechi, Mobayode O. Akinsolu, Wilson Sakpere* Abimbola O. Sangodoyin, Folahanmi T. Akinsolu (5th IEEE International Conference on Emerging Trends in Electrical, Electronic and Communications Engineering November 2024). ISBN 979-8-3315-3246-8.

iii) "Software-Defined Networking in Cyber-Physical Systems". Chekwube Ezechi, Mobayode O. Akinsolu, Wilson Sakpere, Abimbola O. Sangodoyin, Folahanmi T. Akinsolu (Cyber Physical System 2.0: Communication and Computational Technologies, p.44, December 2024).

iv) "Selected Applications of Artificial Intelligence in Pursuing the Sustainable Development Goals". Mobayode O. Akinsolu, Chekwube Ezechi, Wilson Sakpere, Mingwei Sun (7th EAI International Conference on Emerging Technologies for Developing Countries, pp. 64-85, July 2024).

E. Major Conferences/Workshop Attended

i. AI, Research and Grant Writing Training (2024).

ii. 7th EAI International Conference on Emerging Technologies for Developing Countries (EAI AFRITEK 2024).

iii. 5th IEEE International Conference on Emerging Trends in Electrical, Electronic and Communications Engineering (ELECOM 2024)

iv. Postgraduate Multidisciplinary International Conference 2023

Lead City University DO NOT COPY

F. Others

i) Certifications

1) AltSchool School of Engineering Certificate

2) Developer Certification Responsive Web Design

Lead City University DO NOT COPY

G. Referees

Dr. Wilson Sakpere

Faculty of Computer Science

Lead City University

wilson.sakpere@lcu.edu.ng

+234(0)8159582869

Dr. Mobayode O. Akinsolu

Faculty of Arts, Computing and Engineering

Wrexham University, UK

mobayode.akinsolu@wrexham.ac.uk

+447857952156

Lead City University DO NOT COPY

_____ _____

Signature Date

University Compliance Form

This is to certify that this thesis by Chekwube EZECHI with Matriculation Number LCU/PG/0003043 in the Department of Computer Science, Faculty of Natural and Applied Sciences, Lead City University, Ibadan is in full compliance with the approval of the University's format and style.

…… ……

Signature Date

Lead City University DO NOT COPY