Journal Article

A Hybrid Framework for the Sensitivity Analysis of Software-Defined Networking Performance Metrics Using Design of Experiments and Machine Learning Techniques

Ezechi, C., Akinsolu, M.O., Sakpere, W., Sangodoyin, A.O., Uyoata, U.E., Owusu-Nyarko, I. and Akinsolu, F.T.

information

MDPI

Article

A Hybrid Framework for the Sensitivity Analysis of Software-Defined Networking Performance Metrics Using Design of Experiments and Machine Learning Techniques

Chekwube Ezechi [1], Mobayode O. Akinsolu [1,2,*], Wilson Sakpere [1], Abimbola O. Sangodoyin [1,3], Uyoata E. Uyoata [4], Isaac Owusu-Nyarko [5] and Folahanmi T. Akinsolu [1]

[1] Faculty of Natural and Applied Sciences, Lead City University, Ibadan 200255, Oyo State, Nigeria; chekwube.ezechi@lcu.edu.ng (C.E.); sakpere.wilson@lcu.edu.ng (W.S.); asangodoyin@lincoln.ac.uk (A.O.S.); akinsolu.folahanmi@lcu.edu.ng (F.T.A.)
[2] Faculty of Arts, Computing, and Engineering, Wrexham University, Wrexham LL11 2AW, UK
[3] School of Engineering and Physical Sciences, University of Lincoln, Lincoln LN6 7TS, UK
[4] Department of Electrical and Electronics Engineering, Modibbo Adama University, Yola 640231, Adamawa State, Nigeria; uyoataue@mau.edu.ng
[5] Department of Electrical and Electronic Engineering, Regional Maritime University, Accra P.O. GP 1115, Ghana; isaac.owusu-nyarko@rmu.edu.gh
* Correspondence: mobayode.akinsolu@wrexham.ac.uk or m.o.akinsolu@ieee.org

Abstract

Software-defined networking (SDN) is a transformative approach for managing modern network architectures, particularly in Internet-of-Things (IoT) applications. However, ensuring the optimal SDN performance and security often needs a robust sensitivity analysis (SA). To complement existing SA methods, this study proposes a new SA framework that integrates design of experiments (DOE) and machine-learning (ML) techniques. Although existing SA methods have been shown to be effective and scalable, most of these methods have yet to hybridize anomaly detection and classification (ADC) and data augmentation into a single, unified framework. To fill this gap, a targeted application of well-established existing techniques is proposed. This is achieved by hybridizing these existing techniques to undertake a more robust SA of a typified SDN-reliant IoT network. The proposed hybrid framework combines Latin hypercube sampling (LHS)-based DOE and generative adversarial network (GAN)-driven data augmentation to improve SA and support ADC in SDN-reliant IoT networks. Hence, it is called DOE-GAN-SA. In DOE-GAN-SA, LHS is used to ensure uniform parameter sampling, while GAN is used to generate synthetic data to augment data derived from typified real-world SDN-reliant IoT network scenarios. DOE-GAN-SA also employs a classification and regression tree (CART) to validate the GAN-generated synthetic dataset. Through the proposed framework, ADC is implemented, and an artificial neural network (ANN)-driven SA on an SDN-reliant IoT network is carried out. The performance of the SDN-reliant IoT network is analyzed under two conditions: namely, a normal operating scenario and a distributed-denial-of-service (DDoS) flooding attack scenario, using throughput, jitter, and response time as performance metrics. To statistically validate the experimental findings, hypothesis tests are conducted to confirm the significance of all the inferences. The results demonstrate that integrating LHS and GAN significantly enhances SA, enabling the identification of critical SDN parameters affecting the modeled SDN-reliant IoT network performance. Additionally, ADC is also better supported, achieving higher DDoS flooding attack detection accuracy through the incorporation of synthetic network observations that emulate real-time traffic. Overall, this work highlights the potential of hybridizing LHS-based DOE, GAN-driven data augmenta-

Academic Editors: Xu Zheng, Zhuojun Duan, Yingjie Wang and Lorenzo Mucchi

Received: 30 March 2025
Revised: 5 August 2025
Accepted: 28 August 2025
Published: 9 September 2025

Citation: Ezechi, C.; Akinsolu, M.O.; Sakpere, W.; Sangodoyin, A.O.; Uyoata, U.E.; Owusu-Nyarko, I.; Akinsolu, F.T. A Hybrid Framework for the Sensitivity Analysis of Software-Defined Networking Performance Metrics Using Design of Experiments and Machine-Learning Techniques. *Information* **2025**, *16*, 783. https://doi.org/10.3390/info16090783

tion, and ANN-assisted SA for robust network behavioral analysis and characterization in a new hybrid framework.

Keywords: artificial neural network (ANN); design of experiments (DOE); generative adversarial networks (GANs); classification and regression tree (CART); Internet of Things (IoT); machine learning (ML); Latin hypercube sampling (LHS); sensitivity analysis (SA); software-defined networking (SDN)

1. Introduction

Recent projections have indicated that the number of connected devices globally will reach several billions in the coming years, due to the proliferation of internet technologies, such as the Internet of Things (IoT) and its associated systems and devices [1,2]. A parallel trend is also noticeable in the growth of cloud technologies, including cloud-based software, surveillance solutions, real-time network automation, and distributed connections [3]. In relation to this, software-defined networking (SDN), which introduces a decoupling of control and data planes in IoT networks, has also become a prevalent technology, highlighting the need for scalable and intelligent network management solutions [4]. Both past and ongoing research point to SDN as a foundational component of IoT architectures [5–7]. One early contribution in this area employed multilayer SDN controllers specifically designed for heterogeneous vehicular IoT traffic, demonstrating end-to-end quality of service (QoS) guarantees through network-calculus-based scheduling mechanisms [8].

SDN, which leverages the OpenFlow protocol, is a rapidly evolving technology that significantly enhances network management, administration, and monitoring processes [4]. By enabling programmable interaction with data plane elements and providing network administrators with a holistic view of the network through a controller, SDN positions the controller as the network's intelligence, which is responsible for managing devices in the data plane via the application-programming interface (API) [4]. This capability has led to the adoption of SDN-based frameworks in advanced IoT network topologies, including fifth-generation (5G) and next-generation network architectures, such as sixth-generation (6G) [9–11].

While the programmability, flexibility, and decentralized control of SDN are critical to its effective implementation, these same features can introduce significant security challenges [12]. IoT networks and other SDN-reliant systems are susceptible to a number of security risks, such as an increased likelihood of distributed-denial-of-service (DDoS) flooding attacks [13,14]. These inherent vulnerabilities associated with SDN highlight the pressing need for advanced and intelligent security frameworks that are capable of mitigating evolving threats, such as DDoS flooding attacks, while preserving the operational benefits of SDN-based architectures in network environments, such as IoT systems.

DDoS flooding attacks are carefully planned assaults that use a network of hacked devices, called botnets, to overwhelm a system. The goal is to clog up the network's bandwidth or shut down specific servers and devices, making them unusable for regular users [15]. These attacks constitute a significant portion of the most dangerous malicious traffic on the internet [16]. Typically, attackers deploy these botnets to execute these operations covertly [17]. As a result, end users of the targeted network's device nodes often remain clueless that their devices and internet protocol (IP) addresses are being used to perpetrate DDoS flooding attacks. IoT devices are especially vulnerable to coordinated DDoS flooding attacks because of how they are built and the fact that the internet does not enforce strict traffic control on individual devices [16]. This combination makes it easier

for attackers to target and overwhelm IoT devices. This vulnerability is made even worse by the rapid growth in the number of IoT devices out there, even though their security is improving over time [18]. In the face of increasingly complex DDoS flooding attacks, the critical need for innovative and proactive security techniques that can address the particular vulnerabilities of IoT networks becomes imperative.

As IoT networks that utilize SDN architectures expand in complexity and scale, optimizing their performance and security presents significant challenges that demand robust and well-optimized network configurations [19]. Identifying critical network performance metrics and assessing their influences on the overall network performance through sensitivity analysis (SA) have consistently been put forward as pivotal tools for enhancing the reliability of network configurations [20]. This is primarily because developing a reliable IoT network model requires a comprehensive understanding of how variations in model parameters influence the network's behavior and the accuracy of its predictive outcomes. SA plays critical roles in this process by systematically analyzing the importance of input parameters, evaluating their contributions to the network model's outputs, and quantifying the impacts of individual inputs on the overall system performance [21].

SA techniques are commonly categorized based on their scope, as either local or global, and their framework, as either deterministic or statistical [22,23]. As discussed in [21,24,25], statistical frameworks for SA are generally derived from principles associated with the design of experiments (DOE). These frameworks also classify SA methods into local and global approaches, depending on the parameter space under consideration and the specific objectives to be achieved [26]. SA, both local (LSA) and global (GSA), is critical for understanding the influences of different parameters on the performance of IoT networks that are SDN reliant [27]. Therefore, advancing the application of SA in SDN-reliant IoT networks is vital for developing adaptive and data-driven models that enhance predictive accuracy and system resilience.

The application of artificial intelligence (AI) techniques (machine-learning (ML) techniques in particular) is growing significantly in numerous fields and disciplines, including SA, within SDN [10,26]. In the context of the local SA (LSA), ML-based approaches often rely on gradient-based methods to evaluate how specific parameters affect performance metrics [28]. Techniques such as gradient-boosting machines (GBMs) and artificial neural networks (ANNs) or neural networks are widely used to approximate gradients efficiently to support parameter tuning and optimization [28,29]. ANNs, in particular, offer enhanced capabilities to capture complex dependencies between parameters and performance for ML-assisted LSA in SDN environments [30]. These capabilities not only facilitate the identification of critical parameters but also provide actionable insights for improving SDN configurations for IoT networks.

For global SA (GSA), ML techniques, such as the Gaussian process [31], can play an important role in improving traditional methods, such as Monte Carlo simulations [32]. By training ML models on subsets of data, predictions can be extended across a larger parameter space, enabling a more comprehensive analysis [33]. Gaussian processes are also effective for modeling the distribution of network performance metrics under varying parametric conditions, providing a probabilistic framework that supports GSA [34]. Furthermore, Monte Carlo simulations, when combined with these ML models, provide a detailed evaluation of how parameter variations affect the overall network performance [35]. ML techniques, such as regression methods for surrogate modeling, further streamline GSA by approximating the behaviors of complex network systems, reducing computational costs, and improving scalability [26,35]. These ML-based approaches emphasize the potential of ML-assisted SA methodologies to optimize SDN configurations for IoT network

applications and improve their performance and reliability in increasingly complex IoT network environments.

Practitioners often aim to enhance the effectiveness and efficiency of ML models by regulating and optimizing key processes in SDN environments [36]. This practice inherently aligns with the principles of DOE, a data-driven approach that plays a crucial role in SA [25,37,38]. DOE provides a structured experimental framework that is essential for addressing the complexities of both SA and ML challenges, particularly in the behavioral analysis of IoT networks [39]. For instance, QoS is a vital metric in IoT networks [40]. DOE can be used to systematically identify relationships between factors that influence QoS in IoT networks, thus improving network performance while minimizing the need for extensive experimental trials [25,41]. Furthermore, DOE can facilitate a targeted investigation of cause-and-effect relationships in scenarios involving the application of ANNs and evolutionary algorithms (EAs), both of which are commonly applied in IoT networks [42,43]. This structured approach reduces reliance on trial-and-error methods, saving time and computational resources. Hence, the integration of DOE with ML clearly offers great potential for enhancing model interpretability, reproducibility, and optimization efficiency within complex network environments, such as IoT networks.

A popular example of DOE in practice is Latin hypercube sampling (LHS) [44,45]. LHS is a widely used technique for addressing complex and high-dimensional problems, including ML-assisted global optimization of complex models [46,47]. LHS works by stratifying the sample space to ensure uniform sampling of each variable across its range, which is particularly advantageous for simulation-based optimization tasks in computational intelligence [46]. This approach significantly improves the convergence rates of ML models, such as ANNs, making it a valuable tool in data-driven analytics [47]. By integrating DOE methodologies, such as LHS, with ML for SA implementation, more robust insights can be obtained about model behavior, enabling the development of more reliable and resilient network systems. This is why extending the application of DOE within ML-assisted SA frameworks remains a potential direction for research efforts aimed at optimizing the performance and reliability of network systems, such as SDN-reliant IoT networks.

Generative adversarial networks (GANs), a widely used class of ML frameworks, have also become quite popular in recent years due to their ability to generate synthetic data by learning the underlying distribution of real-world datasets [48]. Originally introduced in [49], GANs typically comprise two neural networks: a generator, which creates synthetic data, and a discriminator, which aims to differentiate between generated and real data [50]. These networks are trained simultaneously in a zero-sum game, where the generator continuously and iteratively refines its output to deceive the discriminator [51]. Due to this continuous iterative process, the generator can create extremely realistic synthetic data, making GANs an effective tool for a variety of computing and networking applications [52,53]. The growing roles of GANs in optimizing the performance and addressing the complexities of modern network systems [54] further emphasize their potential to support ML-driven SA in SDN-reliant IoT networks.

This work introduces a new data-driven framework that hybridizes LHS-based DOE, GAN-based synthetic data generation, and ANN-assisted SA to facilitate an enhanced behavioral study (including anomaly detection and classification (ADC)) in SDN-reliant IoT networks. Termed as DOE-GAN-SA, the proposed hybrid framework leverages simulated network scenarios, LHS-driven augmentation of network performance data, GAN-enabled synthetic data generation, and ANN-assisted SA to provide a comprehensive approach to behavioral analysis (including ADC) within SDN-reliant IoT networks. DOE-GAN-SA explores the harmonious co-working of LHS-based DOE, GAN-driven synthetic data generation, and ANN-based supervised learning by focusing on their complementary roles

to support ADC and SA in SDN-reliant IoT network environments. The key contributions of this work include:

- Using LHS and GAN for data augmentation in SDN-reliant IoT networks;
- Showing the application of a newly augmented SDN-reliant IoT network dataset for detecting and classifying DDoS flooding attacks;
- Improving the mitigation of DDoS flooding attacks through improved detection accuracy and ANN-assisted SA;
- Demonstrating DOE-GAN-SA as a new hybrid ADC and SA framework for SDN-reliant IoT networks.

To the best of our knowledge, and based on the literature reviewed in Section 2, this study is the first to introduce a hybrid approach combining DOE and ML techniques for the specific purpose of conducting SA on network performance metrics within SDN-dependent IoT networks. Although the individual methodologies employed are well-established within their respective domains, this work represents the first documented integration of these techniques for this targeted application. The structure of the remainder of this paper is as follows: Section 2 reviews the recent literature relevant to the research conducted to provide a foundation for the study. Section 3 outlines the SDN architecture and IoT network topology utilized to simulate various network scenarios to establish the technical context. The basic techniques guiding the formulation of the proposed DOE-GAN-SA framework are introduced in Section 4, and the proposed DOE-GAN-SA framework is detailed in Section 5, highlighting its components and functionalities. Section 6 describes the experimental setup, presents the results, and discusses the findings, offering insights into DOE-GAN-SA's performance, scope, and recommended approach for its practical adoption. Finally, Section 7 provides the concluding remarks, summarizing the key contributions and potential directions for future work.

2. Related Work

SDN is a widely used approach for scaling heterogeneous IoT deployments, primarily due to its decoupling of control and data planes. However, much of the existing research in this area still relies heavily on heuristic-based parameter tuning, often overlooking the importance of feedback mechanisms that link anomaly detection with flow-rule adaptation [55]. Early contributions, such as [8], demonstrated the feasibility of SDN for vehicular IoT infrastructures via the multipurpose infrastructure for network applications (MINA)-SDN controller, yet their study did not address the critical constraints related to sensor energy consumption and embedded security mechanisms. Subsequent works, including the survey by [56], emphasized the necessity of slice isolation and controller scalability in supporting large-scale IoT deployments. Similarly, Ref. [5] introduced software-defined APIs designed for smart city infrastructures to enable the shared use of gateways and cloud services, accompanied by some quantitative analysis using a case study. More recently, Ref. [6] showcased a multilevel architecture that effectively reduced packet loss in smart home networks; however, several challenges persist. Further highlighting these challenges, a comprehensive meta-analysis, conducted by [7] and spanning over 160 studies, identified microservice-based controllers, flow table compression techniques, and energy-efficient routing as pivotal components in the secure deployment of SDN-reliant IoT systems for smart communities. Collectively, these studies highlight the importance of programmability and slice isolation, among other design strategies, in supporting the development of secure and scalable IoT infrastructures. Nonetheless, none of the reviewed approaches adequately addresses data-scarce SA or integrates ADC mechanisms directly within SDN architectures, an evident gap that the DOE-GAN-SA framework proposed in this work seeks to bridge.

While GANs have found several applications across multiple disciplines [57], their adoption for ADC and SA, particularly in SDN-reliant IoT networks, can be said to still be in its early stages, arguably. Since the introduction of GANs in [49], numerous studies have demonstrated their practicality in data augmentation. For instance, the work in [58] investigated the use of GANs to generate realistic network traffic data, enabling the simulation of network behaviors under various conditions. However, this study, like other similar recent works [59], did not specifically address ADC and SA, highlighting the untapped prospect of GAN-based synthetic data generation in supporting robust ADC and SA within SDN-reliant IoT network environments. Supporting this perspective is a recent study that demonstrated the effectiveness of GANs in augmenting datasets necessary for analyzing complex high-dimensional systems and improving classification performance [60]. Despite these promising developments, the current literature indicates that the full potential of GANs in the context of ADC and SA within SDN-reliant IoT networks still remains largely unexplored, offering significant opportunities for further research. Hence, this current work explores the integration of GANs into ADC and SA workflows with the aim of exploiting their synthetic data generation capabilities for deeper insight into parameter-driven behaviors within complex network environments.

SA in SDN-reliant networks, such as IoT networks, has traditionally relied on standard techniques, like variance-based methods and Monte Carlo simulations [24,61]. These conventional approaches, while effective for several applications, often require large datasets and substantial computational resources [62], making them less practical for low-latency, real-time, large-scale systems, such as IoT networks [63]. Variance-based SA methods, for example, are very practical in measuring the contribution of each input factor to the overall variance of the output, which helps to identify key influential parameters [30,64]. However, in scenarios that involve a large number of parameters or complex interaction effects, they could become impractical due to increased computational demands and instability in estimations [65]. To address these challenges, researchers have explored alternative techniques, such as sample-based estimations, that reduce computational overhead without sacrificing analytical accuracy [66]. However, this is often at the cost of introducing new design parameters [66].

Statistical sampling methods, such as LHS, are well-known for their efficiency in parameter selection and analysis [67]. Unlike random sampling, LHS ensures a more uniform coverage of the input space to improve robustness and minimize computational costs [47]. Despite its advantages, LHS is not without its own limitations. One significant drawback of LHS is its inability to thoroughly account for statistical relationships between input variables [47,67], which can compromise SA accuracy in systems such as SDN-reliant IoT networks, where inputs could be highly correlated [39]. Combining LHS with GAN offers a promising solution to this limitation. GANs are capable of generating synthetic datasets that tend to preserve some statistical properties of real-world network data while also reasonably covering the input space of real-world network traffic comprehensively and robustly [68]. By combining LHS-based data and GAN-generated data into a unified framework, harmonizing their advantages in data augmentation could be realized as carried out in this work. This hybrid approach has the potential to improve SA in SDN-reliant IoT networks, enabling more effective analyses of such complex and high-dimensional systems while meeting the demands of real-time applications.

Although GANs have demonstrated significant potential in synthetic data generation across various applications [69], as established earlier, their application for SA in SDN-reliant networks, such as IoT networks, remains underexplored. Much of the existing research in this area has concentrated more on enhancing GAN architectures for data generation or refining synthetic data generation techniques using GANs [69,70]. Therefore,

there is a gap in studies that harness the complementary strengths of GANs and LHS to perform SA in SDN-reliant IoT networks. Conventional SA frameworks, while effective in many scenarios, are often purpose built and may not necessarily offer additional insights into other operations, such as data augmentation and ADC, when implemented. These limitations also emphasize the need for innovative approaches that can efficiently address the complexities of modern network systems by providing more robust insights while maintaining analytical precision and complementing existing SA methods. Hence, exploring a hybrid SA framework that combines DOE and ML techniques, as typified in this work, can potentially redefine the methodological landscape for analyzing intelligent, large-scale network infrastructures in a resource-efficient and scalable manner.

Specifically, this work addresses the existing gaps discussed above by proposing the DOE-GAN-SA framework, which integrates LHS-based DOE with GAN-driven synthetic data generation to enable efficient and scalable ADC and ANN-assisted SA in SDN environments. GANs are leveraged to generate high-quality synthetic datasets that preserve some properties of real-world network data while enhancing data diversity. This capability is paired with the structured uniform sampling offered by LHS, which ensures comprehensive coverage of the input parameter space with reduced computational costs. By combining these methodologies, the DOE-GAN-SA framework facilitates a more robust and comprehensive approach to SA and ADC, enabling the precise identification of critical parameters that impact the SDN performance. As a result, the DOE-GAN-SA framework introduced in this study is expected to help create more efficient and reliable IoT network setups by enhancing ADC and improving SA for SDN-reliant IoT networks.

3. SDN Architecture and IoT Network Topology

This section introduces the SDN architecture and IoT network topology employed in this work. Additionally, it outlines the simulation of traffic scenarios conducted using the SDN-reliant IoT network model and methodologies established in the authors' earlier works to make the study more self-contained [30,39,71]. The subsections cover the architecture and topology setup of the modeled SDN-reliant IoT network, the network traffic simulation approach, and the integration of insights from previous research to validate the SDN-reliant IoT network model.

3.1. SDN Architecture

In contrast to conventional networking architectures, SDN architectures principally work by employing a fundamental structure around the decoupling of the control and data planes to enable a more centralized and intelligent management of network devices [72]. This architecture is critical for achieving programmability and dynamic network configuration. As depicted in Figure 1, the SDN architecture typically consists of three distinct planes: the application, control, and data planes, each serving specialized roles. The application plane encompasses applications, such as intrusion detection systems, intrusion prevention systems, and security-monitoring tools, which provide critical inputs for decision making [73]. The control plane, dominated by the SDN controller, governs network operations through security policies, routing strategies, databases, and rule placement algorithms [10]. Finally, the data plane comprises infrastructure devices responsible for executing network operations based on predefined rules, actions, and counters [10].

In the SDN architecture typified in Figure 1, the controller operates as the centralized decision-making entity, bridging the application and data planes [4]. By leveraging inputs from applications in the application plane, the controller determines appropriate actions for the data plane [14]. Forwarding devices within the data plane, implemented in hardware or software, execute these instructions by handling packet routing and forwarding based

on established forwarding policies [14]. This arrangement, illustrated in Figure 1, allows for efficient policy implementation and dynamic network adaptability. For this study, the described SDN architecture forms the foundational framework, enabling the exploration of SDN in enhancing IoT network security and operational efficiency.

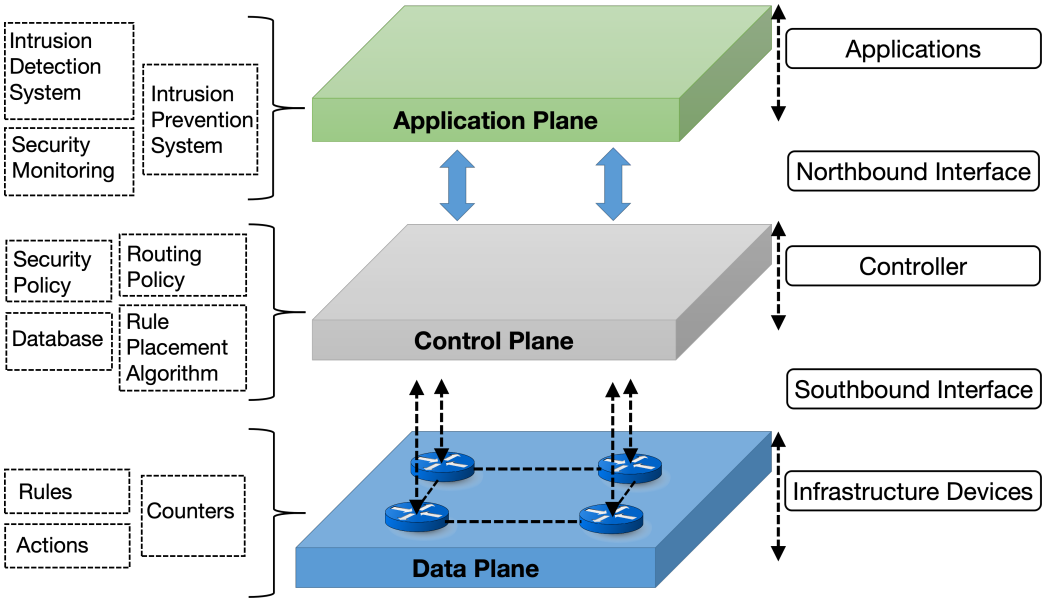


Figure 1. Typical SDN architecture.

3.2. IoT Network Topology

IoT networks are characterized by their diverse topologies (mostly inherited from traditional data and communication networks) that enable flexible arrangements and interconnections of devices [74]. Among the common topologies employed are mesh, star, tree, and point-to-point configurations, each offering specific advantages depending on the application context [75]. Within IoT systems, the data center (DC) serves as the central hub, which manages substantial traffic and ensures efficient data processing. For DC architectures, the Fat-Tree and BCube topologies are widely utilized [76]. The Fat-Tree topology shown in Figure 2, in particular, is favored in this study due to its scalability, consistent bandwidth distribution, and cost-effective design [77].

By utilizing switches of uniform capacity, Fat-Tree networks support predictable performance and enable line-speed transmission when packets are evenly distributed [78]. These characteristics provide an ideal foundation for IoT networks that require efficient and scalable data management. IoT networks are generally multilayered [79] and highly reliant on DCs that are also a part of the network infrastructure. This is especially true when taking into account the data-processing, storage, and management (including traffic management) levels in IoT networks [80]. The network topology used in this work, therefore, closely resembles a typical IoT network as previously established [71]. Similar IoT architectures, featuring tree topologies, hierarchical layouts, and multilayer network designs, have also been reported in previous studies [6,7,56,81]. Therefore, the custom SDN architecture in this work adopts an IoT network with a tree topology (see Figure 2), facilitating wide-area network adaptability and scalability.

Using emulators or simulation techniques to analyze network behavior and characteristics before deployment is a common practice in network management [6,82,83]. Among these tools, Mininet is widely recognized in SDN research for its effectiveness [6,71,82,83]. Numerous studies have demonstrated that networks modeled within the Mininet environment accurately represent real-world SDN scenarios [6,82,83]. The SDN-reliant IoT network model adopted in this work was implemented using the Mininet emulator on a

system featuring 32 GB of RAM and Kali Linux operating on an Intel Xeon E3-1220 CPU. Floodlight controller [71] was hosted in VirtualBox, running on Ubuntu 18.10 LTS, while the Mininet environment operated on Ubuntu 16.10 LTS. The experimental setup consisted of 10 OpenFlow switches and 16 interconnected hosts connected by 100 Mbps links. Though relatively small, this network configuration reflects real-world scenarios [30,39,71], such as enterprise or campus networks. By bridging the simplicity of a tree topology with the advanced features of SDN, this study explores the sensitivity and vulnerabilities of IoT networks under DDoS flooding attacks, highlighting the adaptability of tree and/or Fat-Tree approaches in designing resilient network infrastructures. It should also be noted that the network topology used in this work has also been adopted in the authors' previous works [30,39,71].

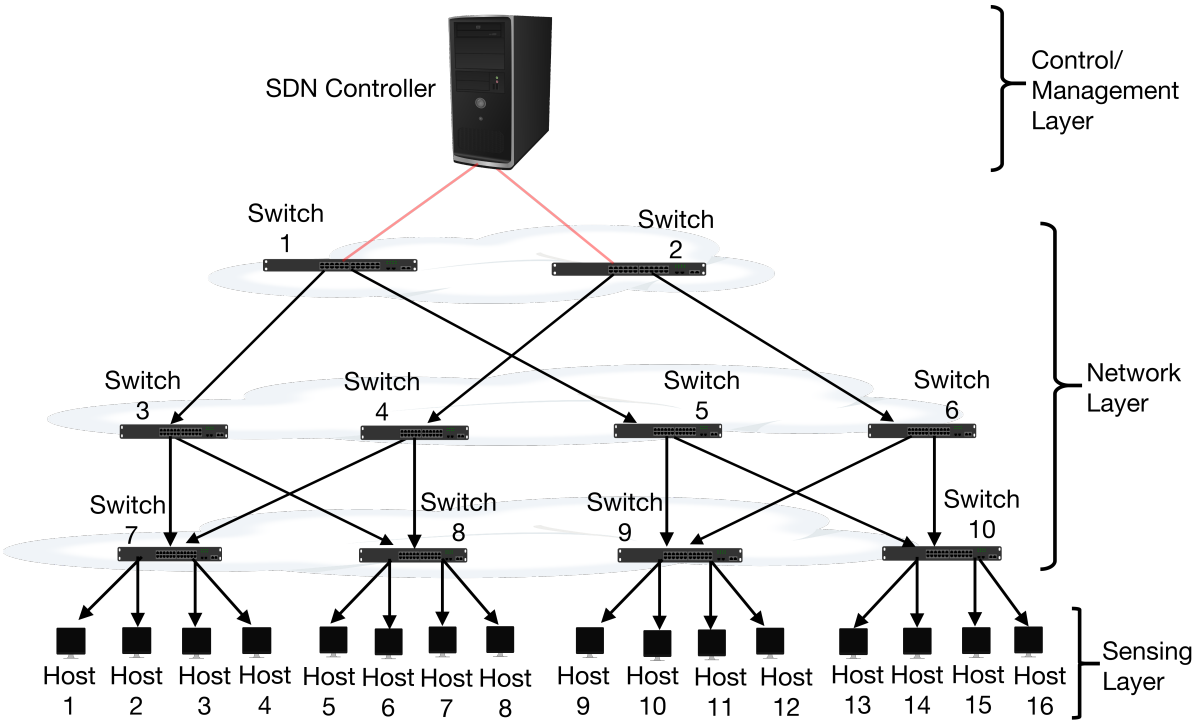


Figure 2. Illustration of the modeled SDN-reliant IoT network architecture (a typification).

3.3. Simulation of Traffic Scenarios in the SDN-Reliant IoT Network Model

To simulate non-malicious and malicious traffic in the SDN environment, the tools *iperf* and *ping* were employed to generate legitimate network communication between nodes, while the Low-Orbit Ion Cannon (LOIC) was utilized to execute DDoS flooding attacks. The experiments involved six compromised hosts (hosts six to eight and ten to twelve, see Figure 2) used to target the network server with sequential HTTP (hypertext transfer protocol), TCP (transmission control protocol), and UDP (user datagram protocol) DDoS flooding attacks, each lasting 15 min and resulting in a total attack duration of 45 min per experimental round. During these periods, critical system properties (throughput (T_p), jitter (J_t), and response time (R_t)) were recorded every second for analysis to have 900 observations per DDoS flooding attack. These properties represent essential performance metrics: T_p quantifies the actual data transfer rate within the network, J_t measures the variation in packet transmission delays, and R_t indicates the latency between task initiation and completion [39]. The data were collected under both normal traffic conditions (900 observations) and DDos flooding attack conditions (2700 observations), providing comprehensive insights into network performance variations.

3.4. Performance Metrics of the SDN-Reliant IoT Network Model

To specifically generate the dataset for the performance metrics of the SDN-reliant IoT network model, a structured methodology was followed, starting with a confirmation of the connectivity using the *pingall* command. *Iperf* was then used to create TCP and UDP servers at different ports, allowing hosts to send traffic to measure baseline or normal T_p, J_t, and R_t metrics over a 15 min interval. Subsequently, LOIC was employed from compromised hosts to launch successive HTTP, UDP, and TCP DDoS flooding attacks to emulate the typical DDoS flooding attack strategy (Figure 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, were processed using Konstanz Information Miner (KNIME) to extract the target metrics and eliminate duplicates to ensure a clean and structured dataset [39]. This dataset not only included normal and anomalous traffic patterns but also addressed limitations of existing SDN benchmarks by incorporating modern attack footprints and traffic variations reflective of the IoT era [30,39,71,84]. The resulting dataset formed the basis for ML-based ADC and SA, carried out in this study, to support a more robust behavioral analysis of SDN-reliant IoT networks and the mitigation of modern DDoS flooding attacks using the proposed DOE-GAN-SA framework. It should be noted that these simulation scenarios have also been adopted in our previous works [30,39,71]. The statistical distributions of the generated data for each of the SDN-reliant IoT network traffic scenarios are presented later on in Section 6.

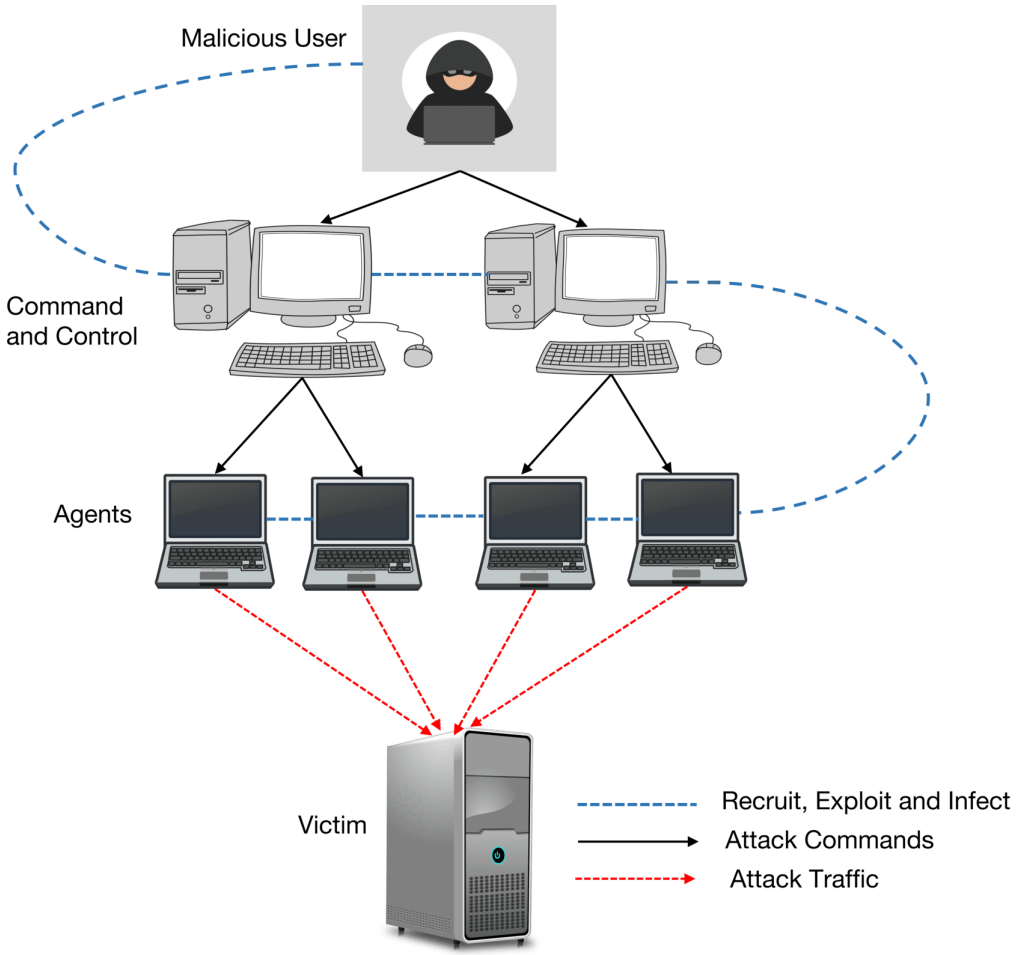


Figure 3. Typification of the DDoS attack strategy.

4. Basic Techniques

To support the development of the proposed DOE-GAN-SA framework, several fundamental techniques have been explored and adopted in this work. Even though these techniques are well known and established in the literature, detailing them in this work, with a strong focus on how they have been specifically applied, serves the primary purpose of ensuring that this work is self-contained and complete for understanding. Hence, these techniques are briefly discussed as follows.

4.1. Design of Experiments (DOE)

One popular strategy for building a surrogate model is to use a well-fitted DOE method to sample the design space and then approximate the computationally costly model data. In this way, any strategy that directs sample allocation in the design space to maximize the amount of information obtained is generally referred to as a DOE method [25,85]. To create the training data needed to construct the surrogate model, the computationally costly models are assessed at these sampled points [25]. An efficient DOE approach makes sure the samples are usually spread apart as much as possible to better capture global trends in the design space because of the inherent tradeoff between the number of sample points and the amount of information available from these points [25,85]. Classical factorial designs, Hammersley sampling, LHS, and Quasi Monte Carlo sampling are among the available DOE approaches [44,86].

Latin Hypercube Sampling (LHS)

For uniform sampling distributions, LHS is possibly the most popular DOE technique [44]. Numerous works in mathematics, engineering, and computational science have adopted LHS [25,46,87]. LHS has also been demonstrated to offer superior qualities that support effective filtering and significant variance reduction for numerous applications in comparison to other DOE approaches [25,87,88]. To assign n samples, LHS separates each sample's range into d bins, resulting in a total of dn bins in the design space for the n samples. Numerous LHS variations have been developed and/or proposed to minimize the possibility of non-uniform distributions [89], improve space filling [90], optimize projective properties [91], lessen spurious correlations [92], minimize least-square error, and maximize entropy [93]. As a statistical method, LHS samples input variables efficiently, ensuring that the entire range of each input parameter is represented. In the context of SDN-reliant IoT networks, LHS can be used to generate representative samples of network parameters, such as T_p, J_t, and R_t, indicative of bandwidth, latency, and packet loss, respectively, which are then used for SA, as typified in the proposed DOE-GAN-SA framework. By efficiently covering the input space, LHS reduces the number of simulations required to assess the effects of input parameters on the network's performance [94]. Since LHS is a well-known technique [44,95,96], its full algorithmic details are not provided herein. However, the essential steps in the implementation of LHS, in the context of this work, are described as follows:

- **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 the 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 given network scenario. LHS then works by generating k sample points, with each input parameter divided into k equally probable intervals. It should be noted that k is typically specified by the user, as discussed 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 es-

sential because it ensures that the samples are distributed uniformly across their respective ranges;

- **Step 3:** A value from each interval is randomly selected for each input parameter (X_i). This ensures that all the parts of the parameter space are represented. The resulting sample points for each parameter are then arranged in a Latin hypercube structure. The Latin hypercube structure allows for a more even and representative sampling in comparison to that of conventional random sampling, hence, the name LHS;
- **Step 4:** The sampled values for each parameter are shuffled to form k distinct sample points, with each sample point having one value from each parameter's range.

It should be noted that regardless of the value of k that the user specifies, the sample size affects the results of techniques like LHS, which seek to distribute samples uniformly across the range of feature values [96]. In this work, the sample space has been divided into bins, and new samples have been drawn at a predetermined bin fraction, 30 bins to be specific, as recommended by [95]. This procedure is implemented to ensure that LHS effectively covers the space of the SDN performance metrics across various scenarios within the SDN-reliant IoT network, thereby maximizing insight into parameter trends. The primary objective of this approach is to progressively expand the sample size while maintaining consistent marginal distributions of the SDN performance metrics [95].

4.2. Machine Learning (ML)

ML is a key component of contemporary IoT systems, providing data-driven methods for automated analysis and decision making [97,98]. ML usually provides systems with the ability to learn and enhance from experience automatically without being specifically programmed. To intelligently and robustly analyze data and develop the corresponding real-world data-driven paradigms, ML is essential. ML techniques can be generally categorized into four broad archetypes: supervised, unsupervised, semi-supervised, and reinforcement learning techniques [99]. Among these ML techniques, supervised learning techniques stand out for their abilities to model complex relationships between input features and desired outcomes using labeled data. This makes them particularly well suited for the proposed DOE-GAN-SA framework, which requires precise prediction, to support ADC and SA in the typified SDN-reliant IoT network considered in this work [99]. The main supervised learning techniques that underpin the proposed DOE-GAN-SA framework are neural networks (ANN and GAN to be precise) and the classification and regression tree (CART). These supervised learning techniques are discussed as follows.

4.2.1. Neural Networks

Neural networks (NNs) can be viewed as computational frameworks designed to simulate learning and data processing, inspired by the structure of the human brain, where interconnected nodes or neurons function collaboratively to process inputs and generate outputs [100]. Generally referred to as universal approximators, NNs play a crucial role in the characterization of complex systems, such as IoT networks, by enabling the analysis and optimization of system responses based on learned patterns and knowledge [101]. Their abilities to model and process information mirror biological neural systems, supporting advancements in network analysis and system optimization. Further exploration of NNs encompasses specific applications and architectures, including ANNs for structured data processing and GANs for generating and enhancing data representations, as discussed below.

Artificial Neural Networks (ANNs)

ANNs have been used to address several real-world problems, such as classification, prediction, optimization, and pattern recognition, including distinguishing traffic in intrusion detection and prevention systems (IDPSs) [102]. The efficiency and effectiveness of ANNs generally depend on data improvement, prior information, data representation, and feedback [103–106]. By adjusting the connection weights through training algorithms, like backpropagation, the performance of ANNs can be optimized to make them more suited for target applications [107]. Other algorithms exist for training neural networks, each differing in speed, precision, and memory requirements [108,109].

To develop a robust ANN model that fits the given data (see Section 3.3), any of the learning strategies can be merged or altered. In this work, the Levenberg–Marquardt training algorithm, widely regarded as one of the most efficient for ANNs [110], has been employed. The Levenberg–Marquardt training algorithm works by combining gradient descent and the Gauss–Newton method, addressing their limitations [110,111]. The primary drawback of the Levenberg–Marquardt training algorithm is its computational cost, stemming from the inversion of the Hessian matrix and storage requirements for the Jacobian matrix, which depend on the number of patterns, outputs, and weights [111]. For large-sized networks, this can become prohibitively expensive. However, considering our dataset of a few thousand instances or observations, having three inputs and one output, under two hundred fifty epochs on average, the Levenberg–Marquardt training algorithm offered a balance of speed, stable convergence, and manageable memory consumption for all the ANN models built, making it well suited for this work (see Section 6.2). Since ANNs are a well-known technique [30,100], their full algorithmic details are not provided herein. However, the essential steps in the ANN implementation in the proposed DOE-GAN-SA framework are described as follows:

- **Step 1:** Initialize the weights ($W^{(l)}$) and biases ($b^{(l)}$) randomly or using a predefined scheme for all the layers (l), where the activation or the neuron's output (A) is fed to the next layer;
- **Step 2:** For each layer (l), compute the weighted sum of the inputs as follows:

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

Apply an activation function (σ) to get

$$A^{(l)} = \sigma(Z^{(l)}) \tag{2}$$

- **Step 3:** Calculate the loss function ($\mathcal{L}$) to measure the error between the predicted output ($\hat{Y}$) and the actual target (Y) as follows:

$$\mathcal{L} = \frac{1}{m} \sum_{i=1}^{m} \ell(\hat{Y}^{(i)}, Y^{(i)}) \tag{3}$$

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

- **Step 4:** Compute the gradients of the loss for parameters using the chain rule as follows:

$$\frac{\partial \mathcal{L}}{\partial W^{(l)}} = \delta^{(l)} A^{(l-1)T}, \quad \frac{\partial \mathcal{L}}{\partial b^{(l)}} = \delta^{(l)} \tag{4}$$

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

$$\delta^{(l)} = \left(W^{(l+1)T} \delta^{(l+1)} \right) \odot \sigma'(Z^{(l)}) \tag{5}$$

- **Step 5:** Update the weights and biases using the Levenberg–Marquardt algorithm as follows:

$$W^{(l)} = W^{(l)} - \alpha \frac{\partial \mathcal{L}}{\partial W^{(l)}}, \quad b^{(l)} = b^{(l)} - \alpha \frac{\partial \mathcal{L}}{\partial b^{(l)}} \tag{6}$$

where α is the learning rate.

Generative Adversarial Networks (GANs)

GANs, introduced by [49], are a class of generative models that operate through an adversarial learning process. They consist of two NNs: a generator (G), which creates synthetic data intended to mimic real-world samples, and a discriminator (D), which evaluates the data and distinguishes between real and generated samples [49]. These two NNs are trained simultaneously in a dynamic competition, where the generator improves its ability to produce realistic data, while the discriminator becomes more adept at identifying differences. This adversarial interplay drives both neural networks to reach a state where the generator's outputs are virtually indistinguishable from real data, making GANs particularly effective for modeling complex and high-dimensional data distributions [49,70]. GANs are particularly relevant to this work because of their proven abilities to generate high-fidelity synthetic data that can capture intricate patterns and structures in various data types, including images, text, and time series [49,70]. More specifically, their adaptability makes them versatile tools for several applications requiring synthetic data [112,113].

Aside from GANs, several deep learning architectures have been explored for data generation, classification, and modeling. Convolutional neural networks (CNNs) excel at extracting visual features but are less effective for sequential network traffic [114]. Recurrent neural networks (RNNs) and their variants, such as long short-term memory (LSTM) and gated recurrent unit (GRU) networks, are better suited for time-series data [115]. However, GANs outperform them in generating high-quality synthetic data [116]. Due to their adversarial training process, GANs are highly effective in synthesizing realistic network traffic [49]. Unlike conventional methods for synthetic data generation, they also excel in scenarios involving challenging data landscapes, providing a robust mechanism for modeling and synthesizing realistic datasets [49,113]. These attributes justify their adoption for data augmentation in the proposed DOE-GAN-SA framework.

In the context of SDN-reliant IoT networks, GANs can be used to generate synthetic network traffic data that mimic the characteristics of real traffic. These data can then be used to explore how different input parameters affect the network's performance without the need for extensive and computationally expensive real-world simulations. The generated data can serve as input for SA to support a more scalable and cost-effective way to explore the parameter space [117]. This approach allows for a more flexible exploration of the network parameter space [118]. To do this, the generator ($G(z)$) takes a random noise vector (z), sampled from a noise distribution ($p_z(z)$), and generates synthetic data ($G(z)$) that resemble the real network traffic data. The discriminator ($D(x)$) then takes either real data (x) from the distribution ($p_{data}(x)$) or synthetic data ($G(z)$) and outputs the probability that the input is real data rather than synthetic data. Since GANs are a well-known technique [49,70,119], their full algorithmic details are not provided herein. However, the essential steps in the GAN architecture featured in the proposed DOE-GAN-SA framework are further (briefly) described as follows:

- **Step 1:** Initialize the parameters θ_g and θ_d of $G(z)$ and $D(x)$, respectively;

- **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 as follows:

$$\max_{\theta_d} \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \quad (7)$$

- **Step 3:** Update θ_g to minimize the ability of $D(x)$ to distinguish real from synthetic data as follows:

$$\min_{\theta_g} \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \quad (8)$$

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

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

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \quad (9)$$

4.2.2. Classification and Regression Tree (CART)

CART is a vital supervised learning technique employed within the proposed DOE-GAN-SA framework, mainly due to its robust and interpretable decision-making capabilities. Unlike other classification methods, CART is renowned for its simplicity, efficiency, and ability to handle both categorical and continuous data that are commonplace in complex SDN-reliant network scenarios [71]. Its relevance is further highlighted by its inclusion in the top 10 data-mining algorithms [120] and its demonstrated high level of accuracy in predictive tasks within SDN-reliant network scenarios [71]. CART constructs decision trees (DTs) by recursively partitioning the input feature space to maximize information gain at each split. This technique is non-parametric, meaning it makes no assumptions about the data distribution, which enhances its applicability across diverse datasets. For a set of input attributes ($X = \{x_1, x_2, \ldots, x_d\}$), the CART algorithm selects the optimal split at each node by evaluating the impurity reduction using a metric such as Gini's diversity index [121]. Then, the following essential steps are carried out for CART implementation in the proposed DOE-GAN-SA framework (since CART is a well-known technique [71,122] its full algorithmic details are not provided herein):

- **Step 1:** Determine the impurity of each node (I) by applying Gini's diversity index as follows:

$$gdi = 1 - \sum_{n=1}^{N} (p_n)^2 \quad (10)$$

  where the number of classes is N, and p_n is the proportion of observations in class n;
- **Step 2:** Evaluate all the possible splits and select the one that maximizes the impurity gain as follows:

$$I_G = p(T) - \Big(p(T_L) + p(T_R) \Big) \quad (11)$$

  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 the generalization.

It should be emphasized that ML techniques generally achieve superior performance after their hyperparameters have been properly optimized [71]. This is because the relia-

bility of ML models is inherently tied to the quality of the data and methodologies used during training, and spurious correlations that are interpretable by human analysts after evaluation are often inevitable [71]. In reality, a solution between high precision (positive predictive value (PPV)) and high recall (sensitivity or true positive rate (TPR)) is typically unavoidable, as achieving optimality in both metrics simultaneously is rarely feasible [123]. Hyperparameter tuning is one approach to optimize (i.e., maximize) either the precision or recall in the validation set (the set of observations used for hyperparameter optimization) [123]. However, this process falls outside the scope of the current study. Consequently, this paper does not investigate hyperparameter optimization. Instead, all ML the techniques employed within the DOE-GAN-SA framework utilize default hyperparameter settings from MATLAB R2023b (Matrix Laboratory) and the Scikit-learn library [124,125]. It is presumed that the default configurations in MATLAB and the Scikit-learn library are unlikely to be modified by the majority of network engineers, particularly those without substantial expertise in ML. The default values for a few of the critical hyperparameters are provided in Table 1.

Table 1. Table of hyperparameters.

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

5. The Proposed DOE-GAN-SA Framework

The flow diagram for the proposed DOE-GAN-SA framework is shown in Figure 4, and the principal steps in the framework are discussed as follows:

- **Step 1:** Collect the simulated metric 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 in the SDN-reliant IoT network can be described as follows:

$$IoT_{DB} = \{T_p,\ J_t,\ R_t\} \ \forall \text{ IoT Network Scenarios} \tag{12}$$

- **Step 2:** Use the collected data from Step 1 (i.e., IoT_{DB}) to implement LHS-based DOE (see Section 4.1) and GAN-based synthetic data generation (see Section "Generative Adversarial Networks (GANs)") concurrently and 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:

$$IoT_{LHS} = \{T_p^{LHS},\ J_t^{LHS},\ R_t^{LHS}\} \ \forall \text{ IoT Network Scenarios} \tag{13}$$

$$IoT_{GAN} = \{T_p^{GAN},\ J_t^{GAN},\ R_t^{GAN}\} \ \forall \text{ IoT Network Scenarios} \tag{14}$$

  In this Step, N is set at 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}, and R_t^{LHS} and T_p^{GAN}, J_t^{GAN}, and R_t^{GAN} in Equations (13) and (14), respectively) from Step 2 for each of the considered SDN-reliant IoT network scenarios (i.e., normal, HTTP DDoS flooding attack, TCP DDoS flooding attack, and UDP DDoS flooding attack operating

conditions) using 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 [39,71,126], and it can be described as follows:

$$Z_i = \frac{(X_i - \mu_i)}{\sigma_i} \; \forall \text{ IoT Network Scenarios} \tag{15}$$

where Z_i is the Z-score of the ith observation for any metric ($X_i \in \{IoT_{DB}, IoT_{LHS}$, and $IoT_{GAN}\}$) having a mean and a 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 scenarios 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:

$$IoT_{AUG} = \{T_p^{AUG}, J_t^{AUG}, R_t^{AUG}\} \; \forall \text{ IoT Network Scenarios} \tag{16}$$

- **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 [30,126,127], and it can be described as follows:

$$IoT_{AUG^i}^{norm} = \frac{IoT_{AUG^i} - IoT_{AUG^i}^{min}}{IoT_{AUG^i}^{max} - IoT_{AUG^i}^{min}} \; \forall \text{ IoT Network Scenarios} \tag{17}$$

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 the 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:

$$IoT_{AUG}^{norm} = \{T_{p_{norm}}^{AUG}, J_{t_{norm}}^{AUG}, R_{t_{norm}}^{AUG}\} \; \forall \text{ IoT Network Scenarios} \tag{18}$$

- **Step 6:** Generate response variables for every observation in IoT_{AUG}^{norm} by deriving a weighted cost function (C_F^w), as recommended in [30]. Specifically, $C_{F_i}^w$ for the ith observation in IoT_{AUG}^{norm} can be derived as follows:

$$C_{F_i}^w = [(abs(T_{p_{norm}}^{AUG_i} - J_{t_{norm}}^{AUG_i}) \times R_{t_{norm}}^{AUG_i})] \times w \tag{19}$$

where w is set at 1, 2, 3, and 4 when the SDN-reliant IoT network is operating normally, under a TCP DDoS flooding attack, under a UDP DDoS flooding attack, and under an HTTP DDoS flooding attack, respectively, as recommended in [30];

- **Step 7:** Build ANN models (see Section "Artificial Neural Networks (ANNs)") using IoT_{AUG}^{norm} as the explanatory variables and C_F^w as the corresponding response variables $\forall$ IoT network scenarios. Then, compute the MSE values for the respective ANN models and store them in MSE_{norm};

- **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 [30]. The new noisy datasets can be described mathematically as follows:

$$IoT_{AUG}^{T_p^{noisy}} = \{T_{p_{noisy}}^{AUG},\ J_{t_{norm}}^{AUG},\ R_{t_{norm}}^{AUG},\ C_F^w\}\ \forall \text{ IoT Network Scenarios} \tag{20}$$

$$IoT_{AUG}^{J_t^{noisy}} = \{T_{p_{norm}}^{AUG},\ J_{t_{noisy}}^{AUG},\ R_{t_{norm}}^{AUG},\ C_F^w\}\ \forall \text{ IoT Network Scenarios} \tag{21}$$

$$IoT_{AUG}^{R_t^{noisy}} = \{T_{p_{norm}}^{AUG},\ J_{t_{norm}}^{AUG},\ R_{t_{noisy}}^{AUG},\ C_F^w\}\ \forall \text{ IoT Network Scenarios} \tag{22}$$

- **Step 9:** Evaluate the same ANN models as in Step 7 using the datasets described in Equations (20), (21), and (22), respectively. Then, 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, the Wilcoxon test [128]) to ascertain and statistically validate the sensitivities of T_p, J_t, and R_t in [T_p^{AUG}, J_t^{AUG}, and R_t^{AUG}], respectively. The comparisons are mathematically described as follows:

$$SA_I \begin{cases} MSE_{norm} \text{ and } MSE_{noisy}^{T_p};\ \forall \text{ IoT Network Scenarios} \\ MSE_{norm} \text{ and } MSE_{noisy}^{J_t};\ \forall \text{ IoT Network Scenarios} \\ MSE_{norm} \text{ and } MSE_{noisy}^{R_t};\ \forall \text{ IoT Network Scenarios} \end{cases} \tag{23}$$

where SA_I denotes the SA inferences drawn from the comparisons in Equation (23).

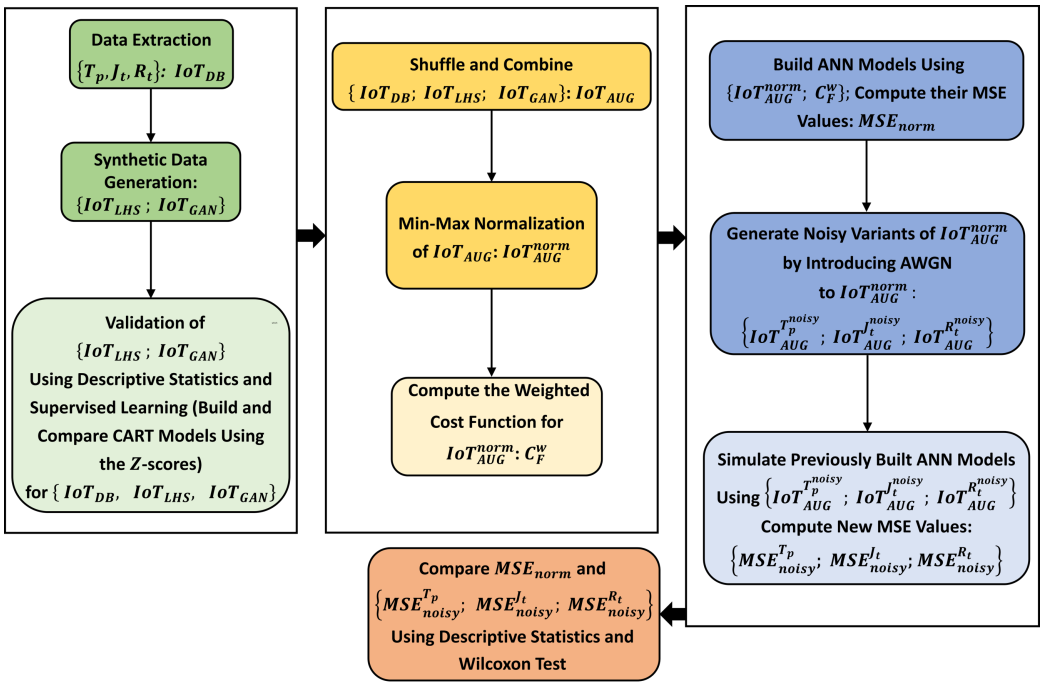


Figure 4. Flow diagram of the proposed DOE-GAN-SA framework.

In Step 3, the preferred method is standardization (Z-score normalization) over min–max normalization. This is because classifiers used to solve classification problems are assumed to have a distribution that is normal or close to it [129]. As discussed in Section 4.2.2, the classifier used to implement Step 3 is the CART method, due to its popularity and demonstrated ability to perform better than other popular classifiers in predicting SDN-

reliant network scenarios [71,120]. To assess the predictive accuracy of the fitted CART models in Step 3 and mitigate the risk of overfitting, a cross-validation approach has been employed. The selection of this validation method is informed by the dataset size, which comprises around 11,000 observations. In line with established practices for addressing classification challenges, a fivefold nested cross-validation framework has been uniformly applied across all the classifiers and experiments [71,130]. This approach aligns with widely recognized methodologies for enhancing model generalization and reliability [71,130].

In Steps 7 and 9, an ANN is preferred because of its robustness and popularity as a universal approximation function [131], 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 [30] for the ML-driven SA of SDN parameters. It should also be noted that the quantity of processing elements within each layer, as well as the overall number of layers, significantly impacts the training process of the ANN models built in Step 7. In other words, an insufficient number of processing elements may hinder the learning process, while an excessive number can result in overfitting to the training dataset [105,132]. For this study, the dataset was partitioned in accordance with methodologies suggested in prior research [30,133]. Specifically, 7560 data samples, representing 70% of the total dataset, were allocated for training. The remaining data were equally divided, with 1620 data samples (15%) designated for validation and an additional 1620 data samples (15%) reserved for testing purposes. This partitioning strategy ensures a balanced approach to model development and evaluation.

Unless otherwise noted, all the experiments carried out in this work to implement the DOE-GAN-SA framework outlined above were conducted on a workstation equipped with an Intel 6-core i7-8700 3.20 GHz CPU and 32.0 GB of RAM. All the independent experimental runs are repeated 50 times to achieve a sufficiently large sample size (50 in this instance) that allows a z-statistic to be utilized to determine the probability values for all the hypothesis tests performed [71,127]. For the synthetic data generated, i.e., IoT_{LHS} and IoT_{GAN}, the median of the 50 instances of generating them is employed to minimize the potential impact of extreme values and take into account the stochasticity of the LHS and GAN procedures [44,49].

6. Experimental Results and Discussion

Before employing IoT_{LHS} and IoT_{GAN} for the ML-driven SA carried out in this work, they are first validated to ascertain their representativeness of the modeled SDN-reliant IoT network with respect to the performance metrics and the network scenarios considered. The validation procedure and subsequent ML-driven SA implementation are detailed as follows, in line with the proposed DOE-GAN-SA framework (see Section 5).

6.1. Validation of Generated Synthetic Data

To carefully validate the generated synthetic datasets (IoT_{LHS} and IoT_{GAN}), a systematic comparison with the originally simulated dataset (IoT_{DB}) is conducted using both descriptive statistics and supervised learning techniques. The statistical characteristics of IoT_{DB}, IoT_{LHS}, and IoT_{GAN} are detailed in Tables 2–5, and their distributions are visually represented in Figures 5–8. A thorough analysis of these tables and figures reveals that both IoT_{LHS} and IoT_{GAN} successfully generate synthetic instances of T_p, J_t, and R_t (i.e., T_p^{LHS}, J_t^{LHS}, R_t^{LHS}, T_p^{GAN}, J_t^{GAN}, and R_t^{GAN}), which can be utilized to augment the original metrics for emulating real-time traffic scenarios within the modeled SDN-reliant IoT network. To ensure a robust comparison, the median values from 50 generated instances of IoT_{LHS} and IoT_{GAN} have been employed rather than individual generated instances for reasons stated earlier. Despite this methodological choice, both IoT_{LHS} and IoT_{GAN} exhibit a high degree of similarity to IoT_{DB} in terms of statistical attributes and overall trends. Furthermore,

across all the analyzed network scenarios, the descriptive statistics and distributions of IoT_{LHS}, IoT_{GAN}, and IoT_{DB} maintain strong alignment, as evidenced by the observations in Tables 2–5 and Figures 5–8.

Table 2. Descriptive statistics of all the metrics over 900 observations (normal network scenario).

Metric	Minimum	Maximum	Mean	Median	Standard Deviation
T_p	95.1000	95.9000	95.6332	95.6000	0.1402
$T_p{}^{LHS}$	95.5206	95.7434	95.6332	95.6338	0.0270
$T_p{}^{GAN}$	95.5960	95.7011	95.6253	95.6066	0.0347
J_t	0.0040	0.4930	0.2271	0.1940	0.0943
$J_t{}^{LHS}$	0.1528	0.2888	0.2264	0.2256	0.0212
$J_t{}^{GAN}$	0.1604	0.2715	0.1898	0.2237	0.0153
R_t	0.0320	2.1200	0.2114	0.1980	0.1286
$R_t{}^{LHS}$	0.1165	0.6057	0.2116	0.1918	0.0650
$R_t{}^{GAN}$	0.1252	0.2497	0.1905	0.1900	0.0189

Table 3. Descriptive statistics for all the metrics over 900 observations (TCP DDoS flooding attack network scenario).

Metric	Minimum	Maximum	Mean	Median	Standard Deviation
T_p	0.0000	95.9000	0.5441	0.0238	7.0999
$T_p{}^{LHS}$	0.0000	27.5431	0.5332	0.0150	3.0105
$T_p{}^{GAN}$	−0.0822	0.0884	0.0028	0.0019	0.0252
J_t	0.0040	0.4930	0.2271	0.1940	0.0943
$J_t{}^{LHS}$	0.1604	0.2878	0.2274	0.2264	0.0207
$J_t{}^{GAN}$	0.1579	0.2750	0.1963	0.1946	0.0164
R_t	0.2650	678.0000	302.2676	299.0000	110.6598
$R_t{}^{LHS}$	2.9487	413.3743	302.6604	308.0122	69.2561
$R_t{}^{GAN}$	275.1999	364.6439	304.1860	303.4058	11.935

Table 4. Descriptive statistics for all the metrics over 900 observations (UDP DDoS flooding attack network scenario).

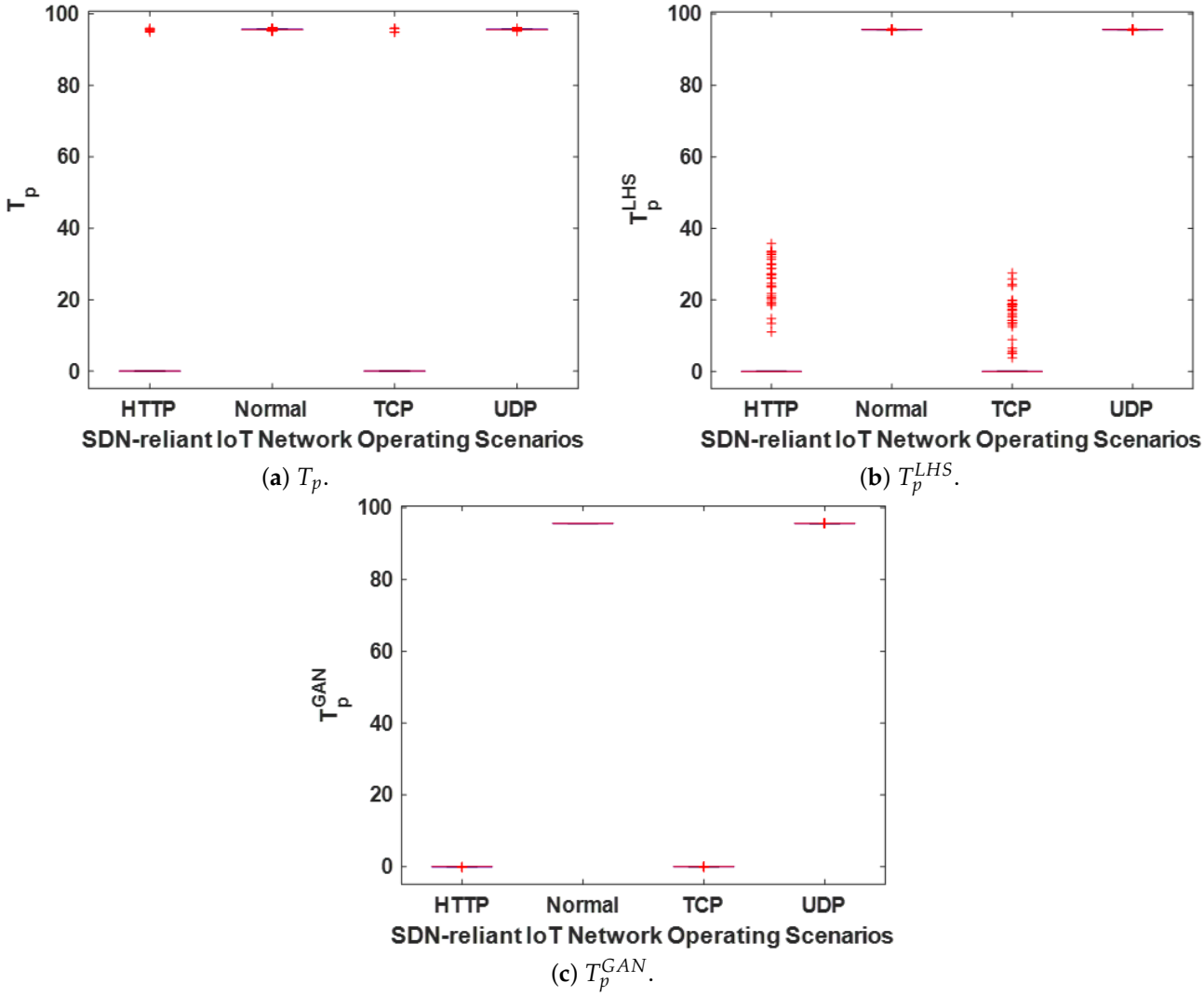
Metric	Minimum	Maximum	Mean	Median	Standard Deviation
T_p	95.1000	95.9000	95.6332	95.6000	0.1402
$T_p{}^{LHS}$	95.5138	95.7334	95.6331	95.6341	0.0258
$T_p{}^{GAN}$	95.5955	95.7007	95.6118	95.6033	0.0249
J_t	9.1610	18.4280	10.5100	10.1725	1.0496
$J_t{}^{LHS}$	9.9098	12.2078	10.5090	10.4302	0.3390
$J_t{}^{GAN}$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

Based on the results presented in Tables 2–5 and Figure 5, several conclusions can be drawn regarding the behaviors of T_p, T_p^{LHS}, and T_p^{GAN}: (1) A DDoS flooding attack on the SDN-reliant IoT network significantly affects or alters T_p, T_p^{LHS}, and T_p^{GAN}. (2) When the SDN is subjected to a UDP DDoS flooding attack, there is no substantial change observed in the distributions of T_p, T_p^{LHS}, and T_p^{GAN}. (3) In contrast, exposure to TCP and HTTP DDoS flooding attacks results in noticeable shifts in the distributions of T_p, T_p^{LHS}, and T_p^{GAN}, with the majority of the values clustered around 0. This differs from the typical range of approximately 95 to 96 observed under normal conditions or during UDP DDoS

flooding attacks. From a practical standpoint, the metrics T_p, T_p^{LHS}, and T_p^{GAN} (illustrated in Figure 5 and reported in Tables 2–5) appear to be more vulnerable to TCP and HTTP flooding attacks. This susceptibility is attributed to the operational characteristics of these attacks, which involve saturating the target server with excessive connection requests or numerous browser-based HTTP requests. These actions deplete network resources and ultimately lead to denial-of-service conditions for legitimate traffic [39,134,135].

Table 5. Descriptive statistics for all the metrics over 900 observations (HTTP DDoS flooding attack network scenario).

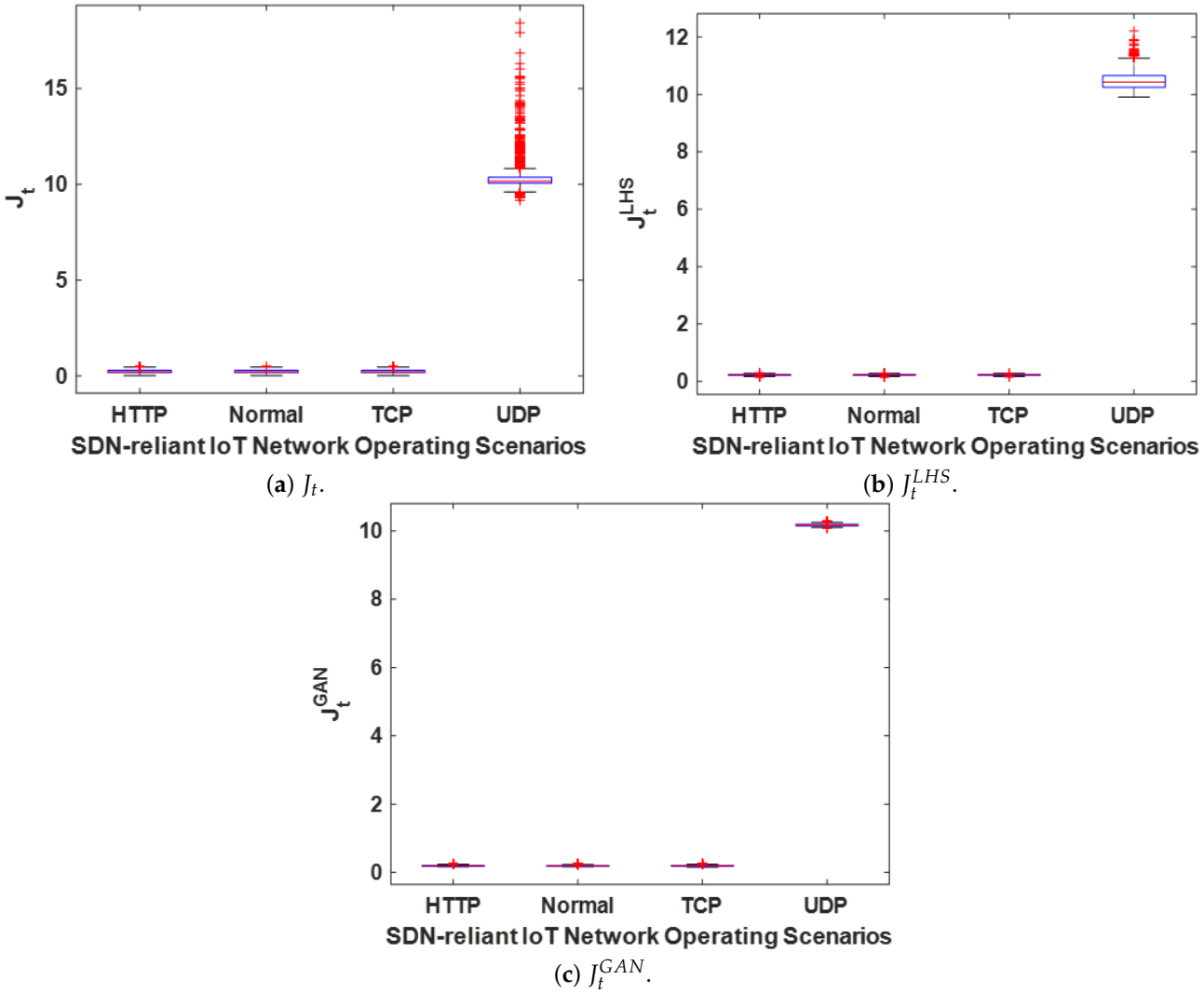
Metric	Minimum	Maximum	Mean	Median	Standard Deviation
T_p	0.0000	95.9000	0.7429	0.0000	8.3955
T_p^{LHS}	0.0000	35.7642	0.8434	2.685×10^{-5}	4.6874
T_p^{GAN}	−0.1042	0.0704	−0.0087	−0.0078	0.0283
J_t	0.0040	0.4930	0.2271	0.1940	0.0943
J_t^{LHS}	0.1617	0.3030	0.2259	0.2253	0.0214
J_t^{GAN}	0.1621	0.2572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



(**a**) T_p. (**b**) T_p^{LHS}. (**c**) T_p^{GAN}.

Figure 5. Box plots showing the distributions of the original and synthetic throughput metrics for all the scenarios in the modeled SDN-reliant IoT network.

Based on the results presented in Tables 2–5 and Figure 6, several conclusions can be drawn regarding the behaviors of J_t, J_t^{LHS}, and J_t^{GAN}: (1) A DDoS flooding attack on the SDN-reliant IoT network impacts or alters the values of J_t, J_t^{LHS}, and J_t^{GAN}. (2) The distributions of J_t, J_t^{LHS}, and J_t^{GAN} remain largely unchanged when the SDN is subjected to HTTP or TCP DDoS flooding attacks. (3) When exposed to a UDP DDoS flooding attack, the distributions of J_t, J_t^{LHS}, and J_t^{GAN} exhibit significant variation, with most values clustering around 10. In contrast, under normal operating conditions or during TCP and HTTP DDoS flooding attacks, the values are typically distributed within the range from approximately 0 to 0.3. J_t is highly influenced by the timing and sequence of packet arrivals. High J_t values occur when packets arrive out of order or in bursts separated by irregular gaps. From a practical standpoint, Figure 6 and Tables 2–5 indicate that J_t, J_t^{LHS}, and J_t^{GAN} are particularly susceptible to UDP DDoS flooding attacks. This vulnerability arises from the operational nature of such attacks, which do not require a handshake process to flood the target server with unsolicited UDP traffic, bypassing any initial consent from the server [39,136].

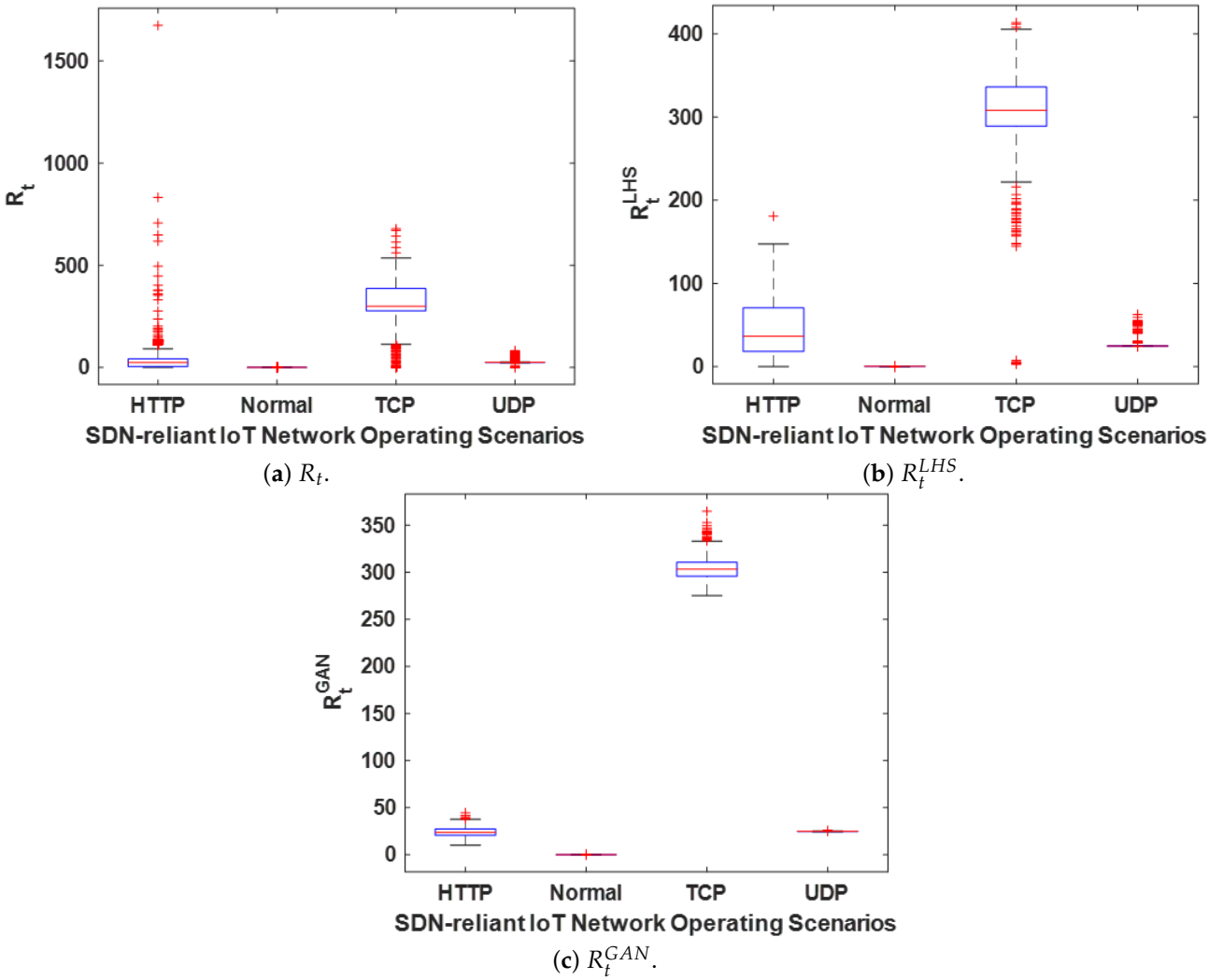


(**a**) J_t. (**b**) J_t^{LHS}. (**c**) J_t^{GAN}.

Figure 6. Box plots showing the distributions of the original and synthetic jitter metrics for all the scenarios in the modeled SDN-reliant IoT network.

Based on the results presented in Tables 2–5 and Figure 7, several conclusions can be drawn regarding the behaviors of R_t, R_t^{LHS}, and R_t^{GAN}: (1) A DDoS flooding attack on the SDN-reliant IoT network has measurable impacts on R_t, R_t^{LHS}, and R_t^{GAN}. (2) Under TCP, UDP, and HTTP DDoS flooding attacks, the distributions of R_t, R_t^{LHS}, and R_t^{GAN}

exhibit significant variation. (3) The majority of the values fall within the ranges from approximately 0 to 500 for TCP DDoS flooding attacks, approximately 10 to 50 for UDP DDoS flooding attacks, and approximately 0 to 200 for HTTP DDoS flooding attacks. These distributions differ markedly from the typical range of approximately 0 to 0.5 observed under normal operating conditions. From a practical standpoint, Figure 7 and Tables 2–5 indicate that R_t, R_t^{LHS}, and R_t^{GAN} are susceptible to all the investigated DDoS flooding attacks, owing to the inherent operational characteristics of these attacks, as previously discussed [39,134–136]. Consequently, R_t is a critical network-monitoring metric, with the potential to be significantly degraded by DDoS flooding attacks. This is particularly relevant in applications that rely on acknowledgment-based communication before the transmission of subsequent packets. Under such conditions, the unified communication systems within SDN-reliant IoT networks may experience substantial disruption.



(**a**) R_t. (**b**) R_t^{LHS}. (**c**) R_t^{GAN}.

Figure 7. Box plots showing the distributions of the original and synthetic response time metrics for all the scenarios in the modeled SDN-reliant IoT network.

A closer examination of Tables 2–5 highlights that the mean and median values for the corresponding metrics in IoT_{LHS}, IoT_{GAN}, and IoT_{DB} remain largely consistent (have minimal or acceptable deviations), with the exception of specific large deviations observed in the HTTP DDoS flooding attack scenario. These discrepancies can be attributed to the inherent distributional properties of T_p and R_t under the HTTP DDoS flooding attack, where noticeable outliers are present (see Figures 5 and 7). Similarly, Figures 5–8 demonstrate that under normal network conditions, the distributions of IoT_{LHS}, IoT_{GAN}, and IoT_{DB}

are in strong agreement, while in most DDoS flooding attack scenarios, their distributions exhibit reasonable conformity. Additionally, similar to IoT_{DB}, both IoT_{LHS} and IoT_{GAN} effectively distinguish between normal and DDoS flooding attack scenarios, as observed in Figures 5–8. This attribute is particularly advantageous for the implementation of ADC, which is further explored in Section 6.1.

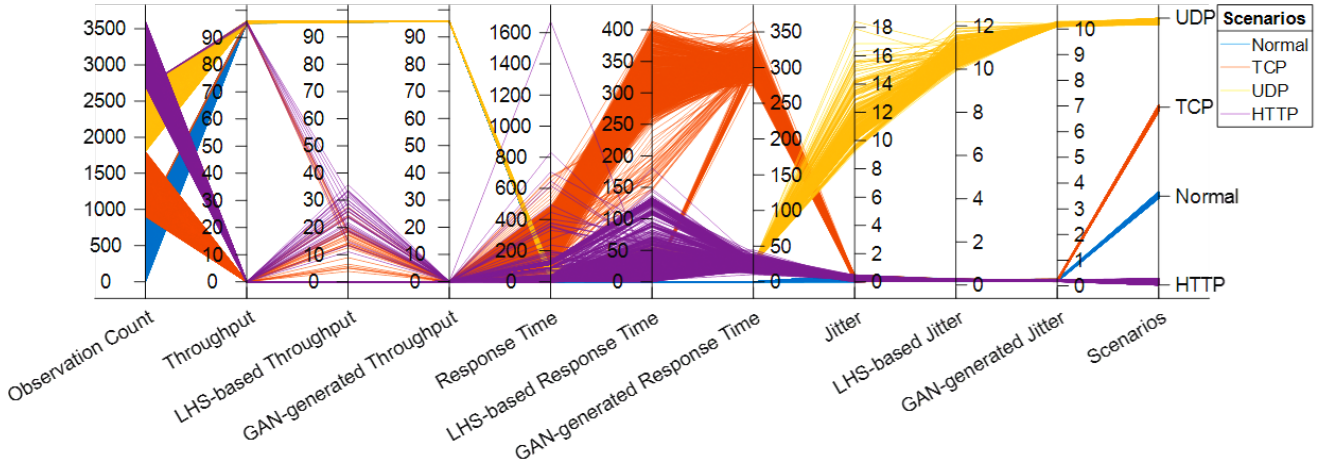


Figure 8. Parallel coordinates of the simulated, LHS-based, and GAN-generated T_p, R_t, and J_t metrics for all the network scenarios.

Comparisons and Hypothesis Testing for the Validation of Generated Datasets

In cases where discrepancies are observed between the generated datasets (IoT_{LHS} and IoT_{GAN}) and IoT_{DB}, it is important to consider the collective behavior of IoT_{LHS} and IoT_{GAN}. When combined, these synthetic datasets are expected to effectively capture the distributional patterns present in IoT_{DB} better, demonstrating their complementary nature and making them particularly valuable in augmenting IoT_{DB} to enhance its utility for the predictive modeling of diverse scenarios on SDN-reliant IoT networks. To verify this, supervised learning is employed to compare how well SDN-reliant IoT network scenarios can be classified considering IoT_{DB} and the combination of IoT_{LHS} and IoT_{GAN}. The supervised learning technique employed is CART for reasons stated earlier (see Section 4.2.2). The implementation framework and settings used have also been earlier provided in Section 4.2.2. To compare how IoT_{DB} and the combination of IoT_{LHS} and IoT_{GAN} perform for the predictive modeling of the SDN-reliant IoT network scenarios with a strong focus on ADC, 50 CART models are built using IoT_{DB} and the combination of IoT_{LHS} and IoT_{GAN}, respectively, and the descriptive statistics of the predictive accuracy (P_A) and $F1$-score are compared.

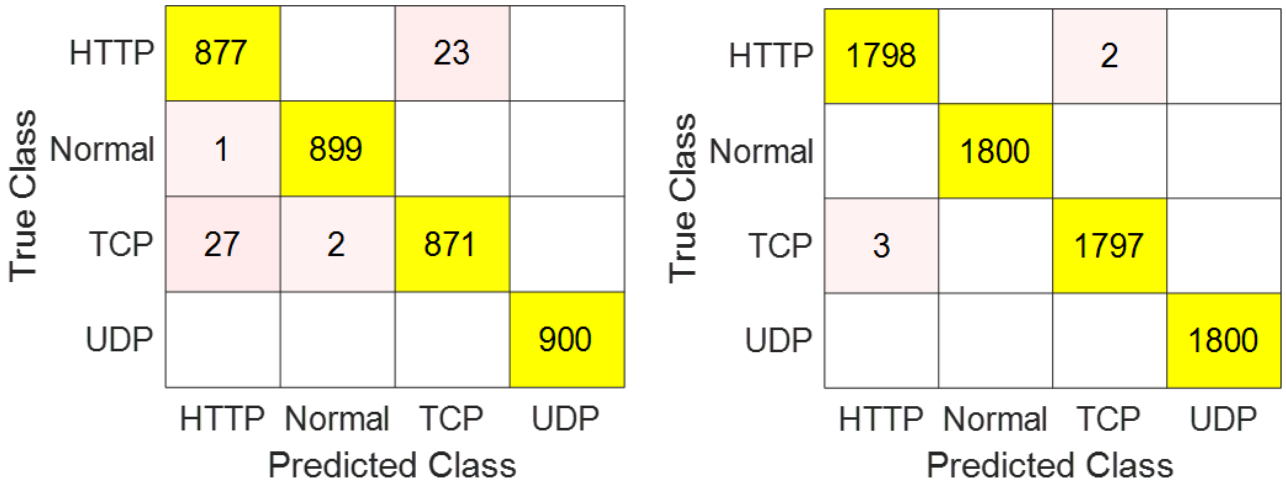
The P_A value of a classifier is generally defined as the ratio of correctly classified observations to the total number of observations classified. The P_A values of the CART models constructed using IoT_{DB} and the combination of IoT_{LHS} and IoT_{GAN} are deduced from each of their confusion matrices. Figure 9 displays the confusion matrices for typical CART models among the 50 models built using the two datasets (i.e., IoT_{DB} and the combination of IoT_{LHS} and IoT_{GAN}), respectively. From a given confusion matrix, the P_A value of the CART classifier can be evaluated as follows [71,123]:

$$P_A = \frac{TP + TN}{TP + TN + FP + FN} \tag{24}$$

where TP is a true positive (i.e., observations classified as O^* that are actually O^*), TN is a true negative (i.e., observations classified as not O^* and are not O^*), FP is a false

positive (i.e., observations classified as O^* but are not O^*), and FN is a false negative (i.e., observations classified as not O^* but are actually O^*).

The descriptive statistics of P_A over the 50 independent statistical runs, in which a CART model was constructed each time using IoT_{DB} and the combination of IoT_{LHS} and IoT_{GAN}, are reported in Table 6. Consistent with the observations from the typical confusion matrices shown in Figure 9, these results indicate that P_A is consistently high for both IoT_{DB} and the combination of IoT_{LHS} and IoT_{GAN}, suggesting that both datasets are suitable for CART-based ADC for the modeled SDN-reliant IoT network, even in the worst-case scenarios. Furthermore, the P_A value achieved using the combination of IoT_{LHS} and IoT_{GAN} is slightly higher than that obtained using IoT_{DB}, hinting that the synthetic dataset may possess features that enhance its ability to distinguish between the scenarios in the modeled SDN-reliant IoT network compared to IoT_{DB}, likely due to its larger number of observations. Additionally, the very low standard deviations observed for both datasets further confirm their suitability for CART-based ADC, considering the SDN-reliant IoT network scenarios investigated.



(**a**) IoT_{DB}. (**b**) Combination of IoT_{LHS} and IoT_{GAN}.

Figure 9. Typical confusion matrices for the CART models built.

Table 6. 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}

Since all the CART models provide confidence scores (i.e., posterior probabilities) for their predictions, their receiver-operating characteristic (ROC) curves and areas under the ROC curves (AUCs) can also be used to analyze the summaries of their individual P_A values [71,123]. Figure 10 shows the ROC curves for typical CART models among the 50 models built using the two datasets (i.e., IoT_{DB} and the combination of IoT_{LHS} and IoT_{GAN}), respectively. When the confidence scores of a classification model (the CART model in this case) are discretized, the sensitivity (also known as recall or true positive rate (TPR)) and the false positive rate (FPR) are combined to create the P_A summary for a specific ROC curve. The observations in the dataset are then classified using these discrete scores as prediction thresholds.

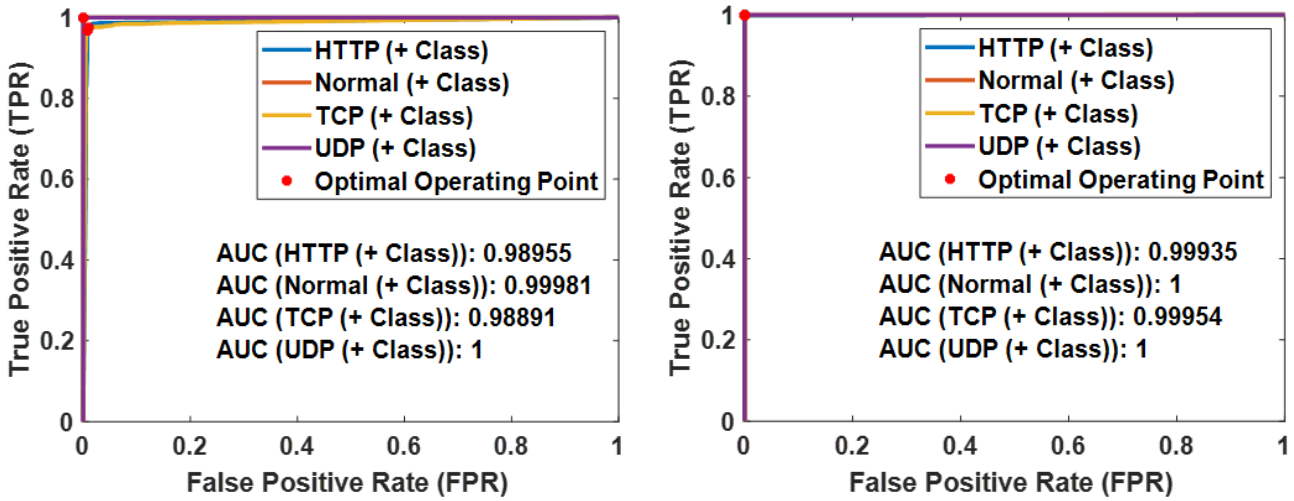
For a typical ROC curve, TPR, FPR, PPV, and the resulting F1-score can be estimated as follows [71,123]:

$$\text{TPR or Recall } = \frac{TP}{TP + FN} \tag{25}$$

$$\text{FPR } = \frac{FP}{FP + TN} \tag{26}$$

$$PPV = \frac{TP}{TP + FP} \tag{27}$$

$$F1 = 2 \times \frac{PPV \times TPR}{PPV + TPR} \tag{28}$$



(**a**) IoT_{DB}. (**b**) Combination of IoT_{LHS} and IoT_{GAN}.

Figure 10. Typical ROC curves for the CART models built.

The descriptive statistics of the F1-scores over the 50 independent runs, in which a CART model was constructed each time using IoT_{DB} and the combination of IoT_{LHS} and IoT_{GAN}, are reported in Table 7. Consistent with the observations from the typical ROC curves shown in Figure 10, these results indicate that the F1-scores are consistently high for both IoT_{DB} and the combination of IoT_{LHS} and IoT_{GAN}, suggesting that both datasets are suitable for CART-based ADC for the modeled SDN-reliant IoT network, even in the worst-case scenarios. Furthermore, the F1-score achieved using the combination of IoT_{LHS} and IoT_{GAN} is slightly higher than that obtained using IoT_{DB}, hinting that the synthetic dataset may possess features that enhance its ability to distinguish between the scenarios in the modeled SDN-reliant IoT network compared to IoT_{DB}, likely due to its larger number of observations. Additionally, the very low standard deviations observed for both datasets further confirm their suitability for CART-based ADC, considering the SDN-reliant IoT network scenarios investigated.

Table 7. Descriptive statistics of F1 over 50 independent runs.

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

To statistically verify that the CART models built using the combination of IoT_{LHS} and IoT_{GAN} are slightly more robust for ADC for the modeled SDN-reliant IoT network in comparison to the CART models built using IoT_{DB}, a hypothesis test (the Wilcoxon

test [128]) is carried out. To carry out the test, the results for the P_A and $F1$-scores obtained using the two types of datasets (i.e., IoT_{DB} and the combination of IoT_{LHS} and IoT_{GAN}) over 50 independent statistical runs are used as the data samples. In this instance, the null hypothesis is that the data samples of IoT_{DB} and the combination of IoT_{LHS} and IoT_{GAN} have equal medians at the 5% significance level (i.e., the 95% confidence level) against the alternative that they do not. The null hypothesis is rejected if there is substantial evidence against it, as indicated by the two-sided probability value (p-value) of the hypothesis test (i.e., the Wilcoxon rank-sum test) being less than or equal to 0.05.

From Table 8, it can be seen that the slight improvement in the P_A and F1-scores of the CART models built using the combination of IoT_{LHS} and IoT_{GAN} over the P_A and F1-scores of the CART models built using IoT_{DB} is statistically significant, with p-values of 6.0493×10^{-18} and 2.9193×10^{-22}, respectively. Overall, the findings from the validation procedure and comparisons suggest that synthetic data from the combination of IoT_{LHS} and IoT_{GAN} closely reflect the modeled SDN-reliant IoT network's characteristics, both under normal operating conditions and in the presence of DDoS flooding attacks. Particularly, their combined use presents a promising approach to enrich existing datasets and improve the fidelity of SDN-reliant IoT network simulations. In the next section, a robust ML-driven SA is carried out using a combination of the original simulated dataset (IoT_{DB}) and the validated synthetic data set (the combination of IoT_{LHS} and IoT_{GAN}), i.e., IoT_{AUG}.

Table 8. Hypothesis test: IoT_{DB} vs. the combination of IoT_{LHS} and IoT_{GAN}.

Metric	p-Value
P_A	6.0493×10^{-18}
F1	2.9193×10^{-22}

6.2. ML-Driven SA

The ML-driven SA component of the proposed DOE-GAN-SA framework is implemented as described in Section 5. Figure 11 illustrates the trend of C_F^w (see Equation (11)) across all the scenarios in the modeled SDN-reliant IoT network, considering IoT_{AUG}^{norm} (see Equation (18)). From Figure 11, it can be observed that using C_F^w as the response or target variable for the ANN models to be built enables, relatively, an additional layer of distinction between different scenarios in the modeled SDN-reliant IoT network. Specifically, C_F^w approaches or remains close to null when the network operates normally, while it exhibits higher values when the network is subjected to DDoS flooding attacks. The same structure, comprising three input nodes, ten hidden layers, and one output node, as shown in Figure 12, is used for all the ANN models built. Using IoT_{AUG}^{norm}, a total of 50 ANN models were built over 50 independent statistical runs. The training details, including the MSE trend, error histogram, gradient plot, and regression plots of a typical ANN model trained with IoT_{AUG}^{norm}, are presented in Figures 13, 14, 15, and 16, respectively.

From Figures 13 and 14, it can be observed that the training of the ANN models was not computationally expensive, as they typically converged to low MSE values in fewer than 250 epochs. The performances of the ANN models can also be considered as generally good, given their typically low error rates and high correlation coefficients according to Figures 15 and 16. As described in Section 5, to evaluate the sensitivity of the performance metrics of the modeled SDN-reliant IoT network (i.e., T_p, R_t, and J_t), the ANN models built using IoT_{AUG}^{norm} were re-simulated severally over 50 independent statistical runs using $IoT_{AUG}^{T_p^{noisy}}$, $IoT_{AUG}^{J_t^{noisy}}$, and $IoT_{AUG}^{R_t^{noisy}}$, respectively. The descriptive statistics of the MSE values for all the datasets used in simulating the ANN models are reported in Table 9.

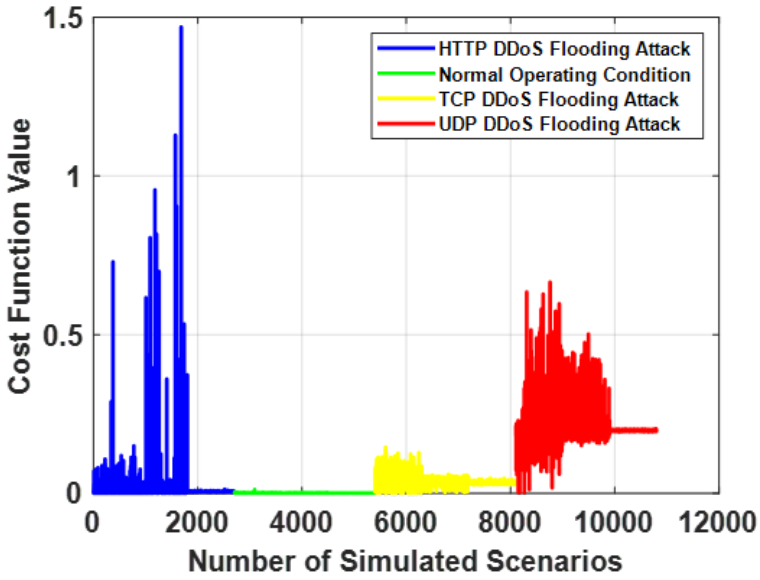


Figure 11. Plot of C_F^w against the IoT network scenarios.

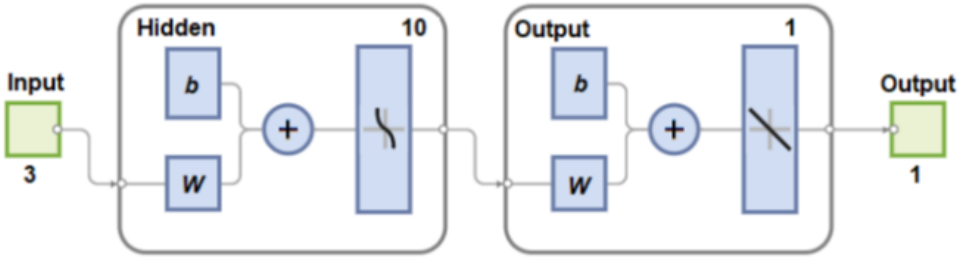


Figure 12. Layout of all the ANN models built.

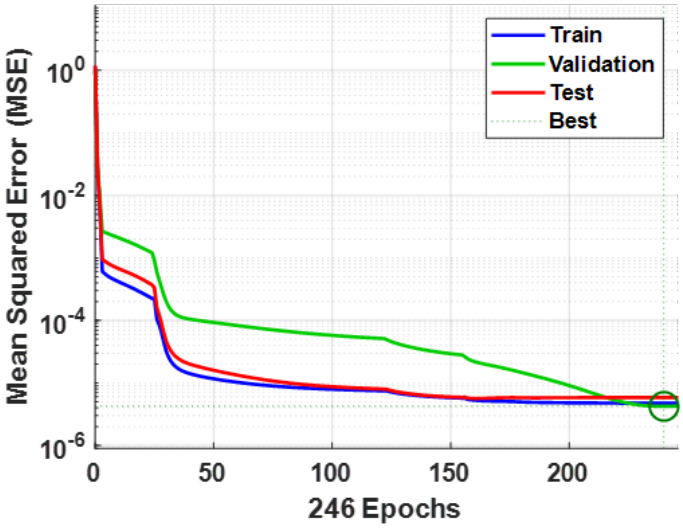


Figure 13. Typical convergence trend of the MSE values for the ANN models built using IoT_{AUG}^{norm} (the best validation performance is 4.2194×10^{-6} at epoch 240).

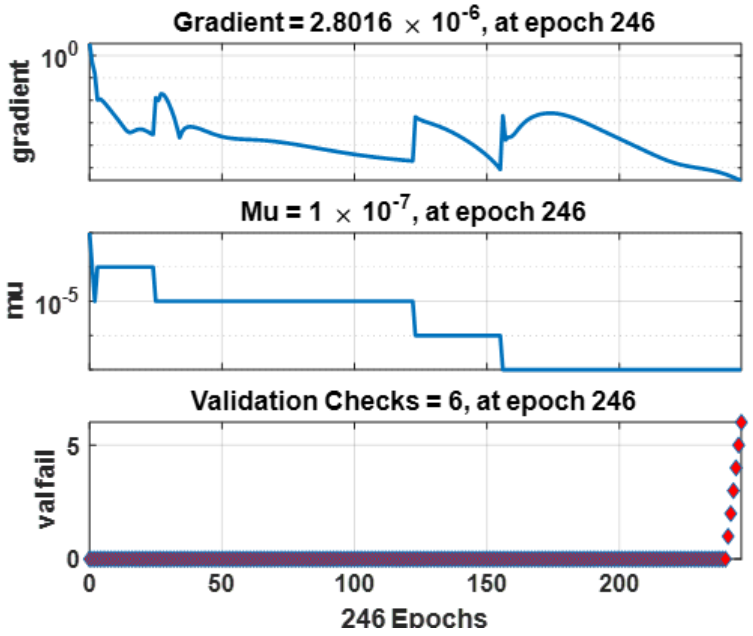


Figure 14. Typical gradient information trend for the ANN models built using IoT_{AUG}^{norm}.

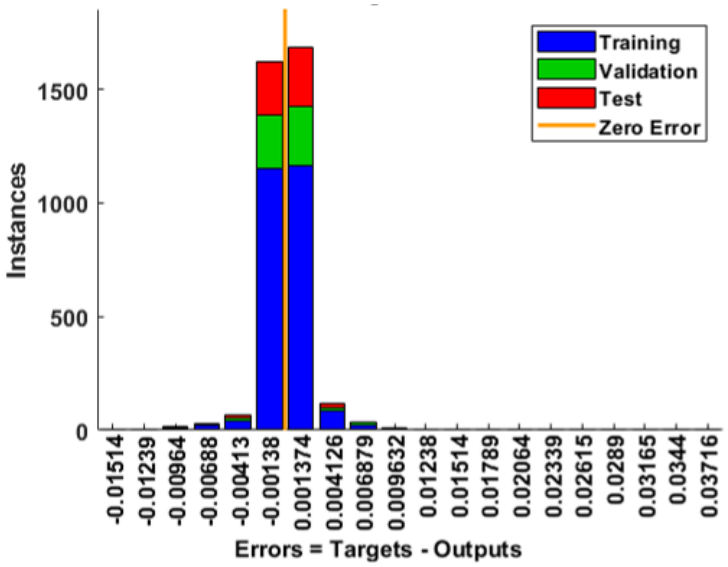


Figure 15. Typical error histogram (with 20 bins) for the ANN models built using IoT_{AUG}^{norm}.

Table 9. 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

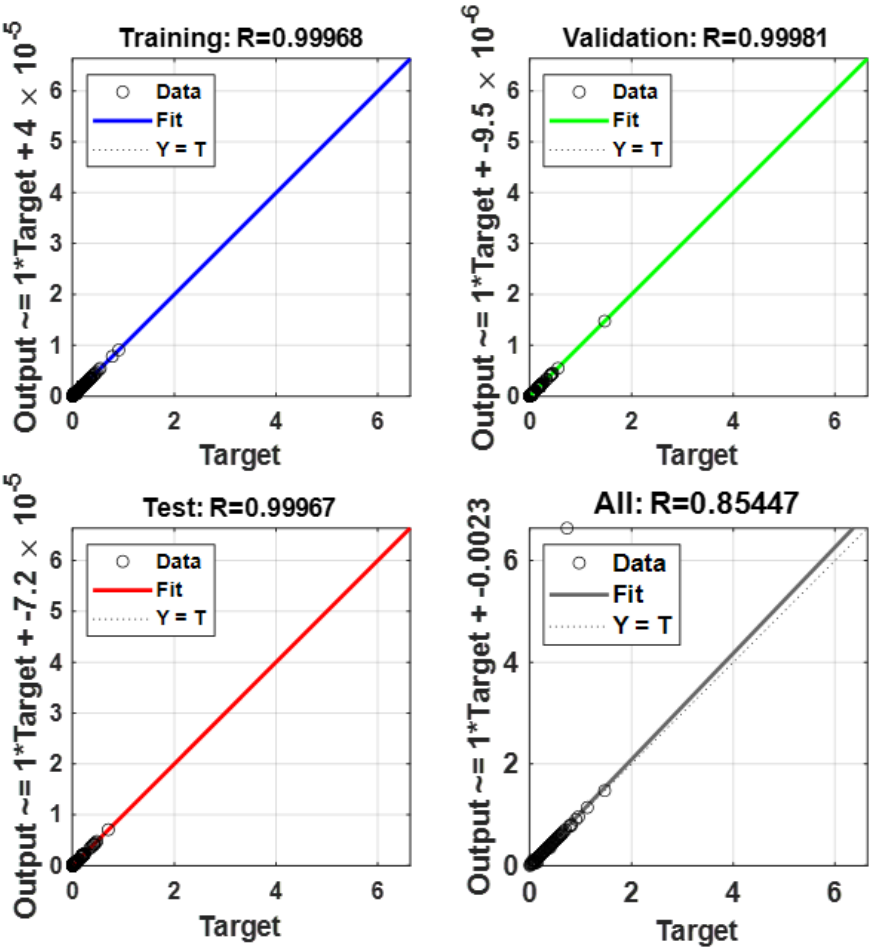


Figure 16. Typical regression plots for the ANN models built using IoT_{AUG}^{norm}.

From Table 9, the following observations can be made: (1) The MSE values of the ANN models are generally sensitive to changes in T_p, R_t, and J_t when comparing MSE_{norm} with $MSE_{noisy}^{T_p}$, $MSE_{noisy}^{R_t}$, and $MSE_{noisy}^{J_t}$, respectively. (2) On average, R_t appears to be 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 $MSE_{noisy}^{T_p}$. (3) In the best-case scenario, T_p induced the least change in MSE_{norm}, with a minimum MSE value of 0.0108 for $MSE_{noisy}^{T_p}$ (similar to R_t, with a minimum MSE value of 0.0109 for $MSE_{noisy}^{R_t}$), whereas J_t induced the most change, with a minimum MSE value of 0.0146 for $MSE_{noisy}^{J_t}$. (4) In the worst-case scenario, R_t caused the most disruption to MSE_{norm}, with a maximum MSE value of 22.3078 for $MSE_{noisy}^{R_t}$, while T_p caused the least disruption, with a maximum MSE value of 0.0302 for $MSE_{noisy}^{T_p}$. (5) In terms of the robustness and consistency, T_p appears to have induced the most consistent changes in MSE_{norm}, with a standard deviation of 0.0520 for $MSE_{noisy}^{T_p}$, whereas R_t induced the most inconsistent changes, with a standard deviation of 5.0983 for $MSE_{noisy}^{R_t}$. These observations, which indicate that T_p, R_t, and J_t are all sensitive, with R_t being more sensitive compared to T_p and J_t, are consistent with findings in the literature [39], thereby validating the approach.

Hypothesis Testing for Sensitivity Analysis

Similar to the hypothesis test reported earlier in Section 6.1, a hypothesis test (the Wilcoxon test [128]) is also conducted to statistically verify whether MSE_{norm} is impacted by changes in T_p, R_t, and J_t when their respective noisy variants (i.e., $IoT_{AUG}^{T_p^{noisy}}$, $IoT_{AUG}^{R_t^{noisy}}$,

and $IoT_{AUG}^{J_t^{noisy}}$) are used to evaluate the ANN models built using IoT_{AUG}^{norm}. To perform the test, the results for MSE_{norm}, $MSE_{noisy}^{T_p}$, $MSE_{noisy}^{R_t}$, and $MSE_{noisy}^{J_t}$, obtained over 50 independent statistical runs, are used as the data samples. The null hypothesis in this case is that the data samples of MSE_{norm} and those of $MSE_{noisy}^{T_p}$, $MSE_{noisy}^{R_t}$, and $MSE_{noisy}^{J_t}$ have equal medians at a 5% significance level (i.e., a 95% confidence level) against the alternative hypothesis that they do not. As previously mentioned, there is strong evidence against the null hypothesis if the two-sided probability value (p-value) of the hypothesis test (Wilcoxon rank-sum test) is less than or equal to 0.05, leading to the rejection of the hypothesis.

From Table 10, it can be observed that $MSE_{noisy}^{T_p}$, $MSE_{noisy}^{R_t}$, and $MSE_{noisy}^{J_t}$ are all statistically significantly different from MSE_{norm}, with a p-value of 7.0661×10^{-18}. This further reinforces the notion that T_p, R_t, and J_t are sensitive performance metrics for the modeled SDN-reliant IoT network.

Table 10. Hypothesis test: IoT_{AUG}^{norm} (MSE_{norm}) vs. other datasets.

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

6.3. Comparisons with Other Methods

Due to the scope of this study and time limitations, a comprehensive comparative analysis between the proposed DOE-GAN-SA framework and alternative approaches has not been conducted. However, considering the successful implementation of the DOE-GAN-SA framework demonstrated in this work, alongside the reported effectiveness of alternative methodologies in the literature [66,137–139], meaningful insights can be drawn from the comparisons presented in Table 11. As noted in Table 11, the proposed framework not only aligns with existing methods but also offers distinct advantages. Specifically, its purpose-built hybridization of DOE and ML techniques improves the applicability of hybrid approaches in understanding the behaviors of SDN-reliant IoT networks. Rather than serving as a direct replacement, the DOE-GAN-SA framework complements existing methods by addressing specific challenges and facilitating a more robust adoption of ML-based solutions.

Table 11. Comparative assessment.

Method	ADC	Data Augmentation	Efficiency	SA	Scalability	References
Monte Carlo	N/A	N/A	High	Yes	Yes	[137,139]
Variance-based	N/A	N/A	High	Yes	Yes	[66,138]
This work	Yes	Yes	High	Yes	Yes	N/A

6.4. Recommended Approach for Future Practical Implementation

Although the experiments carried out in this work have been primarily conducted within emulated network environments, the proposed DOE-GAN-SA framework is explicitly designed to generalize to real-world SDN-reliant IoT network scenarios. For a future practical implementation, a representative architectural configuration can be used to demonstrate how key network parameters, specifically, the throughput, jitter, and response time, can be dynamically monitored and incorporated into the DOE-GAN-SA framework

for advanced behavioral modeling and performance evaluation of SDN-reliant IoT networks. This approach aligns with practical implementations, notably, the study presented in [140], which describes a cost-effective IoT-SDN testbed constructed using Raspberry Pi devices configured as OpenFlow switches. The testbed integrates SDN controllers, such as Floodlight, to dynamically manage flow entries, mirroring the architecture employed in the DOE-GAN-SA framework. The emulated environment in DOE-GAN-SA utilizes a tree-based topology in Mininet and employs Floodlight to simulate SDN control operations with tools, such as *iperf* and LOIC, used to evaluate performance under DDoS flooding attack conditions. The study in [140] confirms the feasibility of real-time metric collection, including throughput, jitter, and bandwidth utilization, thereby reinforcing the applicability of SDN-based traffic control strategies in IoT systems.

The modular IoT-SDN testbed presented in [140] enables the collection of time-series data for key performance indicators (KPIs) under diverse network conditions, with specific emphasis on throughput and jitter metrics, i.e., T_p and J_t. The hierarchical network design implemented in the testbed adopts a tree-based topology, closely resembling the Fat-Tree topology used in DOE-GAN-SA, which is selected for its scalability and deterministic behavior in packet forwarding. These shared structural and analytical design elements underscore the alignment with contemporary engineering strategies aimed at enhancing the resilience and performance of SDN-reliant IoT networks. The study in [140] also identifies throughput as a critical metric reflecting the rate of successful data transmission across the network. Evaluations conducted on the SDN-based testbed demonstrate that centralized flow control improves bandwidth utilization in various traffic configurations. A parallel approach is adopted in the DOE-GAN-SA framework, wherein throughput is systematically recorded under both normal and adversarial (DDoS flooding attack) conditions. The degradation in throughput under attack scenarios serves as a sensitive indicator for performance deterioration and contributes to the SA.

Jitter, representing the variability in packet inter-arrival times, is another core metric evaluated in [140]. The testbed monitors jitter in real time to assess QoS levels in latency-sensitive applications, with low and consistent jitter values corresponding to stable network performance. A comparable methodology is employed in the DOE-GAN-SA framework, where jitter is analyzed across normal and attack conditions, further demonstrating the framework's robustness in detecting service anomalies. Both implementations rely on programmable, OpenFlow-enabled switches governed by a centralized SDN controller, allowing for adaptive routing and holistic monitoring. While the DOE-GAN-SA framework emphasizes data augmentation through LHS and GANs to enhance dataset diversity and model generalization, the testbed developed in [140] adopts an empirical strategy by collecting extensive, real-time data from live network interactions. The IoT-based SDN testbed in [140], provides a practical validation environment that mirrors the structural, functional, and analytical components of the DOE-GAN-SA framework. Both approaches highlight the significance of performance metrics, such as throughput and jitter, as primary indicators of network health. Although the DOE-GAN-SA approach introduces methodological innovations through the integration of DOE and ML techniques, the experimental insights from the referenced testbed establish a solid foundation for real-world applicability.

7. Conclusions

This study presented the DOE-GAN-SA framework, a novel methodology integrating DOE and ML techniques to enhance the behavioral analysis and characterization of SDN-reliant IoT networks. Specifically, it focused on data augmentation, ADC, and SA. Through the complementary use of LHS, GAN, CART, and ANN, DOE-GAN-SA enables a more comprehensive and systematic exploration of SDN-reliant IoT network behavior under

various operational conditions. The structured workflow of the DOE-GAN-SA framework begins with network scenario simulation, followed by LHS and GAN-driven synthetic data generation, statistical validation, CART-based ADC, and ANN-based SA. This hybridized approach enhanced conventional SA techniques by unifying DOE and ML techniques within a single framework, effectively implementing data augmentation, ADC, and SA. DOE-GAN-SA's reliability was further reinforced through the validation of synthetic data using descriptive statistics and supervised learning, confirming its suitability for the behavioral analysis of SDN-reliant IoT networks. Additionally, hypothesis testing was employed to statistically validate key experimental findings, ensuring the robustness of analytical outcomes. Empirical results demonstrated that DOE-GAN-SA improved ADC performance and enhanced SA for the SDN-reliant IoT network modeled in this study. Future work will focus on extending DOE-GAN-SA to assess the impacts of DDoS flooding attacks and other network anomalies, comparing its performance against those of alternative methods to further validate its effectiveness for SDN-reliant IoT networks.

Author Contributions: Conceptualization, C.E. and M.O.A.; methodology, C.E., M.O.A., A.O.S., W.S., U.E.U., I.O.-N. and F.T.A.; software, C.E. and M.O.A.; validation, M.O.A., A.O.S., W.S., U.E.U., I.O.-N. and F.T.A.; formal analysis, M.O.A.; investigation, C.E., M.O.A., A.O.S., W.S., U.E.U., I.O.-N. and F.T.A.; resources, C.E. and M.O.A.; data curation, C.E. and M.O.A.; writing—original draft preparation, C.E. and M.O.A.; writing—review and editing, C.E., M.O.A., W.S., A.O.S., U.E.U., I.O.-N. and F.T.A.; visualization, C.E. and M.O.A.; supervision, M.O.A., W.S. and A.O.S.; project administration, M.O.A., W.S. and A.O.S.; funding acquisition, M.O.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding. The APC for the publication of this work was fully supported by the editorial office.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: SDN Dataset for DDoS Flooding Attack Detection. Available at: Kaggle (https://doi.org/10.34740/KAGGLE/DSV/3965784, accessed on 07 October 2024).

Acknowledgments: The authors thank the editorial office for the invitation to submit this manuscript and the APC waiver, and they acknowledge the support of Lead City University, Nigeria, Wrexham University, U.K., University of Lincoln, U.K., Modibbo Adama University, Nigeria, and Regional Maritime University, Ghana.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Puthal, D.; Zhang, X. Secure Computing for the Internet of Things and Network Edges: Protecting Communication in the Worldwide Network of Devices. *IEEE Consum. Electron. Mag.* **2018**, *7*, 29–30. [CrossRef]
2. Alam, T. A Reliable Communication Framework and its Use in Internet of Things (IoT). *EngRN Commun. Syst. (Topic)* **2018**, *3*, 450–456.
3. Hakiri, A.; Gokhale, A.S.; Berthou, P.; Schmidt, D.C.; Gayraud, T. Software-Defined Networking: Challenges and research opportunities for Future Internet. *Comput. Netw.* **2014**, *75*, 453–471. [CrossRef]
4. Kreutz, D.; Ramos, F.; Veríssimo, P.; Esteve Rothenberg, C.; Azodolmolky, S.; Uhlig, S. Software-Defined Networking: A Comprehensive Survey. *Proc. IEEE* **2014**, *103*, 14–76. [CrossRef]
5. Liu, J.; Li, Y.; Chen, M.; Dong, W.; Jin, D. Software-defined internet of things for smart urban sensing. *IEEE Commun. Mag.* **2015**, *53*, 55–63. [CrossRef]
6. Gilani, S.M.M.; Usman, M.; Daud, S.; Kabir, A.; Nawaz, Q.; Judit, O. SDN-based multi-level framework for smart home services. *Multimed. Tools Appl.* **2024**, *83*, 327–347. [CrossRef]
7. Chaudhary, R.; Aujla, G.S.; Kumar, N.; Chouhan, P.K. A comprehensive survey on software-defined networking for smart communities. *Int. J. Commun. Syst.* **2025**, *38*, e5296. [CrossRef]

8. Qin, Z.; Denker, G.; Giannelli, C.; Bellavista, P.; Venkatasubramanian, N. A software defined networking architecture for the internet-of-things. In Proceedings of the 2014 IEEE Network Operations and Management Symposium (NOMS), Krakow, Poland, 5–9 May 2014; pp. 1–9.
9. Sarica, A.; Angin, P. Explainable security in SDN-based IoT networks. *Sensors* **2020**, *20*, 7326. [CrossRef]
10. Ezechi, C.; Akinsolu, M.O.; Sangodoyin, A.O.; Akinsolu, F.T.; Sakpere, W. Software-defined networking in cyber-physical systems. In *Cyber Physical System 2.0: Communication and Computational Technologies*; CRC Press: Boca Raton, FL, USA, 2024; p. 44.
11. Doorgakant, B.; Fowdur, T.P.; Akinsolu, M.O. End-to-End Power Models for 5G Radio Access Network Architectures with a Perspective on 6G. *Mathematics* **2025**, *13*, 466. [CrossRef]
12. Sangodoyin, A.; Sigwele, T.; Pillai, P.; Hu, Y.; Awan, I.; Pagna Diss, J. DoS Attack Impact Assessment on Software Defined Networks. In *Wireless and Satellite Systems*; Springer: Cham, Switzerland, 2018; pp. 11–22. [CrossRef]
13. Sezer, S.; Scott-Hayward, S.; Chouhan, P.; Fraser, B.; Lake, D.; Finnegan, J.; Viljoen, N.; Miller, M.; Rao, N. Are We Ready for SDN? Implementation Challenges for Software-Defined Networks. *Commun. Mag. IEEE* **2013**, *51*, 36–43. [CrossRef]
14. Kreutz, D.; Ramos, F.; Veríssimo, P. Towards secure and dependable software-defined networks. In Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, Hong Kong, China, 16 August 2013; pp. 55–60. [CrossRef]
15. Bhattacharyya, D.K.; Kalita, J. *DDoS Attacks: Evolution, Detection, Prevention, Reaction, and Tolerance*; CRC Press, Taylor & Francis Group: Boca Raton, FL, USA, 2016 ; pp. 1–283. [CrossRef]
16. Cao, Y.; Gao, Y.; Tan, R.; Han, Q.; Liu, Z. Understanding Internet DDoS Mitigation from Academic and Industrial Perspectives. *IEEE Access* **2018**, *6*, 66641–66648. [CrossRef]
17. Cirillo, M.; Di Mauro, M.; Matta, V.; Tambasco, M. Botnet Identification in DDoS Attacks With Multiple Emulation Dictionaries. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 3554–3569. [CrossRef]
18. Liao, B.; Ali, Y.; Nazir, S.; He, L.; Khan, H. Security Analysis of IoT Devices by Using Mobile Computing: A Systematic Literature Review. *IEEE Access* **2020**, *8*, 120331–120350. [CrossRef]
19. Qureshi, K.; Hussain, R.; Jeon, G. A Distributed Software Defined Networking Model to Improve the Scalability and Quality of Services for Flexible Green Energy Internet for Smart Grid Systems. *Comput. Electr. Eng.* **2020**, *84*, 106634. [CrossRef]
20. Trucano, T.G.; Swiler, L.P.; Igusa, T.; Oberkampf, W.L.; Pilch, M. Calibration, validation, and sensitivity analysis: What's what. *Reliab. Eng. Syst. Saf.* **2006**, *91*, 1331–1357. [CrossRef]
21. Iooss, B.; Lemaître, P. A review on global sensitivity analysis methods. In *Uncertainty Management in Simulation-Optimization of Complex Systems: Algorithms and Applications*; Springer: New York, NY, USA, 2015; pp. 101–122.
22. Ionescu-Bujor, M.; Cacuci, D.G. A comparative review of sensitivity and uncertainty analysis of large-scale systems—I: Deterministic methods. *Nucl. Sci. Eng.* **2004**, *147*, 189–203. [CrossRef]
23. Cacuci, D.G.; Ionescu-Bujor, M. A comparative review of sensitivity and uncertainty analysis of large-scale systems—II: Statistical methods. *Nucl. Sci. Eng.* **2004**, *147*, 204–217. [CrossRef]
24. Saltelli, A. *Global Sensitivity Analysis: The Primer*; John Wiley & Sons: Hoboken, NJ, USA, 2008; Volume 304, pp. 1–51. [CrossRef]
25. Santner, T.J.; Williams, B.J.; Notz, W.I.; Williams, B.J. *The Design and Analysis of Computer Experiments*; Springer: Berlin/Heidelberg, Germany, 2003; Volume 1, p. 284. [CrossRef]
26. Ezechi, C.; Akinsolu, M.O.; Sakpere, W.; Sangodoyin, A.O.; Akinsolu, F.T. Artificial Intelligence-Driven Sensitivity Analysis: Present-Day Approaches in Software-Defined Networking. In Proceedings of the 2024 5th International Conference on Emerging Trends in Electrical, Electronic and Communications Engineering (ELECOM), Balaclava, Mauritius, 20–22 November 2024; pp. 1–5. [CrossRef]
27. 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* **2023**, *16*, 5915. [CrossRef]
28. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
29. Ferrari, S.; Stengel, R. Smooth function approximation using neural networks. *IEEE Trans. Neural Netw.* **2005**, *16*, 24–38. [CrossRef]
30. Sangodoyin, A.O.; Akinsolu, M.O.; Awan, I. A deductive approach for the sensitivity analysis of software defined network parameters. *Simul. Model. Pract. Theory* **2020**, *103*, 102099. [CrossRef]
31. Williams, C.K.; Rasmussen, C.E. *Gaussian Processes for Machine Learning*; MIT Press: Cambridge, MA, USA, 2006; Volume 2, pp. 1–266.
32. Pandita, P.; Tsilifis, P.; Ghosh, S.; Wang, L. Scalable fully Bayesian Gaussian process modeling and calibration with adaptive sequential Monte Carlo for industrial applications. *J. Mech. Des.* **2021**, *143*, 074502. [CrossRef]
33. Tukan, M.; Zhou, S.; Maalouf, A.; Rus, D.; Braverman, V.; Feldman, D. Provable data subset selection for efficient neural networks training. In Proceedings of the International Conference on Machine Learning, PMLR, Honolulu, HI, USA, 23–29 July 2023; pp. 34533–34555.
34. Mehrizi, S.; Chatzinotas, S. Network Traffic Modeling and Prediction Using Graph Gaussian Processes. *IEEE Access* **2022**, *10*, 132644–132655. [CrossRef]

35. Stern, R.E.; Song, J.; Work, D.B. Accelerated Monte Carlo system reliability analysis through machine-learning-based surrogate models of network connectivity. *Reliab. Eng. Syst. Saf.* **2017**, *164*, 1–9. [CrossRef]
36. More, S.A.; Kachavimath, A.V. SDN Intrusion Detection using Meta-Heuristic Optimization and K-Nearest Neighbors Classifier. *Procedia Comput. Sci.* **2025**, *260*, 1137–1144. [CrossRef]
37. Glushkovsky, A. Designing Complex Experiments by Applying Unsupervised Machine Learning. *arXiv* **2021**, arXiv:2110.01458. [CrossRef]
38. Singh, D.; Ng, B.; Lai, Y.C.; Lin, Y.D.; Seah, W.K. Modelling software-defined networking: Software and hardware switches. *J. Netw. Comput. Appl.* **2018**, *122*, 24–36. [CrossRef]
39. Akinsolu, M.; Sangodoyin, A.; Uyoata, U. Behavioral Study of Software-Defined Network Parameters Using Exploratory Data Analysis and Regression-Based Sensitivity Analysis. *Mathematics* **2022**, *10*, 2536. [CrossRef]
40. White, G.; Nallur, V.; Clarke, S. Quality of service approaches in IoT: A systematic mapping. *J. Syst. Softw.* **2017**, *132*, 186–203. [CrossRef]
41. Montgomery, D.C. *Design and Analysis of Experiments*; John Wiley & Sons: Hoboken, NJ, USA, 2017; p. 734.
42. Singh, S.P.; Singh, P.; Diwakar, M.; Kumar, P. Improving quality of service for Internet of Things (IoT) in real life application: A novel adaptation based Hybrid Evolutionary Algorithm. *Internet Things* **2024**, *27*, 101323. [CrossRef]
43. Ali, Y.; Khan, H.U.; Khalid, M. Engineering the advances of the artificial neural networks (ANNs) for the security requirements of Internet of Things: A systematic review. *J. Big Data* **2023**, *10*, 128. [CrossRef]
44. Stein, M. Large sample properties of simulations using Latin hypercube sampling. *Technometrics* **1987**, *29*, 143–151. [CrossRef]
45. Garud, S.S.; Karimi, I.A.; Kraft, M. Design of computer experiments: A review. *Comput. Chem. Eng.* **2017**, *106*, 71–95. [CrossRef]
46. Akinsolu, M.O.; Liu, B.; Grout, V.; Lazaridis, P.I.; Mognaschi, M.E.; Di Barba, P. A parallel surrogate model assisted evolutionary algorithm for electromagnetic design optimization. *IEEE Trans. Emerg. Top. Comput. Intell.* **2019**, *3*, 93–105. [CrossRef]
47. Borisut, P.; Nuchitprasittichai, A. Adaptive Latin Hypercube Sampling for a Surrogate-Based Optimization with Artificial Neural Network. *Processes* **2023**, *11*, 3232. [CrossRef]
48. Ramzan, F.; Sartori, C.; Consoli, S.; Reforgiato Recupero, D. Generative Adversarial Networks for Synthetic Data Generation in Finance: Evaluating Statistical Similarities and Quality Assessment. *AI* **2024**, *5*, 667–685. [CrossRef]
49. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.C.; Bengio, Y. Generative Adversarial Nets. In Proceedings of the Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; Volume 3, pp. 1–9.
50. Wang, K.; Gou, C.; Duan, Y.; Lin, Y.; Zheng, X.; Wang, F.Y. Generative adversarial networks: Introduction and outlook. *IEEE/CAA J. Autom. Sin.* **2017**, *4*, 588–598. [CrossRef]
51. Mohebbi Moghaddam, M.; Boroomand, B.; Jalali, M.; Zareian, A.; Daeijavad, A.; Manshaei, M.H.; Krunz, M. Games of GANs: Game-theoretical models for generative adversarial networks. *Artif. Intell. Rev.* **2023**, *56*, 9771–9807. [CrossRef]
52. Sabuhi, M.; Zhou, M.; Bezemer, C.P.; Musilek, P. Applications of generative adversarial networks in anomaly detection: A systematic literature review. *IEEE Access* **2021**, *9*, 161003–161029. [CrossRef]
53. Navidan, H.; Moshiri, P.F.; Nabati, M.; Shahbazian, R.; Ghorashi, S.A.; Shah-Mansouri, V.; Windridge, D. Generative Adversarial Networks (GANs) in networking: A comprehensive survey & evaluation. *Comput. Netw.* **2021**, *194*, 108149.
54. Qiu, T.; Yang, X.; Chen, N.; Zhang, S.; Min, G.; Wu, D.O. A Self-Adaptive Robustness Optimization Method With Evolutionary Multi-Agent for IoT Topology. *IEEE/ACM Trans. Netw.* **2024**, *32*, 1346–1361. [CrossRef]
55. Shin, S.Y.; Nejati, S.; Sabetzadeh, M.; Briand, L.C.; Arora, C.; Zimmer, F. 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, Seoul, Republic of Korea, 29 June–3 July 2020; pp. 137–148.
56. Bizanis, N.; Kuipers, F.A. SDN and virtualization solutions for the Internet of Things: A survey. *IEEE Access* **2016**, *4*, 5591–5606. [CrossRef]
57. Biswas, A.; Md Abdullah Al, N.; Imran, A.; Sejuty, A.T.; Fairooz, F.; Puppala, S.; Talukder, S. Generative adversarial networks for data augmentation. In *Data Driven Approaches on Medical Imaging*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 159–177.
58. Ring, M.; Schlör, D.; Landes, D.; Hotho, A. Flow-based network traffic generation using generative adversarial networks. *Comput. Secur.* **2019**, *82*, 156–172. [CrossRef]
59. Shukla, R.S.; Alatawi, A.M. Enhancing Internet Traffic Forecasting in MEC Environments With 5GT-Trans: Leveraging Synthetic Data and Transformer-Based Models. *IEEE Access* **2025**, *13*, 83607–83618. [CrossRef]
60. Vaz, B.; Figueira, Á. GANs in the Panorama of Synthetic Data Generation Methods. *ACM Trans. Multimed. Comput. Commun. Appl.* **2024**, *21*, 1–28. [CrossRef]
61. Fox, J.; Ökten, G.; Uzunoğlu, B. 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), Rome, Italy, 20–22 November 2019; pp. 446–451. [CrossRef]

62. Zhang, J. Modern Monte Carlo methods for efficient uncertainty quantification and propagation: A survey. *Wiley Interdiscip. Rev. Comput. Stat.* **2021**, *13*, e1539. [CrossRef]
63. Ma, Z.; Xiao, M.; Xiao, Y.; Pang, Z.; Poor, H.V.; Vucetic, B. High-Reliability and Low-Latency Wireless Communication for Internet of Things: Challenges, Fundamentals, and Enabling Technologies. *IEEE Internet Things J.* **2019**, *6*, 7946–7970. [CrossRef]
64. Sobol, I.M. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Math. Comput. Simul.* **2001**, *55*, 271–280. [CrossRef]
65. Zadeh, F.K.; Nossent, J.; Sarrazin, F.; Pianosi, F.; Van Griensven, A.; Wagener, T.; Bauwens, W. Comparison of variance-based and moment-independent global sensitivity analysis approaches by application to the SWAT model. *Environ. Model. Softw.* **2017**, *91*, 210–222. [CrossRef]
66. Piano, S.L.; Ferretti, F.; Puy, A.; Albrecht, D.; Saltelli, A. Variance-based sensitivity analysis: The quest for better estimators and designs between explorativity and economy. *Reliab. Eng. Syst. Saf.* **2021**, *206*, 107300. [CrossRef]
67. Phromphan, P.; Suvisuthikasame, J.; Kaewmongkol, M.; Chanpichitwanich, W.; Sleesongsom, S. A New Latin Hypercube Sampling with Maximum Diversity Factor for Reliability-Based Design Optimization of HLM. *Symmetry* **2024**, *16*, 901. [CrossRef]
68. Pandey, C.; Tiwari, V.; Rodrigues, J.J.P.C.; Roy, D.S. 5GT-GAN-NET: Internet Traffic Data Forecasting With Supervised Loss Based Synthetic Data Over 5G. *IEEE Trans. Mob. Comput.* **2024**, *23*, 10694–10705. [CrossRef]
69. Goyal, M.; Mahmoud, Q.H. A Systematic Review of Synthetic Data Generation Techniques Using Generative AI. *Electronics* **2024**, *13*, 3509. [CrossRef]
70. Figueira, A.; Vaz, B. Survey on synthetic data generation, evaluation methods and GANs. *Mathematics* **2022**, *10*, 2733. [CrossRef]
71. Sangodoyin, A.O.; Akinsolu, M.O.; Pillai, P.; Grout, V. Detection and Classification of DDoS Flooding Attacks on Software-Defined Networks: A Case Study for the Application of Machine Learning. *IEEE Access* **2021**, *9*, 122495–122508. [CrossRef]
72. 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, and security of software defined networking: Challenges and open issues. *Internet Things* **2020**, *12*, 100289. [CrossRef]
73. Jain, A.K.; Shukla, H.; Goel, D. A comprehensive survey on DDoS detection, mitigation, and defense strategies in software-defined networks. *Cluster Computing* **2024**, *27*, 13129–13164. [CrossRef]
74. Palattella, M.R.; Dohler, M.; Grieco, A.; Rizzo, G.; Torsner, J.; Engel, T.; Ladid, L. Internet of Things in the 5G Era: Enablers, Architecture, and Business Models. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 510–527. [CrossRef]
75. Nurlan, Z.; Zhukabayeva, T.; Othman, M.; Adamova, A.; Zhakiyev, N. Wireless sensor network as a mesh: Vision and challenges. *IEEE Access* **2021**, *10*, 46–67. [CrossRef]
76. Neilson, B.; Rossiter, N. Automating Labour and the Spatial Politics of Data Centre Technologies. In *Topologies of Digital Work: How Digitalisation and Virtualisation Shape Working Spaces and Places*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 77–101.
77. Escudero-Sahuquillo, J.; Garcia, P.J.; Quiles, F.J.; Reinemo, S.A.; Skeie, T.; Lysne, O.; Duato, J. A new proposal to deal with congestion in InfiniBand-based fat-trees. *J. Parallel Distrib. Comput.* **2014**, *74*, 1802–1819. [CrossRef]
78. Lu, P.J.; Lai, M.C.; Chang, J.S. A survey of high-performance interconnection networks in high-performance computer systems. *Electronics* **2022**, *11*, 1369. [CrossRef]
79. Lombardi, M.; Pascale, F.; Santaniello, D. Internet of things: A general overview between architectures, protocols and applications. *Information* **2021**, *12*, 87. [CrossRef]
80. Bittencourt, L.; Immich, R.; Sakellariou, R.; Fonseca, N.; Madeira, E.; Curado, M.; Villas, L.; DaSilva, L.; Lee, C.; Rana, O. The internet of things, fog and cloud continuum: Integration and challenges. *Internet Things* **2018**, *3*, 134–155. [CrossRef]
81. Burhan, M.; Rehman, R.A.; Khan, B.; Kim, B.S. IoT elements, layered architectures and security issues: A comprehensive survey. *Sensors* **2018**, *18*, 2796. [CrossRef]
82. Dos Reis Fontes, R.; Campolo, C.; Esteve Rothenberg, C.; Molinaro, A. From Theory to Experimental Evaluation: Resource Management in Software-Defined Vehicular Networks. *IEEE Access* **2017**, *5*, 3069–3076. [CrossRef]
83. Muelas, D.; Ramos, J.; Vergara, J.E.L.d. Assessing the Limits of Mininet-Based Environments for Network Experimentation. *IEEE Netw.* **2018**, *32*, 168–176. [CrossRef]
84. Sommer, R.; Paxson, V. Outside the closed world: On using machine learning for network intrusion detection. In Proceedings of the 2010 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 16–19 May 2010; pp. 305–316.
85. Koziel, S.; Yang, X.S. *Computational Optimization, Methods and Algorithms*; Springer: Berlin/Heidelberg, Germany, 2011; Volume 356. [CrossRef]
86. Giunta, A.; Wojtkiewicz, S.; Eldred, M. Overview of modern design of experiments methods for computational simulations. In Proceedings of the 41st Aerospace Sciences Meeting and Exhibit, Reno, NV, USA, 6–9 January 2003; p. 649.
87. Shields, M.D.; Zhang, J. The generalization of Latin hypercube sampling. *Reliab. Eng. Syst. Saf.* **2016**, *148*, 96–108. [CrossRef]
88. Akinsolu, M. Efficient Surrogate Model-Assisted Evolutionary Algorithm for Electromagnetic Design Automation with Applications. Ph.D. Thesis, University of Chester, Chester, UK, 2019. Available online: https://chesterrep.openrepository.com/handle/10034/623568 (accessed on 6 January 2025).

89. Palmer, K.; Tsui, K.L. A minimum bias Latin hypercube design. *Iie Trans.* **2001**, *33*, 793–808. [CrossRef]
90. Dalbey, K.; Karystinos, G. Fast generation of space-filling latin hypercube sample designs. In Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference, Ft. Worth, TX, USA, 13–15 September 2010; p. 9085. [CrossRef]
91. Liefvendahl, M.; Stocki, R. A study on algorithms for optimization of Latin hypercubes. *J. Stat. Plan. Inference* **2006**, *136*, 3231–3247. [CrossRef]
92. Vořechovskỳ, M.; Novák, D. Correlation control in small-sample Monte Carlo type simulations I: A simulated annealing approach. *Probabilistic Eng. Mech.* **2009**, *24*, 452–462. [CrossRef]
93. Park, J.S. Optimal Latin-hypercube designs for computer experiments. *J. Stat. Plan. Inference* **1994**, *39*, 95–111. [CrossRef]
94. Helton, J.C.; Davis, F.J. Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. *Reliab. Eng. Syst. Saf.* **2003**, *81*, 23–69. [CrossRef]
95. Pleming, J.; Manteufel, R. Replicated Latin hypercube sampling. In Proceedings of the 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Austin, TX, USA, 18–21 April 2005; p. 1819.
96. Saurette, D.D.; Biswas, A.; Heck, R.J.; Gillespie, A.W.; Berg, A.A. Determining minimum sample size for the conditioned Latin hypercube sampling algorithm. *Pedosphere* **2024**, *34*, 530–539. [CrossRef]
97. Sarker, I.H.; Furhad, M.H.; Nowrozy, R. Ai-driven cybersecurity: An overview, security intelligence modeling and research directions. *SN Comput. Sci.* **2021**, *2*, 173. [CrossRef]
98. Sarker, I.H. Machine learning: Algorithms, real-world applications and research directions. *SN Comput. Sci.* **2021**, *2*, 160. [CrossRef]
99. Mohammed, M.; Khan, M.B.; Bashier, E.B.M. *Machine Learning: Algorithms and Applications*; CRC Press: Boca Raton, FL, USA, 2016; Volume 1, p. 226. [CrossRef]
100. Chen, M.; Challita, U.; Saad, W.; Yin, C.; Debbah, M. Artificial neural networks-based machine learning for wireless networks: A tutorial. *IEEE Commun. Surv. Tutorials* **2019**, *21*, 3039–3071. [CrossRef]
101. van Gerven, M.; Bohte, S. Artificial Neural Networks as Models of Neural Information Processing. *Front. Comput. Neurosci.* **2017**, *11*, 114. [CrossRef] [PubMed]
102. Hodo, E.; Bellekens, X.; Hamilton, A.; Dubouilh, P.L.; Iorkyase, E.; Tachtatzis, C.; Atkinson, R. Threat analysis of IoT networks using artificial neural network intrusion detection system. In Proceedings of the 2016 International Symposium on Networks, Computers and Communications (ISNCC), Yasmine Hammamet, Tunisia, 11–13 May 2016; pp. 1–6. [CrossRef]
103. Kang, M.J.; Kang, J.W. Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security. *PLoS ONE* **2016**, *11*, e0155781. [CrossRef] [PubMed]
104. ElSayed, M.S.; Le-Khac, N.A.; Albahar, M.A.; Jurcut, A. A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique. *J. Netw. Comput. Appl.* **2021**, *191*, 103160. [CrossRef]
105. Shaji, N.S.; Jain, T.; Muthalagu, R.; Pawar, P.M. Deep-discovery: Anomaly discovery in software-defined networks using artificial neural networks. *Comput. Secur.* **2023**, *132*, 103320. [CrossRef]
106. Russell, S.J.; Norvig, P. *Artificial Intelligence: A Modern Approach*; Pearson Education Limited: London, UK, 2016; p. 1132.
107. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
108. Kumar, K.; Thakur, G.S.M. Advanced applications of neural networks and artificial intelligence: A review. *Int. J. Inf. Technol. Comput. Sci.* **2012**, *4*, 57–68. [CrossRef]
109. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press, Cambridge, MA, USA, 2016; Volume 1, p. 800.
110. Hagan, M.T.; Menhaj, M.B. Training feedforward networks with the Marquardt algorithm. *IEEE Trans. Neural Netw.* **1994**, *5*, 989–993. [CrossRef]
111. Li, M.; Wu, H.; Wang, Y.; Handroos, H.; Carbone, G. Modified Levenberg–Marquardt algorithm for backpropagation neural network training in dynamic model identification of mechanical systems. *J. Dyn. Syst. Meas. Control* **2017**, *139*, 031012. [CrossRef]
112. Radford, A.; Metz, L.; Chintala, S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv* **2015**, arXiv:1511.06434. [CrossRef]
113. Creswell, A.; White, T.; Dumoulin, V.; Arulkumaran, K.; Sengupta, B.; Bharath, A.A. Generative adversarial networks: An overview. *IEEE Signal Process. Mag.* **2018**, *35*, 53–65. [CrossRef]
114. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
115. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
116. Esteban, C.; Hyland, S.L.; Rätsch, G. Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs. *arXiv* **2017**, arXiv:1706.02633. [CrossRef]
117. Lim, W.; Chek, K.; Lau, B.; Lin, C. Future of Generative Adversarial Networks (GAN) for Anomaly Detection in Network Security: A Review. *Comput. Secur.* **2024**, *139*, 103733. [CrossRef]

118. Dunmore, A.; Jang-Jaccard, J.; Sabrina, F.; Kwak, J. A Comprehensive Survey of Generative Adversarial Networks (GANs) in Cybersecurity Intrusion Detection. *IEEE Access* **2023**, *11*, 76071–76094. [CrossRef]
119. Goodfellow, I. Nips 2016 tutorial: Generative adversarial networks. *arXiv* **2016**, arXiv:1701.00160.
120. Wu, X.; Kumar, V.; Ross Quinlan, J.; Ghosh, J.; Yang, Q.; Motoda, H.; McLachlan, G.J.; Ng, A.; Liu, B.; Yu, P.S.; et al. Top 10 algorithms in data mining. *Knowl. Inf. Syst.* **2008**, *14*, 1–37. [CrossRef]
121. Breiman, L.; Friedman, J.H.; Olshen, R.A.; Stone, C.J. *Classification and Regression Trees*; Wadsworth International Group: Belmont, CA, USA, 1984 ; p. 368.
122. Loh, W.Y. Classification and regression trees. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2011**, *1*, 14–23. [CrossRef]
123. Burkov, A. *The Hundred-Page Machine Learning Book*; Andriy Burkov: Québec, QC, Canada, 2019; Volume 1, pp. 1–152.
124. The MathWorks, Inc. Statistics and Machine Learning Toolbox. 2025. Available online: https://uk.mathworks.com/products/statistics.html (accessed on 6 January 2025).
125. scikit-Learn. scikit-Learn Machine Learning in Python. 2025. Available online: https://scikit-learn.org/stable/ (accessed on 6 January 2025).
126. Sangodoyin, A.O. Design and Analysis of Anomaly Detection and Mitigation Schemes for Distributed Denial of Service Attacks in Software Defined Networks; Ph.D. Thesis, University of Bradford, Bradford, UK, 2019; pp. 1–157. Available online: https://bradscholars.brad.ac.uk/handle/10454/18777 (accessed on 5 April 2023).
127. Akinsolu, M.O.; Zribi, K. A Generalized Framework for Adopting Regression-Based Predictive Modeling in Manufacturing Environments. *Inventions* **2023**, *8*, 32. [CrossRef]
128. Wilcoxon, F. Individual comparisons by ranking methods. In *Breakthroughs in Statistics: Methodology and Distribution*; Springer: Berlin/Heidelberg, Germany, 1992; pp. 196–202.
129. Forbes, C.; Evans, M.; Hastings, N.; Peacock, B. *Statistical Distributions*; John Wiley & Sons: Hoboken, NJ, USA, 2011; Volume 4, p. 230.
130. Memar, P.; Faradji, F. A Novel Multi-Class EEG-Based Sleep Stage Classification System. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2018**, *26*, 84–95. [CrossRef] [PubMed]
131. Augustine, M.T. A Survey on Universal Approximation Theorems. *arXiv* **2024**, arXiv:2407.12895. [CrossRef]
132. Brown, I.; Mues, C. An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Syst. Appl.* **2012**, *39*, 3446–3453. [CrossRef]
133. Bhardwaj, S.; Dave, M. Enhanced neural network-based attack investigation framework for network forensics: Identification, detection, and analysis of the attack. *Comput. Secur.* **2023**, *135*, 103521. [CrossRef]
134. Sahi, A.; Lai, D.; Li, Y.; Diykh, M. An efficient DDoS TCP flood attack detection and prevention system in a cloud environment. *IEEE Access* **2017**, *5*, 6036–6048. [CrossRef]
135. Nashat, D.; Khairy, S. Detecting http flooding attacks based on uniform model. In Proceedings of the 2021 International Conference on Networking and Network Applications (NaNA), Lijiang, China, 29 October–1 November 2021; pp. 94–98.
136. Shen, Z.Y.; Su, M.W.; Cai, Y.Z.; Tasi, M.H. Mitigating SYN Flooding and UDP Flooding in P4-based SDN. In Proceedings of the 2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS), Tainan, Taiwan, 8–10 September 2021; pp. 374–377.
137. Nouri, A.; van Treeck, C.; Frisch, J. Sensitivity Assessment of Building Energy Performance Simulations Using MARS Meta-Modeling in Combination with Sobol'Method. *Energies* **2024**, *17*, 695. [CrossRef]
138. Liu, Q.; Tong, N.; Wu, X.; Han, X.; Chen, C. A generalized sensitivity analysis method based on variance and covariance decomposition of summatory functions for multi-input multi-output systems. *Comput. Methods Appl. Mech. Eng.* **2021**, *385*, 114009. [CrossRef]
139. Biazar, D.; Khaloozadeh, H.; Siahi, M. Sensitivity analysis for evaluation of the effect of sensors error on the wind turbine variables using Monte Carlo simulation. *IET Renew. Power Gener.* **2022**, *16*, 1623–1635. [CrossRef]
140. Arman, S.A.; Rahman, M.M.; Rahman, S.F.; Urmi, N.P.; Urmee, P.P.; Muslim, N.; Islam, S. Developing an IoT Networks-based Testbed for Software-Defined Networks. In Proceedings of the 2020 IEEE Region 10 Symposium (TENSYMP), Dhaka, Banglades, 5–7 June 2020; pp. 1752–1755.