Securing IIoT Applications in 6G and Beyond using Adaptive Ensemble Learning and Zero-Touch Multi-Resource Provisioning

Zakaria Abou El Houda[a], Bouziane Brik[b] and Adlen Ksentini[c]

[a]Institut National de la Recherche Scientifique, Centre Energie, INRS-EMT, Varennes (Quebec), J3X 1S2 Canada.
[b]Computer Science Department, College of Computing and Informatics, University of Sharjah, Sharjah, UAE.
[c]Communication Systems Department, EURECOM, France.

ARTICLE INFO

Keywords:
Zero-Touch Multi-resource Provisioning (ZSP)
6G and beyond
Software-defined Networking (SDN)
Digital Twins (DT)
Artificial Intelligence (AI)
Ensemble Learning (EL).

ABSTRACT

The advent of Industry 5.0, driven by cutting-edge 6G technologies such as immersive cloud eXtended Reality (XR), Autonomous vehicles, holographic communication, and Digital Twin (DT), is set to trigger a substantial upswing in the deployment of the Industrial Internet of Things (IIoT) devices interconnected within networks. This expansion of IIoT devices will create a wider attack surface and increase the risk of data breaches, privacy violations, and system disruptions. Therefore, it is essential to design innovative mechanisms to ensure the reliability and security of these advanced 6G applications, as well as the IIoT devices that support them. In this context, we propose a novel Software-defined Networking (SDN)-based Ensemble Learning (EL) Framework for Secure IIoT applications in 6G and beyond, called AdaptSDN. The proposed framework leverages SDN technology to dynamically allocate network resources and deploy security measures on demand. It also leverages EL techniques to improve the intrusion detection system's accuracy. By isolating IIoT devices into network slices, the framework limits the impact of attacks and reduces the potential for cascading failures. Digital twins are used for creating a virtual replica of the IIoT network, allowing for a real-time security threat detection. In particular, AdaptSDN includes three main modules: (1) A novel digital Twin (DT)-enabled data gathering and selection of informative features module to achieve two main objectives: reducing computational complexity and improving detection performance; (2) An SDN-based lightweight adaptive boosting module that uses an advanced boosting EL techniques to dynamically adjust weights and to effectively identify and respond to IIoT attacks in real-time; and (3) A zero-touch resources provisioning module that employs a non-cooperative game theory approach. This approach allows for automatically provisioning various resources in a network to efficiently mitigate network attacks; it enables SDN nodes to obtain the required virtual resources, *i.e.*, storage, computing, and bandwidth, from the main orchestrator with respect to IIoT attack type. We carried out comprehensive experiments to assess the effectiveness of our proposed framework in detecting real-world IIoT attacks. The numerical results confirm that AdaptSDN has the potential to enable secure and reliable IIoT applications in 6G and beyond, meeting the stringent service requirements of the new emerging applications.

1. Introduction

The advent of Industry 5.0, driven by cutting-edge 6G technologies such as immersive cloud XR, autonomous vehicles, holographic communication, and digital twins will require networks with extremely high data speeds, low latency, and high bandwidth. This is because these applications rely on real-time data transmission and processing, as well as large amounts of data transfer [1, 2, 3]. To achieve this, 6G networks will need to employ new technologies such as millimeter-wave communication, terahertz frequencies, and advanced antenna systems. These technologies will enable faster and more efficient data transfer, as well as greater network capacity and coverage. In addition to these technological advancements, the emergence of advanced 6G applications is also expected to lead to an increase in the number of Industrial Internet of Things (IIoT) devices connected to networks. This will require the development of new data processing and storage technologies, such as edge computing and distributed cloud computing [3]. These technologies will enable data to be processed and stored closer to the source, reducing latency and improving response times. Also, the increase in the number of IIoT devices connected to networks raises concerns about security and privacy.

The emergence of advanced 6G applications is expected to create even greater security risks than those associated with 5G or earlier generations of wireless technology [4]. Some of the potential security issues of 6G networks include (1) increased attack surface: 6G networks are expected to have a vast number of interconnected devices, which means that the attack surface for cybercriminals will also be much larger. With more devices connected to the network, there will be more potential entry points for attackers to exploit; (2) greater complexity: 6G networks will be more complex than previous generations of wireless technology, with more advanced features and capabilities. This increased complexity could make it more difficult to detect and mitigate new security threats; (3) privacy concerns: with more devices connected to the network and transmitting data, there will be a greater risk of data breaches and privacy violations. It will be important to ensure that the security measures in place are sufficient to protect user data privacy; (4) insider threats:

zakaria.abouelhouda@inrs.ca (Z. Abou El Houda); bouziane.brik@gmail.com (B. Brik); adlen.ksentini@eurecom.fr (A. Ksentini)
ORCID(s):

as with any network, there will always be the risk of insider threats. This includes malicious insiders who may attempt to exploit vulnerabilities or steal sensitive information; and (5) new attack vectors: 6G networks will introduce new attack vectors that were not present in previous generations of wireless technology. For example, the use of edge computing and distributed cloud computing could introduce new security risks [5].

SDN and NFV are two key technologies that have been widely used in the telecommunications industry for several years. As we move towards the development of 6G networks, it is expected that these technologies will play an even more significant role in shaping the future of network architecture and design. Securing IIoT applications in 6G and beyond can be challenging due to the large number of connected devices and the complexity of the network architecture. Thus, the implementation of SDN and Digital twins has the potential to enhance the security of IIoT applications. Specifically, intrusion detection systems (IDSs) are essential for safeguarding 6G networks in the context of SDN. SDN allows for centralized control of network resources, which can make it easier to deploy and manage IDSs across the network.

However, current IDSs may not be effective against zero-day attacks, while adaptive IDSs may suffer from high false positive rates. A possible strategy involves leveraging Artificial Intelligence (AI) to create intrusion detection systems that are more resilient, enabling them to promptly detect security breaches. By analyzing network traffic patterns and identifying anomalies, AI systems could be used to identify potential threats and alert network administrators to take appropriate action. Another potential use for AI in securing 6G networks is to enhance authentication and access control mechanisms. Additionally, AI systems could be used to detect and block unauthorized access attempts in real-time. AI could also be used to enhance encryption and data protection mechanisms. By analyzing patterns in data usage and identifying potential vulnerabilities, AI systems could be used to develop more robust encryption algorithms and protect against data breaches. On the other side, Digital twins can be used for real-time monitoring. Digital twins can be used to create a virtual replica of the IIoT network, allowing operators to monitor the network in real-time and detect potential security threats. By using digital twins, operators can quickly respond to security incidents and prevent them from causing damage. Also, Digital twins can be used to predict when equipment or software may fail, allowing for maintenance to be carried out before a security breach can occur. This helps to prevent security breaches and minimize their impact[6, 7].

EL can be used to enhance the detection capability of individual models and improve the security of 6G networks. One possible approach to using ensemble learning for IIoT attack detection in 6G is to build a system that incorporates multiple machine learning algorithms and models, each designed to detect different types of attacks. The output of each model can then be combined using a weighted average, majority voting, or other ensemble methods to generate a final prediction. For example, Boosting, as a machine learning technique, can be leveraged to improve the reliability of IDSs. Boosting combines a set of weak learners to become strong learners, by combining their outputs through weighted majority voting. This helps in reducing the under-fitting and over-fitting issues and makes the IDS more robust against different types of attacks. Boosting can be used to improve the accuracy of IDSs by enhancing the detection of malicious network traffic. By analyzing the patterns of network traffic, the IDS can identify suspicious behavior and flag potential threats. Boosting can help to improve the accuracy of this process by enabling the IDS to better distinguish between normal and malicious traffic.

In this context, we introduce a novel EL framework, which encompasses three separate modules: (1) A novel digital twin (DT)-enabled data gathering and selection of the most informative features module to achieve two main objectives: reducing computational complexity and improving detection performance. The data collection module can be designed to collect data from various sources within the IIoT network, such as network traffic, system logs, and device behavior; (2) A novel SDN-based lightweight adaptive boosting module that uses advanced boosting EL techniques to dynamically adjust its boosting algorithm and to effectively identify and respond to IIoT attacks in real-time; and (3) A non-cooperative game theory-based module for mitigating IIoT attacks. This module allows SDN controllers, at the edge level or MEC (Multi-Access Edge Computing), to efficiently access virtual resources from the resources orchestrator to effectively combat different types of IIoT attacks. This approach considers the competition between SDN controllers to obtain virtual resources and optimize their objectives, while also contributing to the overall security of the IIoT network. The concept of Zero-Touch Multi-resource Provisioning (ZSP) involves the automatic allocation and management of resources within the network infrastructure to detect and mitigate attacks without human intervention. It leverages advanced technologies and intelligent algorithms to dynamically allocate computing, storage, and networking resources based on the current attack patterns, network conditions, and security requirements. The proposed framework is designed to consider various factors, such as the types of attacks, the available virtual resources, and the objectives of the MEC nodes. By optimizing the allocation of virtual resources and considering the competition between SDN controllers, the framework can improve the efficiency and effectiveness of IIoT attack mitigation, leading to better overall network security and performance.

The paper is structured as follows: Section 2 provides a review of related work, while Section 3 details our proposed framework in terms of its main modules. Section 4 focuses on the zero-touch resources provisioning module, while Section 5 evaluates the effectiveness of our proposed framework. We conclude the paper in Section 6.

2. Related Work

The new emergence of IIoT botnets poses increasing security issues, demanding robust defense mechanisms. IDS, as a crucial component, plays a pivotal role in monitoring and analyzing network activities to identify and respond to potential security threats promptly. IDS-based AI has become an increasingly popular approach for detecting and responding to security threats. In the following, we will examine some of the most notable AI-based IDSs and the associated security issues they face.

AI has been used extensively to enhance the detection capabilities of IDSs. These algorithms learn patterns from historical data and use them to detect new attacks in real time. Advanced AI techniques [8, 9, 10, 11, 12] have emerged as a powerful tool for intrusion detection due to their ability to learn complex patterns and features from data. On the other side, Hybrid approaches [13] that combine multiple AI techniques have been shown to outperform individual techniques in detecting network attacks. Hybrid approaches can also be used to improve the robustness of intrusion detection systems against adversarial attacks. Adversarial machine learning has been leveraged to improve the security of IDSs. This approach involves generating adversarial examples that can evade detection by the intrusion detection system. However, one challenge with adversarial machine learning is the need for large amounts of computing resources and time to generate adversarial examples. Also, Explainable AI (XAI) [14] has gained attention in recent years as a way to enhance the transparency and trustworthiness of AI-based intrusion detection systems. XAI methods can help users understand how the system makes decisions and identify potential biases or errors in the system.

Our analysis indicates that some solutions for detecting cyber attacks in networks, such as SVM [9] and NN [10], rely on a single learner. However, these systems are limited in their ability to identify "zero-day" attacks, and the sheer volume of data generated can make it challenging to deal with different types of attacks effectively. To overcome these limitations, we propose a novel SDN-based Ensemble Learning Framework for Secure IIoT applications in 6G and beyond. The proposed framework leverages SDN technology to dynamically allocate network resources and deploy security measures on demand. It also utilizes ensemble learning techniques to improve the accuracy of intrusion detection and classification.

3. AdaptSDN: A Lightweight adaptive Ensemble Learning-based Framework for IIoT attack detection in SDN

In this section, we describe AdaptSDN. Our proposed architecture is initially introduced, followed by the presentation of our module for IIoT data gathering and feature selection. Additionally, we present our SDN-based lightweight adaptive boosting module, which employs advanced boosting EL techniques to promptly detect and counter IIoT attacks in real-time. Finally, we highlight our zero-touch resources provisioning module that employs a non-cooperative game theory approach to efficiently mitigate IIoT attacks.

3.1. System Architecture

AdaptSDN is designed to enhance the security of IIoT networks by leveraging SDN (see Figure 1); it comprises three layers: the IIoT devices layer, the SDN controller layer, and the cloud-based management layer. The IIoT devices layer comprises all the IIoT devices in the network, while the SDN controller layer comprises the SDN controller and the SDN switches. The cloud-based management layer provides centralized management and monitoring of the entire network. The IIoT data collection and optimized feature selection module are responsible for collecting data from the IIoT devices in the network and selecting the most relevant features for attack detection. This module employs machine learning techniques to analyze the collected data and identify the most important features that can be used to detect attacks. The SDN-based lightweight adaptive boosting module is responsible for detecting attacks in real-time. This module uses advanced boosting EL techniques to identify attacks and adaptively adjust the classification thresholds based on the changing attack patterns. This enables the system to accurately detect attacks while minimizing false positives. The zero-touch resources provisioning module employs a non-cooperative game theory approach to mitigate attacks. This module uses a game-theoretic model to analyze the behavior of the attacker and the defender and determine the optimal mitigation strategy. This approach allows the system to effectively mitigate attacks while minimizing the impact on legitimate traffic. Thus, the AdaptSDN framework provides an effective and efficient approach to enhancing the security of IIoT networks. The framework's modular architecture allows for easy integration and customization of different modules, making it adaptable to different IIoT network environments and attack scenarios.

1. **IIoT devices plane:** comprises all the physical devices that are connected to the network and may gather data and transmit it to other devices, to the MEC nodes, or to the Cloud. These devices include sensors, actuators, cameras, smart appliances, and other types of devices that are designed to collect data or interact with the environment.
2. **SDN plane:** The SDN controller layer is a key component that is responsible for managing and controlling the network infrastructure and traffic flows by providing a centralized interface for network administrators to configure, monitor, and manage the network. The SDN controller layer includes the SDN controller, which is a software application that runs on a server and communicates with the network devices, such as switches and routers, through a southbound interface using protocols such as OpenFlow. The SDN controller also communicates with network management

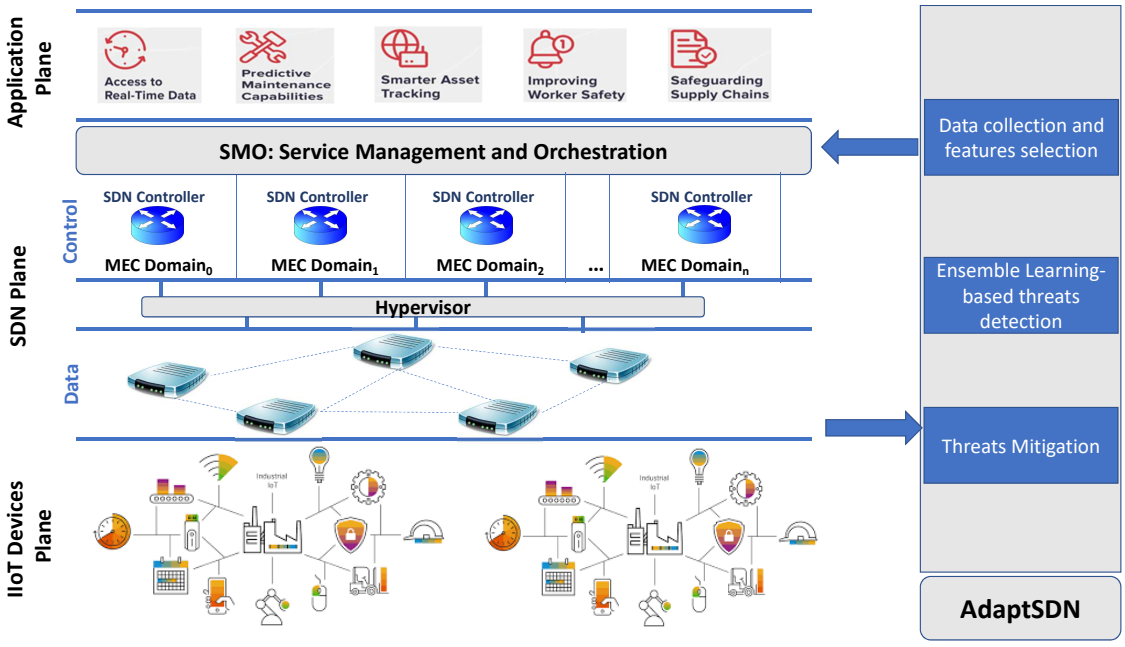

Figure 1: System Architecture.

and orchestration systems through a northbound interface to enable network automation and programmability.

3. **Application plane:** This layer is typically implemented using cloud-based services that can scale easily, and provide high availability and robust security features. One of the primary functions of the cloud-based management layer is to provide a unified view of the network and its devices, applications, and services. This enables network administrators to manage the network from a single pane of glass, making it easier to monitor and troubleshoot issues. In addition, the cloud-based management layer can leverage AI and machine learning algorithms to analyze network data and identify patterns or anomalies. This can help detect and prevent security threats, optimize network performance, and predict potential issues before they occur. AdaptSDN is an EL-based framework that can be used to enhance the security of IIoT devices. At the application plane, the deployment of AdaptSDN takes the form of an application. Deploying this framework at the application plane can provide additional security to the SDN system by analyzing network traffic and detecting potential security threats. The framework can also use historical data to detect patterns and predict potential security threats.

3.2. Digital Twin (DT)-enabled Data collection and Optimized Feature Selection Module

Digital twin technology is a virtual model that represents physical systems, processes, and assets. It can be used to simulate, predict, and optimize the performance of physical systems. In recent years, digital twin technology has been used in various fields, including manufacturing, healthcare, and transportation, to name a few. To achieve the two main objectives of reducing computational complexity and improving detection performance, a novel digital twin-enabled data collection and optimized feature selection module can be developed. This module can be integrated with the existing digital twin technology to enhance its capabilities. The data collection module can be designed to collect real-time data from physical systems using sensors and other monitoring devices. The collected data can be fed into the digital twin model, which can then simulate the behavior of the physical system. This can help to identify patterns and anomalies in the data, which can be used to improve the detection performance.

Data collection Scheme: Data collection and feature selection are important steps in building an effective intrusion detection framework for an SDN-based system. The goal of data collection is to gather relevant network data, such as network traffic flows and packet information, that can be used to train machine learning models for intrusion detection. The data collection module can be implemented using various techniques, such as network taps or port mirroring, to collect data from various points in the network. The collected data can then be preprocessed to extract relevant features for intrusion detection. sFlow and OpenFlow are both protocols that can be used for data network collection in SDN environments. However, they have different functions and features. sFlow is a packet sampling technology that allows network administrators to collect traffic data from network devices, such as switches and routers. It can monitor and analyze traffic flows in real-time and provides information on network performance, usage, and security.

sFlow works by sampling a portion of network traffic and forwarding it to a collector, where it is analyzed and processed. OpenFlow, on the other hand, is a protocol that enables the centralized control of network switches and routers in SDN environments. It allows network administrators to define how traffic should be forwarded through the network by programming the behavior of network devices. OpenFlow can be used to manage and optimize network traffic flows, and it can also be used for network security and monitoring. In terms of data network collection, sFlow is better suited for monitoring network traffic and collecting data on network performance and usage. It provides a detailed view of network activity and can be used to troubleshoot network issues. sFlow is designed for network traffic monitoring and analysis, and it uses packet sampling to collect data from network devices. sFlow is capable of being implemented on a vast scale and acquiring data from numerous network devices without causing any negative effects on network performance. Because sFlow only samples a portion of the traffic, it can handle high-speed traffic and can scale to meet the needs of even the largest networks.

For the attack traffic, we used real-world datasets such as NSW-NB15 and NSL-KDD to simulate attack traffic. These datasets include a variety of IIoT-based attacks, including fuzzers, DDoS, analysis, reconnaissance, backdoors, and others, which can be used to train and test ML models. NSL-KDD is a widely used dataset for evaluating intrusion detection systems in a network environment. This dataset represents an upgrade to the first version of the KDD Cup 99 dataset, which was proposed in 1999, as a standard measure to assess the efficacy of intrusion detection systems. The NSL-KDD dataset was created to address some of the limitations of the original KDD Cup 99 dataset and to provide a more realistic evaluation of intrusion detection systems. The NSL-KDD dataset contains both normal and attack traffic data, and includes a total of 41 features extracted from network traffic flows, including protocol types, source and destination IP addresses, source and destination port numbers, and other network traffic attributes.

The NSW-NB15 dataset includes both normal and attack traffic data and contains a total of 49 features extracted from network traffic flows. The dataset includes six different types of attacks, including fuzzers, analysis, backdoors, DoS, exploits, and reconnaissance. The NSL-KDD dataset has been widely used in research to develop and evaluate intrusion detection systems for network-based systems. By using this dataset, researchers can train and test machine learning models to accurately detect and respond to various types of attacks in a network-based system, leading to better security for the system. Using real-world datasets such as NSW-NB15 and NSL-KDD can provide a more realistic evaluation of the effectiveness of the intrusion detection framework for an IIoT-based system. The data preprocessing step consists of cleaning data and making them suitable for analysis. One common data preprocessing technique is normalization, which scales the values of a feature to a specified range. Normalization can help optimize the performance of machine learning algorithms. The formula for data normalization is as follows:

$$\hat{X} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

where X denotes the original collected value of the feature, while X_{min} and X_{max} represent the lowest and highest values of the original feature, respectively.

Feature Selection Scheme: In this paper, we have considered two common techniques: (1) Correlation-based Feature Selection that selects the most significant/relevant features based on their high correlation score with the target; and (2) Boosting-based Feature Selection, described as follows:

Boosting Feature Selection is an ensemble-based feature selection algorithm that aims to select the most important features in a dataset by iteratively assigning weights to each feature and selecting the top-weighted features. The algorithm works by training a sequence of weak learners on the dataset and assigning importance weights to the features based on how well they perform in the training. The weights are updated at each iteration based on the performance of the previous weak learner.

The objective function for Boosting Feature Selection is defined as follows:

$$\min_{\beta,w} \sum_{i=1}^{n} w_i e^{-y_i f(x_i)} + \lambda \sum_{j=1}^{p} \beta_j \tag{2}$$

where n is the number of samples, p is the number of features, y_i is the binary label for sample i, $f(x_i)$ is the current prediction for sample i, w_i is the weight assigned to sample i, lambda is the regularization parameter, $beta_j$ is the weight assigned to feature j.

At each iteration of Boosting, a weak learner $h_m(x_i)$ is trained on the dataset using the weights assigned to each sample. The weights are updated for each sample i as follows:

$$w_i^{(m+1)} = w_i^{(m)} e^{-\alpha_m y_i h_m(x_i)} \tag{3}$$

where $w_i^{(m)}$ is the weight assigned to sample i at iteration m, $alpha_m$ is the learning rate at iteration m.

The feature importance weights S_j are updated for each feature j as follows:

$$S_j^{(m+1)} = S_j^{(m)} + \frac{\alpha_m}{Z} \sum_{i=1}^{n} w_i^{(m)} y_i h_m(x_i) \mathbb{1}(x_{ij}) \tag{4}$$

where $S_j^{(m)}$ is the feature importance weight for feature j at iteration m, Z is a normalization factor, x_{ij} is the value of feature j for sample i, and $mathbb1$ is the indicator function that equals 1 if x_{ij} is non-zero and 0 otherwise.

The final feature importance weights are obtained by averaging the weights over all iterations:

$$S_j = \frac{1}{M} \sum_{m=1}^{M} S_j^{(m)} \tag{5}$$

where M is the total number of iterations.

Hyperparameter Selection Scheme: Hyperparameter optimization is a crucial step in machine learning model development. Optimizing the hyperparameters of a model can yield substantial enhancements in its performance since the model's efficacy is contingent on the values assigned to its hyperparameters. Here is an example of a hyperparameter optimization scheme using grid search: Let $\mathcal{M}$ be a machine learning model with hyperparameters $\boldsymbol{\theta}$. We want to find the values of $\boldsymbol{\theta}$ that optimize the model's performance on a given dataset $\mathcal{D}$. We define a grid of hyperparameter values $\boldsymbol{\Theta}$ as follows:

$$\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_n\} \tag{6}$$

where each $\boldsymbol{\theta}_i$ is a vector of hyperparameter values.

We then evaluate the model's performance for each combination of hyperparameters in $\boldsymbol{\Theta}$ using cross-validation. Let $L(\mathcal{M}, \boldsymbol{\theta}_i, \mathcal{D})$ be the performance metric for the model $\mathcal{M}$ with hyperparameters $\boldsymbol{\theta}_i$ on the dataset $\mathcal{D}$, obtained using cross-validation. We can then find the optimal hyperparameters $\boldsymbol{\theta}^*$ as follows:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}_i \in \boldsymbol{\Theta}} L(\mathcal{M}, \boldsymbol{\theta}_i, \mathcal{D}) \tag{7}$$

This optimization scheme is known as grid search, and it involves searching over a discrete grid of hyperparameter values. Other optimization schemes, such as random search and Bayesian optimization, involve searching over continuous or probabilistic distributions of hyperparameters.

3.3. A Lightweight adaptive Ensemble Learning-enabled Scheme for IIoT attack detection

AdaptSDN is a novel variation of boosting methods that employs the process of gradient descent to optimize the values of $\hat{z}$, and thus reduce a specified loss function. In the AdaptSDN, the gradient descent process is used to optimize the weights of the weak classifiers used in the ensemble. The weak classifiers are combined using a weighted majority voting scheme to make the final decision on whether a network traffic flow is normal or malicious. AdaptSDN's loss function is defined as follows:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i * log(\hat{y_i}) \tag{8}$$

where N is the number of data, y_i represents the ground truth vector (*i.e.*, observed values) for the i^{th} class and $\hat{y_i}$ represents the predicted value for the i^{th} class.

Assuming a dataset $D = (x_1, y_1), (x_2, y_2), ..., (x_k, y_k)$ where x_k represents the k^{th} input feature vector and y_i is the corresponding binary label, either 0 (normal) or 1 (attack). The goal is to learn a classification model $f(x)$ that can accurately predict whether an input is normal or attacked. AdaptSDN initializes the model with a constant value:

$$f_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma) \tag{9}$$

where L is the loss function, and γ is a constant value.

Next, the computation of pseudo-residuals for the k^{th} data sample at iteration M is carried out in the following manner:

$$r_{km} = -[\frac{\partial \mathcal{L}(y_k, H(X_k))}{\partial (H(X_k))}]_{H(X)=H_{m-1}(x)} \tag{10}$$

Then, we fit a regression tree to the pseudo-residuals:

$$\gamma_m = \arg\min_{\gamma} \sum_{k=1}^{K} \mathcal{L}(y_k, H_{m-1}(x_k) + \gamma f_m(x_k)) \tag{11}$$

Then, we compute the optimal weights for the regression tree:

$$\mathcal{L}(y_k, H_{m-1}(x_k)+\gamma f_m(x_k)) \simeq \sum_{k=1}^{K}[\mathcal{L}(y_k, H_{m-1}(x_k))+g_k \gamma f_m(x_k) + \frac{1}{2}h_k\gamma^2 f_m(x_k)] \tag{12}$$

where g_k and h_k represent the first and second order gradients of $\mathcal{L}(y_k, H(x_k))$ with respect to $(H(x_k))$, respectively.

Next, we calculate the loss function derivative concerning γ in the following manner:

$$\partial\mathcal{L}(y_k, H_{m-1}(x_k)+\gamma f_m(x_k)) \simeq \sum_{k=1}^{K}[g_k f_m(x_k)+h_k\gamma f_m(x_k)] \tag{13}$$

We determine the optimal value of γ at iteration t by:

$$\gamma_m = -\frac{\sum_{k=1}^{K} g_k f_m(x_k)}{\sum_{k=1}^{K} h_k f_m(x_k)} \tag{14}$$

Lastly, we update the final model at iteration t in the following manner:

$$H_m(x) = H_{m-1}(x) + \gamma_m f_m(x_k) \tag{15}$$

While AdaptSDN shares similarities with gradient boosting, it incorporates some minor improvements in its regularization strategy. In gradient boosting, the optimization function at each iteration m is defined as the sum of the previous predictions $H_{m-1}(x_k)$ and a new model $h_m(x_k)$, where x_k is the k-th data sample and $H_{m-1}(x_k)$ is the prediction of the model up to the previous iteration. The objective is to minimize the loss function $\mathcal{L}(y_k, H_m(x_k))$ between the true labels y_k and the current prediction $H_m(x_k)$.

However, this approach can lead to over-fitting issues as the model may learn the training data too well, resulting in degraded performance on unseen data. To address this, AdaptSDN uses a more sophisticated regularization strategy that penalizes the model complexity. Specifically, the optimization function in AdaptSDN at iteration t is defined as the sum of the previous predictions $H_{m-1}(x_k)$, a new model $h_m(x_k)$, and a regularization term $R(\gamma)$. The objective is to minimize the loss function $\mathcal{L}(y_i, H_m(x_k))$ between the true labels y_k and the current prediction $H_m(x_k)$, while also minimizing the regularization term $R(\gamma)$. The regularization term $R(\gamma)$ can take many forms, depending on the specific problem being solved. In general, it encourages the model to have simpler structures that are less likely to over-fit the training data. This helps to improve the generalization performance of the model on unseen data. The optimization function at iteration m is modified as follows:

$$\mathcal{L}^m \simeq \sum_{k=1}^{K} \mathcal{L}(y_k, H_{m-1}(x_k) + f_m(x_k)) + \Omega(f_m) \quad (16)$$

where $\Omega(f_m) = \frac{1}{2}\lambda \parallel \xi \parallel$, λ is a regularization parameter.

To improve the loss function, AdaptSDN employs a Taylor polynomial of second-order approximation, which can be expressed as follows:

$$\mathcal{L}^m \simeq \sum_{i=1}^{N} [\mathcal{L}(y_k, H_{t-1}(x_k)) + g_k f_m(x_k) + \frac{1}{2} h_k f_m(x_k)^2] + \frac{1}{2}\lambda \sum_{j=1}^{m} \xi_j^2$$

where T is the number of tree's leaves, and $I(x_k \in R_k)$ is an indicator function that takes the value 1 if the k-th data sample belongs to the k-th leaf node, and 0 otherwise.

By optimizing this approximation using gradient descent, AdaptSDN can improve the performance of the model and avoid over-fitting issues.

4. Zero-touch resources provisioning module

The AdaptSDN algorithm comprises two integral phases: detection and mitigation, with a specific focus on efficient attack response through a Zero-touch Resources Provisioning Module. Upon detecting an IIoT attack, SDN controllers at the MEC level engage in a competition to acquire additional resources from the resource orchestrator ($ROrche$), to be able then in mitigating the detected attack. The needed amount of virtual resources, including bandwidth, CPU, and storage, is primarily determined by the attack type and the critical applications already running on each MEC node. For example, dealing with a DDoS attack requires more vCPU (virtual Central Processing Unit) resources than handling scanning attacks such as a User to Root Attack (U2R). To address this challenge, we propose a non-cooperative game model for SDN controllers to scale down or up their virtual resources with respect to the attack type and running critical applications. Although our focus is mainly on vCPU resources, our approach can consider other types of resources such as bandwidth and storage.

4.1. Non-cooperative game formulation

To obtain the needed vCPU resources, we design a non-cooperative game, denoted as $G = \left(P, S_i, \Phi_i\right)_{i \in P}$, which models the competitive behavior of SDN controllers as follows:

1. We consider a set P of m SDN controllers players, denoted as $p_1, \ldots, p_i, \ldots, p_m$, connected to a common virtual resource orchestrator, $ROrche_j$.
2. *SDN Controllers' strategies*, S_i: reflects the available actions for each SDN controller p_i in the game, where i belongs to the set of players P, it is possible for players to request vCPU resources within the range of zero to a maximum value of η^{max}.
3. Each player p_i in P has a set of strategies, denoted as S_i, which correspond to the actions that they can take during the game. Specifically, each SDN controller player p_i may request a certain amount of vCPU resources, ranging from zero up to a maximum value of η^{max}. Thus, the strategy profile for all players of SDN controllers can be represented as $S_i = \left[0, \eta_i^{max}\right]$ and $S = \prod_{i=1}^{m} S_i = \left[0, \eta_1^{max}\right] \times \cdots \times \left[0, \eta_i^{max}\right] \times \cdots \times \left[0, \eta_m^{max}\right]$.
4. For each player p_i of the SDN controllers, a payoff function $\Phi_i : S_i \rightarrow \mathbb{R}$ is defined. The goal of each player is to maximize their payoff function Φ, which in turn will increase their profit by obtaining more vCPU (η_i).

In addition, the SDN controllers' payoff function is designed to incorporate three primary components: (1) the objective of the SDN controllers to increase the vCPU resources allocated by the main orchestrator; (2) the attack priority cost is determined to prioritize identified attacks; and (3) the critical applications running on each SDN controller. The following is a definition of these functions:

1. Utility: the utility function represents the profit gained by the SDN controllers as they receive more vCPU resources. Various functions can be used as utility functions, including exponential, logarithmic, sigmoidal, square root, and linear functions [15]. For each player p_i, we choose the square root function as the utility function because of its strictly concave nature:

$$\upsilon_i\left(\eta_i\right) = \sqrt{\eta_i + 1}, with\ i = 1, \ldots, m \quad (17)$$

2. Cost of Attack Priority: the cost of attack priority takes into account both the number of attackers carrying out the attack and the priority of each attack j. To each attack type, we give a priority value $Pri_j =]0, 1]$ with respect to the required amount of vCPU resources. Furthermore, the attack priority cost is affected by the number of attackers involved, as attacks with the

higher number of attackers have a greater impact on the network and hence need more vCPU resources to be mitigated. The cost function is defined as follows:

$$\Upsilon_i\left(\eta_i, j\right) = \begin{cases} \eta_i * \left(\frac{1}{Pri_j * Attackers}\right), \text{Case of attack} \\ 1, \text{No Attack} \end{cases} \tag{18}$$

3. Cost of Critical Applications: When assigning vCPU resources to SDN controllers, it is important to consider the impact on other MEC applications. Our classification of IIoT applications includes two categories: (i) critical applications, which have strict latency requirements and are crucial for safety, such as collision detection and avoidance in industrial mobile robot systems, and (ii) non-critical applications, which encompass other types of applications like entertainment and advertising. Our model prioritizes the allocation of more resources to MEC nodes that support critical applications (Cri_App) to ensure their smooth operation. Therefore, we define the cost of critical applications for each SDN controller as:

$$\varrho_i\left(\eta_i, Cri\_App_i\right) = \eta_i * \left(1 - \frac{Cri\_App_i}{Total\_Apps_i}\right), \forall i \in P \tag{19}$$

The variable Cri_App_i represents the count of critical applications being run on MEC node i, whereas $Total\_Apps_i$ denotes the overall number of applications (both critical and non-critical) operating on the same MEC node.

Therefore, the payoff function for player p_i, who is an SDN controller deployed at a specific MEC node, can be expressed as follows:

$$\Phi_i\left(\eta_i, \eta_{-i}\right) = \alpha_i \upsilon_i\left(\eta_i\right) - \beta_i \Upsilon_i\left(\eta_i, j\right) - \psi_i \varrho_i\left(\eta_i, Cri\_App_i\right) \tag{20}$$

4.2. Proof of Nash equilibrium (NE)

NE corresponds to the situation when no SDN controller can improve their outcome by modifying its strategy, while the SDN controllers maintain their current strategies, the game has reached an NE. In this case, the game has a solution.

In our case, a set of asked vCPU resources, $s^* \in S$ with $s^* = \left[\eta_1^*, \ldots, \eta_i^*, \ldots, \eta_m^*\right]$, corresponds to a NE state if no SDN controller player can increase their payoff by changing their action. Nash equilibrium is represented by an N-tuple $\left\{\eta_i^*\right\}$, which guarantees:

$$\Phi\left(\eta_i^*, \eta_{-i}^*\right) \geq \Phi\left(\eta_i, \eta_{-i}^*\right), \forall i \in P, \eta_i^* \neq \eta_i \tag{21}$$

This subsection demonstrates that there is both a unique and a possible Nash equilibrium for our game G.

Nash Equilibrium Existence:

theorem 1 (Nikaido-Isoda). *We rely on the Nikaido-Isoda theorem to demonstrate the existence of a NE state in our game $G = \left(P, S_i, \Phi_i\right) i \in P$. The theorem states that an NE state exists if and only if certain conditions are met. Specifically, the set of SDN controllers' strategies Si must be both compact and convex, and their payoff function $\Phi\left(\eta_i, \eta_{-i}\right)$ must be continuous across all strategies $s \in S$ and concave in S_i.*

- *Because the range of values for S_i is from 0 to η_i^{max}, for all $i \in P$, the strategies of the SDN controllers are bounded and closed, making S_i a compact set. In addition, for any a_1 and a_2 in S_i and ζ ranging from 0 to 1, it is evident that $0 \leq (1-\zeta)a_2 + \zeta a_1 \leq \eta_i^{max}$. Since $\zeta a_1 + (1-\zeta)a_2$ is also in S_i, the set of strategies, S_i for all $i \in P$, is convex.*
- *We rely on the payoff function's Hessian matrix to demonstrate its property of concavity.*

$$H(s) = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1m} \\ h_{21} & h_{22} & \cdots & h_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ h_{m1} & h_{m2} & \cdots & h_{mm} \end{bmatrix} \tag{22}$$

Noting that $h_{kl} = \left(\frac{\partial^2 \Phi_k}{\partial \eta_k \partial \eta l}\right), \forall k, l \in P$. Thus, we have:

$$h_{kl} = \begin{cases} -\frac{\alpha_k}{\left(2\sqrt{\eta_k + 1}\right)^2} < 0 & ifk = l; \forall k, l \in P \\ 0 & ifk \neq l; \forall k, l \in P \end{cases} \tag{23}$$

It is evident that the Hessian matrix $H(s)$ is negative definite for every strategy $s \in S$. Therefore, using the leading principal minor of $H(s)$, we can conclude that $\Phi(\eta_i, \eta_{-i})$ is strictly concave in S_i. The application of the Nikaido-Isoda theorem leads to the conclusion that there is at least one Nash Equilibrium state in the game G.

Uniqueness of Nash Equilibrium (NE):

Considering a random values sequence $r = (r_1, r_2, \ldots, r_m)$ that are all positive. Rosen theorem [16] defines the positive weighted sum of $\Phi(\eta_i, \eta_{-i})$ for all $i \in P$ as follows:

$$\delta\left(\eta_i, \eta_{-i}; r\right) = \sum_{i=1}^{m} r_i \Phi_i\left(\eta_i, \eta_{-i}\right), r_i \geq 0, \forall i \in P. \tag{24}$$

The pseudo-gradient of $\delta\left(\eta_i, \eta_{-i}; r\right)$ is:

$$g\left(\eta_i, \eta_{-i}; r\right) = \begin{bmatrix} r_1 \nabla \Phi_1\left(\eta_1, \eta_{-1}\right) \\ r_2 \nabla \Phi_2\left(\eta_2, \eta_{-2}\right) \\ \vdots \\ r_m \nabla \Phi_m\left(\eta_m, \eta_{-m}\right) \end{bmatrix} \tag{25}$$

Noting that $\nabla\Phi_i\left(\eta_i,\eta_{-i}\right)=\frac{\alpha_i}{2\sqrt{\eta_i+1}}-\beta_i\left(\frac{1}{Pri_j*Attackers}\right)-\psi_i\left(1-\frac{Cri\_App_i}{Total\_Apps_i}\right)$

Then, we calculate the Jacobian matrix of g:

$$J\left(\eta_i,\eta_{-i},r\right)=\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mm} \end{bmatrix} \tag{26}$$

With $b_{ij}=r_i h_{ij};\forall i,j\in P$.

So, we can infer that for all $(\eta_i,\eta_{-i})\in S$, the symmetric matrix $\left[J+J^T\right]$ is negative definite. As a result, the game G has a unique NE by virtue of Rosen's theorem [16], with $\delta(\eta_i,\eta_{-i};r)$ being diagonally strictly concave.

The main process of our AdaptSDN framework, which is an Ensemble Learning-based IIoT Attack Detection system, is summarized in the following pseudo-algorithm.

Algorithm 1: AdaptSDN Algorithm

Input: Training dataset
$D=(x_1,y_1),(x_2,y_2),...,(x_k,y_k)$, where x_k is the $k-th$ data sample and y_k is the corresponding label
A differentiable cost/loss function $\mathcal{L}(y,H(x))$
Initialize $H_0(x)$ and set the number of iterations M
$H_0(x)=\arg\min_\gamma\sum_{k=1}^{K}\mathcal{L}(y_k,\gamma)$
for $m\leftarrow 1$ *to* M **do**
  Compute $r_{im}=-[\frac{\partial\mathcal{L}(y_k,H(x_k))}{\partial(H(x_k))}]_{H(x)=H_{m-1}(x)}$
  Fit a regression tree $f_m(x)$ *with* M *leaves to the pseudo-residuals* $\{(x_k,r_{km})\}_{k=1}^{K}$
  Compute optimal γ_m *by solving:*
  $\gamma_m=$
  $\arg\min_\gamma\sum_{k=1}^{K}\mathcal{L}(y_k,H_{m-1}(x_m)+\gamma f_m(x_k))$
  Update the model as follows:
  $H_m(x)=H_{m-1}(x)+\gamma_m f_m(x_k)$
end
return $H(x)=sign(H_M(x))$
Mitigation Phase: *Design a non-cooperative game for SDN controllers to obtain vCPU resources, considering attack type and critical applications.*
for *each SDN controller* **do**
  Detect IIoT attack Acquire additional resources based on attack type and critical applications
  Scale up or down virtual resources (CPU, bandwidth, storage) based on the type of attack
end

5. Evaluation of AdaptBoost

This section outlines the evaluation of AdaptBoost, starting with a description of the experimental environment. Subsequently, the experimental results are presented, followed by an assessment of AdaptBoost's performance.

5.1. Experimental Environment

AdaptBoost was implemented using scikit-learn library [17]. Our three modules were implemented as REST applications on top of the SDN controller in the application layer. This allows for easy integration and communication between the modules and the controller. REST, or Representational State Transfer, is an architectural style that provides a standard for creating web services. By using RESTful APIs, the modules can interact with the SDN controller in a standardized way, making it easier to develop, test, and maintain the code. Furthermore, by implementing the modules in the application layer rather than the lower layers of the SDN architecture, it allows for greater flexibility and easier modification in the future. The application layer is responsible for providing high-level functionality to the end user, while the lower layers are responsible for more basic functions such as forwarding and routing. This separation of concerns allows for easier maintenance and development of the system as a whole.

To simulate a realistic scenario, we employed Mininet[18], a widely used SDN emulator that utilizes virtual OpenFlow switches such as OpenVswitch[19] and OpenFlow Switch[20] within containers to create a virtual network environment. Using Mininet, we can create a variety of network topologies and configurations, and simulate traffic flow and network behavior to evaluate the performance and effectiveness of our SDN-based solutions in a controlled and reproducible environment. This allows us to identify potential issues or bottlenecks in the network and test various strategies for optimizing network performance. By utilizing virtual OpenFlow switches within containers, Mininet provides a lightweight and efficient way to emulate network behavior, while also allowing for flexibility and scalability in creating complex network topologies. This makes it a popular tool for researchers and network engineers who want to test and evaluate SDN-based solutions in a simulated environment before deploying them in a production network.

Our test environment comprises four SDN controllers, specifically Floodlight[21], which are responsible for installing OpenFlow rules to detect and block malicious data samples. The use of multiple SDN controllers in our test environment allows for increased scalability and fault tolerance in our network. By distributing the control plane across multiple controllers, we can reduce the likelihood of a central node of failure and improve the overall reliability of the network. Floodlight is a popular open-source SDN controller that provides a modular and extensible platform for building SDN applications. It supports a range of network protocols, including OpenFlow, and provides a RESTful API for easy integration with other network applications.

To monitor and collect network features, we utilized sFlow-RT[22], which is a popular network monitoring tool that supports real-time network telemetry and analysis. sFlow-RT uses the sFlow protocol, which is a network monitoring protocol based on sampling. This enables us to gather real-time network data without causing excessive network traffic. By sampling a small percentage of network traffic, we

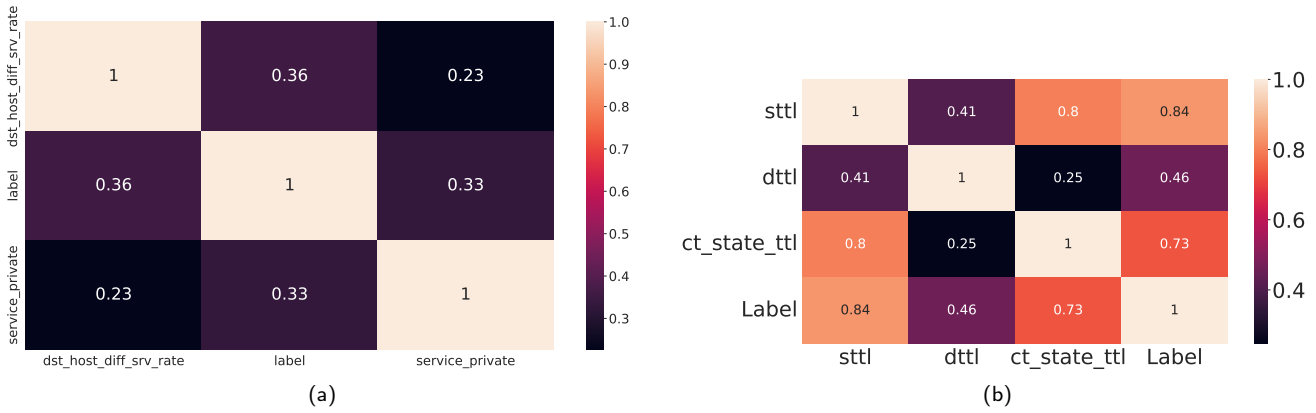

Figure 2: Correlation Feature importance scores on: (a) NSL-KDD, (b) UNSW-NB15.

can gather valuable information about the network's behavior, including flow statistics, packet headers, and application-layer metrics. With sFlow-RT, we can monitor our network and gather a wide range of features, including network traffic patterns, protocol usage, and device performance. This allows us to identify potential security threats, optimize network performance, and troubleshoot issues as they arise. Moreover, sFlow-RT provides a RESTful API that enables easy integration with other network applications, including SDN controllers. This allows us to utilize the data gathered by sFlow-RT to inform the decisions made by our SDN controllers, such as installing OpenFlow rules to block malicious traffic or adjusting network policies to improve performance.

To evaluate the effectiveness of our SDN-based solutions in detecting and mitigating security threats, we simulated attack traffic by employing two widely recognized public network security datasets, specifically UNSW-NB15 and NSL-KDD. It contains over 40 different types of attacks, including DoS, probing, and user-to-root attacks. UNSW-NB15 is a more recent dataset including a wider range of attack types and more complex attack scenarios. It contains different types of attacks, including DDoS attacks, as well as a variety of evasion techniques. By utilizing these datasets to simulate attack traffic in our test environment, we can evaluate the effectiveness of our SDN-based solutions in detecting and mitigating various types of security threats. This allows us to identify potential weaknesses or vulnerabilities in our network and develop strategies to improve its overall security and performance.

5.2. Experimental Results

AdaptSDN's performance evaluation was carried out on two publicly available network security datasets, namely NSL-KDD and UNSW-NB15. NSL-KDD comprises 22 different types of attacks, whereas UNSW-NB15 has 37 types of attacks, including nine new types not found in NSL-KDD. NSL-KDD's input features include 41 variables, three of which are non-numeric/categorical. On the other hand, UNSW-NB15 has 49 input features, four of which are categorical. To prepare the data for analysis, categorical input features in NSL-KDD were encoded into numeric values using the one-hot encoding technique, while UNSW-NB15's input features were encoded using the label encoder method. The input features in both datasets exhibit varying data distributions, which could impact the results. To address this issue, a standardization technique was utilized to rescale the input feature values, standardization was applied to ensure that the input feature values are on a comparable scale. This is important because some input features in these datasets have larger values than others, and without standardization, the larger values could have a disproportionate impact on the model's training process. To standardize the input feature values, the mean and standard deviation of each feature was computed across the entire dataset. Then, each feature value was transformed by subtracting the mean of the feature and dividing it by its standard deviation.

After pre-processing, we applied our feature selection module to eliminate redundant and irrelevant features that could hinder the training process and affect the accuracy of the EL model, particularly in large-scale and high-dimensional data systems. To accomplish this, AdaptSDN utilized non-linear and linear techniques such as correlation and boosting to select the most important features. The correlation measure was applied to both datasets. Feature selection is a critical step in the development of an effective machine-learning model. When working with high-dimensional datasets, it is essential to identify and select the most informative features to avoid overfitting and improve the model's performance.

AdaptSDN employs a feature selection technique to identify important features that can easily distinguish malicious and normal traffic. Linear feature selection techniques involve selecting a subset of input features based on their correlation with the output variable using linear methods such as Pearson correlation coefficient or linear regression. These techniques are effective when there is a linear relationship between the input features and the output variable. Non-linear feature selection techniques, on the other hand, use non-linear methods such as mutual information, chi-squared tests, or decision trees to identify the most informative input features. These techniques are useful when

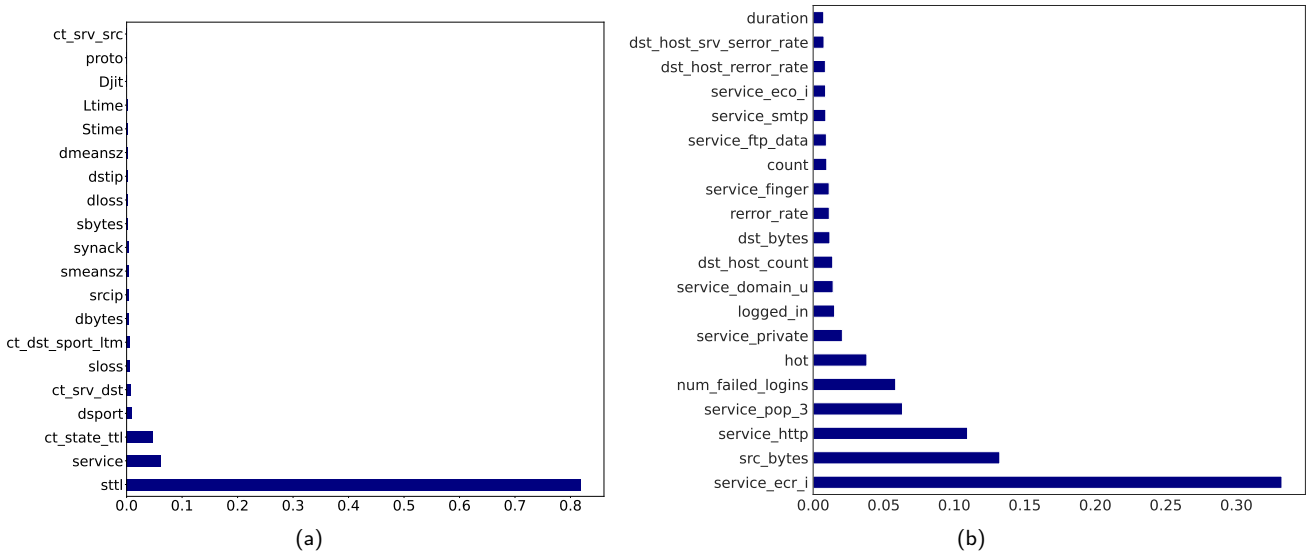

Figure 3: Feature importance scores on (a) UNSW-NB15 and (b) NSL-KDD.

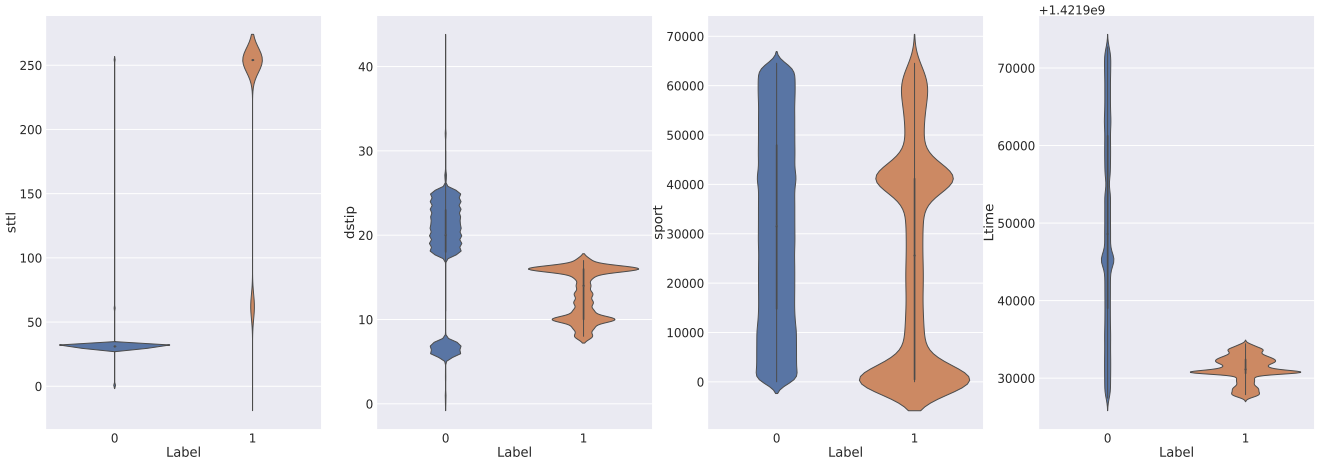

Figure 4: Data distribution of the highest scoring features of $UNSW-NB15$

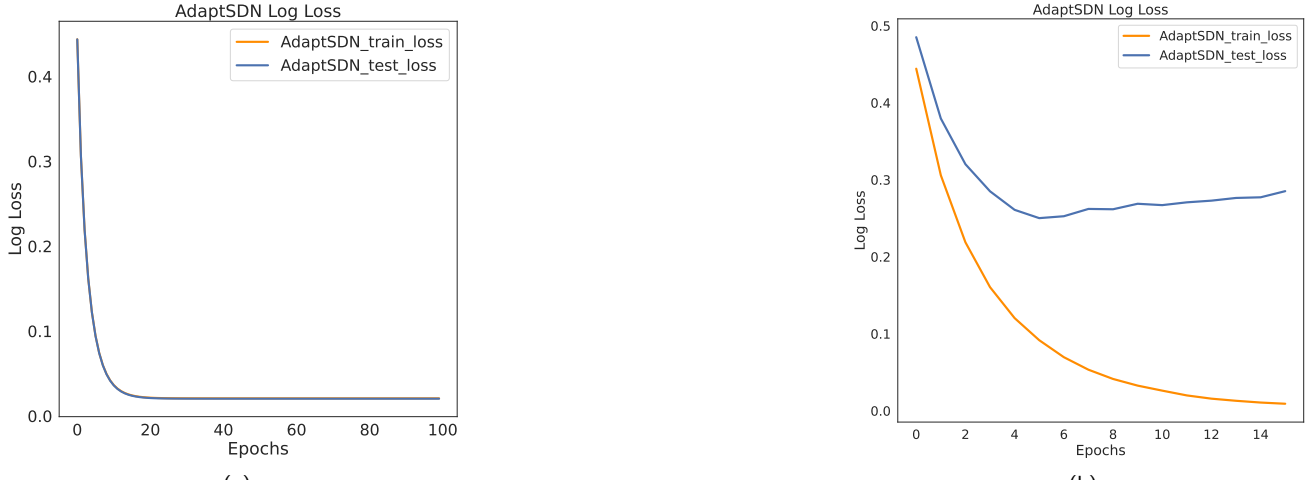

Figure 5: Model loss of AdaptSDN for: (a) UNSW-NB15 dataset, (b) NSL-KDD dataset.

Figure 6: Confusion matrices of AdaptSDN for: (a) UNSW-NB15 dataset, (b) NSL-KDD dataset.

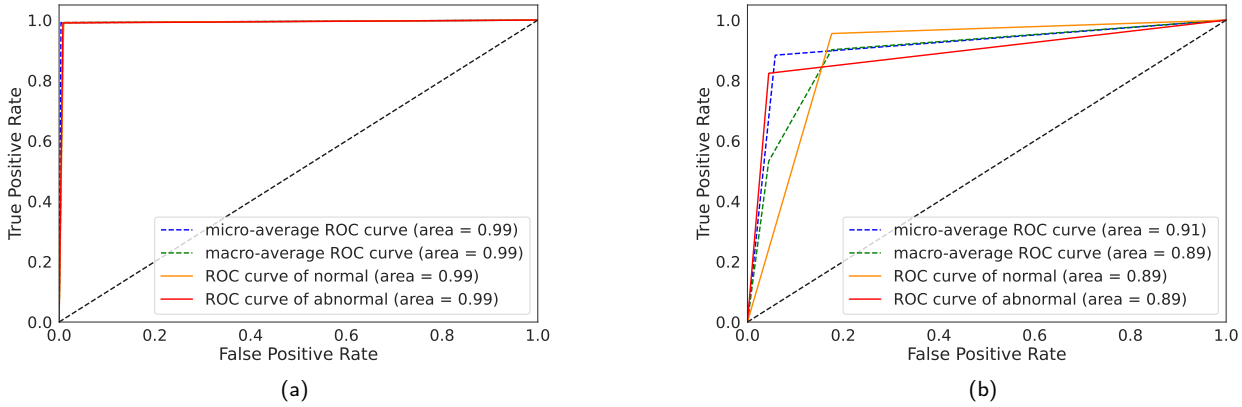

Figure 7: ROC curves of AdaptSDN for: (a) UNSW-NB15 dataset, (b) NSL-KDD dataset.

the correlation between the input and outcome features is non-linear. AdaptSDN combines both linear and non-linear feature selection techniques to achieve a more comprehensive selection of the most promising input features. This approach helps to reduce the dimensionality of the input feature space, which can improve the model's training time and accuracy. By removing redundant or irrelevant input features, the selected features can provide a more accurate representation of the underlying patterns and relationships in the data, leading to better predictive performance.

Figures 2(a) and 2(b) illustrate the most correlated input features with the output feature on top of, respectively, NSL-KDD and UNSW-NB15 datasets. For the NSL-KDD, the results depicted in Figure 2(b) indicate that the promising input features are the ones representing the percentage of connections and having 'SYN' errors and 'dst_host_diff_srv_rate'. These features have a strong positive correlation with the target class, which suggests that they are critical indicators of network security attacks in the NSL-KDD dataset. The 'SYN' error percentage is related to the SYN attack, as the DoS attack. The 'dst_host_diff_srv_rate' feature represents the percentage of connections to different services from the same destination host, which is a crucial feature in detecting network scanning attacks. Therefore, these two features are of great importance in identifying and preventing network security attacks in the NSL-KDD dataset. These features provide insight into the likelihood of a connection being flagged as an attack, making them essential in predicting network attacks accurately. However, in the case of UNSW-NB15, it is worth noting that the time to live values, specifically 'ct_state_ttl' and 'sttl', appear to be the most significant features. These features represent the duration of the connection and are crucial in identifying potential network attacks. By selecting these highly informative features, the AdaptSDN model can efficiently learn the patterns and relationships between the input features and the target class, leading to improved classification accuracy.

To further refine the feature selection process, we used the boosting technique to score the features based on their importance in predicting the target class. Figures 3(a), 3(b), and 4 show the highly scoring features for, respectively, UNSW-NB15 and NSL-KDD datasets. For UNSW-NB15, we observe that the most highly scoring feature is 'sttl', which represents the time to live value of the source. For NSL-KDD, the most highly scoring features are 'service_ecr', 'hot', 'logged_in', and 'dst_host_same_src_port_rate'. These features represent, respectively, the error control scheme of the service, the number of hot indicators, whether

the user is logged in or not, and the rate of connections having the same destination host and source port. The selected features are then used to train and evaluate the EL models. Our feature selection module has successfully identified the most informative features and eliminated the redundant/irrelevant ones for both NSL-KDD and UNSW-NB15 datasets. As a result, we can observe that a significant percentage of features in both datasets are not contributing to making accurate decisions. This indicates that the feature selection module can greatly reduce the computational complexity and improve the efficiency of the EL model.

It is common in machine learning to monitor the performance of a model during training by plotting its learning curves, which typically show the training and validation loss over epochs. In this context, the negative log-likelihood loss is a commonly used metric to evaluate the performance of classification models, such as the EL model. In the case of the NSL-KDD and UNSW-NB15 datasets, Figures 5(a) and 5(b) respectively show the learning curves of the EL model during training and testing. The training loss corresponds to the loss calculated on the training data during each epoch, while the testing loss corresponds to the loss calculated during the validation step on top of the test dataset. The given description suggests that the training and testing losses for both datasets decrease over epochs, implying that the model is acquiring knowledge and enhancing its effectiveness. The minimum training loss represents the best performance achieved on the training data, while the point of stability for the NSL-KDD dataset and almost zero testing loss for the UNSW-NB15 dataset represent the best performance achieved on the validation data.

5.3. Performance Evaluation

We evaluate our proposed framework, AdaptSDN, using various performance metrics, including accuracy, F1 score, and Area Under the ROC Curve (AUC). The AUC is a metric that represents the area under the ROC curve and is used to compare the performance of different models. A higher AUC value indicates better performance. Moreover, confusion matrices are used to evaluate the complete performance of AdaptSDN. A confusion matrix is a table that summarizes the predictions made by a model against the actual labels. From a confusion matrix, various metrics can be calculated, such as Accuracy, Precision, and TPR. AdaptSDN's performance is evaluated using the following metrics:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{27}$$

$$Precision = \frac{TP}{TP + FP} \tag{28}$$

$$DR = TPR = \frac{TP}{TP + FN} \tag{29}$$

Table 1
AdaptSDN Performance.

Dataset	Accuracy	Precision	Recall	F1	Time(s)
UNSW-NB15	99%	99%	99%	99%	37.44
NSL-KDDTest	88%	96%	88%	88%	7.80

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{30}$$

Figures 6(a) and 6(b) display the confusion matrices for the $NSL - KDDTest^{+}$ and UNSW-NB15 datasets, respectively. On the $NSL-KDDTest^{+}$ dataset, AdaptSDN achieves an accuracy of 88%, indicating that the model correctly classified 88% of the instances. The model also obtained high precision, recall, and F1 scores of 96%, 88%, and 88%, respectively, demonstrating the overall effectiveness of AdaptSDN on this dataset. Furthermore, the model achieved this level of performance with a relatively short training time of 7.80 seconds, making it a fast and efficient solution for intrusion detection. Similarly, on the UNSW-NB15 dataset, AdaptSDN achieves excellent performance with accuracy, precision, recall, and F1 score values of 99%, 99%, 99%, and 99%, respectively. The model correctly identified almost all of the instances, demonstrating its ability to handle complex network traffic scenarios. Furthermore, the training time required to achieve this level of performance was only 58.08 seconds, which is a reasonable amount of time given the size and complexity of the dataset.

Table 1 provides AdaptSDN performance metrics on both datasets. The model performs consistently well across different metrics and datasets, highlighting its robustness and reliability for intrusion detection. Overall, the results demonstrate that AdaptSDN is an effective and efficient solution for intrusion detection in the 6G network traffic scenarios. The AUC metric gauges the level of distinctiveness between the output categories and provides a single scalar value to compare different models. A higher AUC score, closer to 1, indicates a better ability to distinguish between abnormal and normal data. Figures 7(a) and 7(b) depict the ROC curves for AdaptSDN on the $NSL - KDDTest^{+}$ and UNSW-NB15 datasets, respectively. The ROC curve for $NSL - KDDTest^{+}$ has an AUC of 0.9, while the curve for UNSW-NB15 has an AUC of 1, indicating excellent performance in both cases. The experiment results demonstrate that AdaptSDN has good performance on both datasets, with a high degree of separability between normal and abnormal data samples.

Besides, we examine a virtual resource orchestrator, denoted by $ROrche_j$, which distributes 500 vCPU resources among four SDN controllers situated at the four MEC nodes. Critical applications that each MEC node supports vary

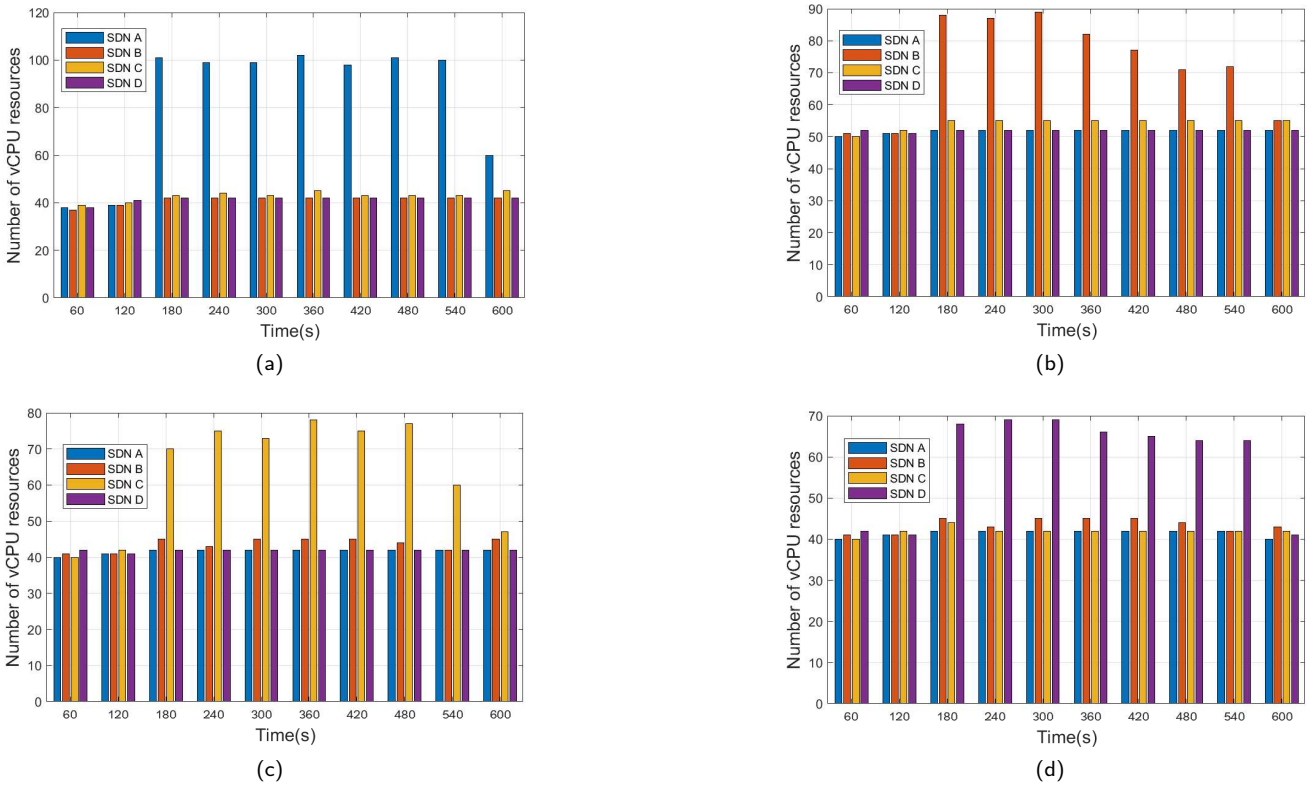

Figure 8: AdaptSDN's Security evaluation (a) SDN A: DDoS attack; (b) SDN B: an analysis attack; (c) SDN C: a fuzzer attack; (d) SDN D: a backdoor attack.

from 10 to 50. A non-cooperative game involving the SDN controllers and the $ROrche_j$ node is initiated when an attack is observed within a MEC domain, and it continues until a NE state is reached. Our game-based strategy is compared to two other approaches: (i) Selfish strategy, where each SDN controller strives to obtain the maximum number of vCPU resources without considering the performance of the centralized $ROrche_j$ or other SDN controllers, and as a result, $ROrche_j$ maximizes the vCPU instances to each SDN controller. (ii) Minimum vCPUs, where the $ROrche_j$ node minimizes the number of vCPUs to each SDN controller.

Figure 8 illustrates the distribution of vCPUs among the SDN controllers for a duration of 600s. We carried out various IIoT attacks, including DDoS, analysis, fuzzers, and backdoors, on specific MEC nodes A, B, C, and D, at time $t = 180$s. Figure8 shows that the number of allocated vCPUs to the corresponding SDN controller increased following the attack, while it remained constant for the other SDN controllers. However, the number of vCPUs assigned to each SDN controller differed, with SDN controller A receiving 100 vCPUs, SDN controller B receiving 75 vCPUs, SDN controller C receiving 70 vCPUs, and SDN controller D receiving 65 vCPUs. This discrepancy was mainly due to the type of attack that was launched in each MEC domain. Our vCPUs allocation strategy, which is based on the attack priority and the number of attackers (as indicated in Equation 18), strongly influenced the vCPUs assignment. Additionally, our results revealed that DDoS attacks required more vCPUs compared to other types of attacks. Overall, our approach ensured that the compromised MEC nodes received the necessary vCPUs resources while maintaining a stable minimum allocation of vCPUs to the other MEC nodes to meet the requirements of their respective applications.

Figure 9 shows a performance comparison of our scheme, Selfish, and Min vCPU in assigning vCPUs to the SDN controller B for 600 seconds. As demonstrated in Figure 9(a), our scheme may experience fluctuations in the allocated vCPUs, while Selfish and Min vCPU schemes exhibit mostly constant allocation throughout the period. This is because the number of critical applications can fluctuate, and unexpected IIoT attacks may occur. To address this issue, our scheme is designed to dynamically adjust the allocated vCPUs. Figure 9(b) illustrates how our scheme responds to an increase in the number of critical applications by increasing the allocation of vCPUs. In contrast, the number of vCPUs assigned is constant for both Min vCPU and Selfish schemes, irrespective of the number of running applications. When assigning vCPU instances to MECs, our scheme considers the number of critical applications, as shown in Equation 19. In Figure 9(c), a comparison of the three schemes is presented on top of two different attacks

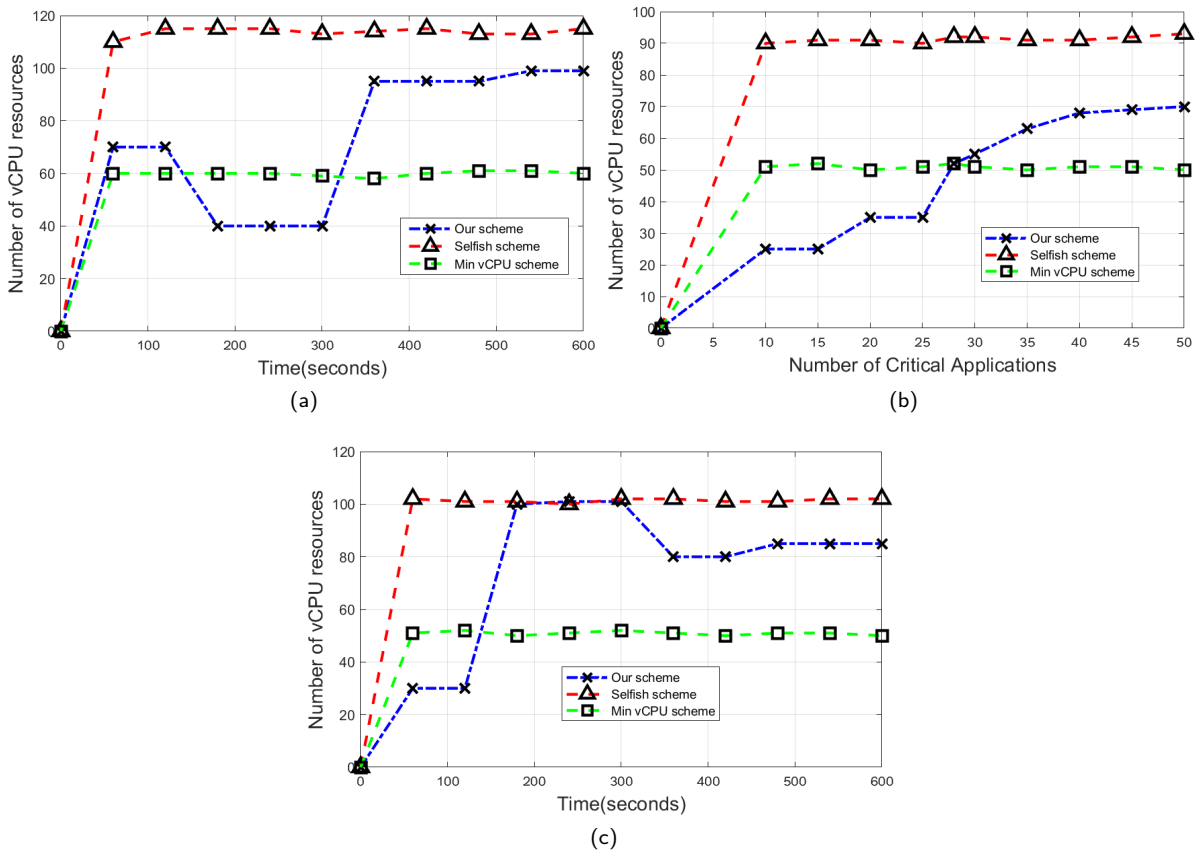

Figure 9: Comparison between Min vCPUs, selfish, and AdaptSDN approaches.

DDoS and analysis attacks at t=180s and at t=480s, respectively. At t=180s, the number of assigned vCPUs reaches its highest point of 100 before dropping to 80 vCPUs due to the DDoS attack. After the DDoS attack, the number of vCPUs instances recovers and achieves 85 (t=480s), due to the analysis attack. The Min vCPU and Selfish schemes exhibit consistent assignment behavior, notably by not factoring in IIoT attacks or critical applications during vCPU allocation. While the Selfish scheme can allocate sufficient vCPUs to address IIoT attacks, a substantial portion of the allocated vCPUs remains unused, particularly in the absence of attacks and critical applications. This can negatively impact the overall system performance. In contrast, our scheme guarantees a dynamic and efficient allocation of virtual resources to SDN controllers when an attack is detected. It considers both the priority of the attack, the critical applications of the MECs, and the performance of the whole system, resulting in optimal resource allocation. In the following sections, we compare AdaptSDN with recent AI models on these datasets. By doing so, we can evaluate the effectiveness of AdaptSDN in detecting network intrusion and assess its competitiveness compared to other approaches.

5.4. Comparative Analysis

In this section, a comparison is made between the results obtained by AdaptSDN and AI models using both datasets. AdaptSDN is a type of AI model designed for intrusion detection, and it has been evaluated on both NSL-KDD and UNSW-NB15 datasets, along with other AI models. The performance metrics are presented in Tables 2 and 3. These metrics provide a quantitative measure of the performance of different AI models on the given datasets, allowing researchers to compare and evaluate the effectiveness of different models for intrusion detection.

Tavallaee et al. [23] conducted a detailed analysis of five ML models (*i.e.*, SVM, RF, J48, NB, and MLP). In addition to these models, we compare AdaptSDN's performance with several recent DL models. In [24], the authors propose a new approach that uses CNNs to detect intrusions at the character level, which can capture more fine-grained information about the traffic data. In [25], the authors propose a CNN-based IDS that uses network traffic data as input to learn a set of discriminative features that can distinguish between normal and malicious traffic. The system is composed of three main components: preprocessing, feature extraction, and classification. The preprocessing step involves converting the raw network traffic data into a suitable format for the feature extraction step. The feature extraction step uses a CNN to learn a set of features that can represent the input data in a discriminative manner. The authors modified the (ResNet50, GoogLeNet) architectures, which is a type of CNN, for the feature extraction step. They used a variant

of SVM known as the C-Support Vector Classification (C-SVC) algorithm for classification.

In [26], the authors use a long short-term memory (LSTM) RNN to learn a temporal representation of network traffic data. The proposed system consists of two main components: feature extraction and classification. In the feature extraction step, the LSTM RNN is used to learn a temporal representation of the input data. In the classification step, a softmax layer is used to classify the input data as either normal or anomalous. In [27], the authors propose a novel scheme (Filter) that focuses on feature selection and proposes a filter-based approach to identify relevant features for intrusion detection. In [28], the authors propose a deep learning approach that combines a sparse autoencoder with SVMs to improve the accuracy of IDSs. In [29], the authors propose an adaptive ensemble machine learning model that combines multiple classifiers, to detect network intrusions. The model adjusts the weights of each classifier dynamically to improve its performance. For the UNSW-NB15 dataset, we compared the performance of AdaptSDN with recent state-of-the-art AI models, including SSL [30], RF [31], Feed [32], OGM [33], and MHMM [34]. AdaptSDN demonstrates superior performance compared to the current solutions, achieving accuracy and F1 score that are 12% higher on the NSL-KDD dataset, with the highest accuracy and F1 score of 88%. On the UNSW-NB15 dataset, AdaptSDN outperforms the other schemes by providing accuracy and a true positive rate of 99%, with a remarkably short training time of only 37.44 seconds. Notably, AdaptSDN achieves these results with reduced computational complexity, with a training time of only 7.80 seconds on the NSL-KDDTest+ dataset, significantly lower than the reported training times for other models. These findings demonstrate the potential for AdaptSDN to be deployed practically in real-world scenarios, highlighting its effectiveness in detecting network security attacks.

Table 2
Performance metrics of AdaptSDN and AI models on $NSL-KDDTest^{+}$

Methods	Accuracy	Precision	Recall	F1	Time (second)
J48	0.81	NA	NA	NA	NA
Naive Bayes	0.76	NA	NA	NA	NA
Random Forest	0.80	NA	NA	NA	NA
Multi-layer Perceptron	0.77	NA	NA	NA	NA
Support Vector Machine	0.70	NA	NA	NA	NA
CharCNN	0.85	0.91	0.81	0.86	NA
ResNet50	0.79	0.91	0.69	0.79	NA
GoogleNet	0.77	0.91	0.65	0.76	NA
Deep Neural Networks	0.75	0.83	0.75	0.74	NA
recurrent neural networks	0.83	NA	0.83	NA	5516
Filter	0.78	NA	0.78	NA	NA
Autoencoder	0.84	0.96	0.76	0.85	673.031
Adaboost	0.85	0.86	0.85	0.84	NA
AdaptSDN	**0.88**	**0.96**	**0.88**	**0.88**	**7.80**

Table 3
Performance metrics of AdaptSDN and AI models on UNSW-NB15

Schemes	Accuracy	TPR	Time (second)
SSL	0.86	0.85	NA
RF	0.93	0.92	NA
Feed	0.92	0.91	NA
OGM	0.95	0.94	NA
MHMM	0.96	0.95	NA
AdaptSDN	**0.99**	**0.99**	**37.44**

6. Conclusion

In this paper, we proposed a novel framework that includes three modules: (1) A novel module for data collection and optimized feature selection to achieve two main objectives: reducing computational complexity and improving detection performance; (2) An SDN-based lightweight adaptive boosting module that uses advanced boosting EL techniques to dynamically adjust weights and to effectively identify and respond to IIoT attacks in real-time; and (3) a zero-touch resources provisioning module that employs a non-cooperative game theory approach. We evaluated our proposed framework's performance, assessing accuracy, F1 score, and computational complexity. We compared our framework's performance to state-of-the-art solutions and utilized real-world attack scenarios in our experiments. The numerical results confirm that AdaptSDN has the potential to enable secure and reliable IIoT applications in 6G and beyond, meeting the stringent service requirements of the new emerging applications.

References

[1] A. Mohammed and R. Kora, "A comprehensive review on ensemble deep learning: Opportunities and challenges," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 2, pp. 757–774, 2023.

[2] R. S. C. B. e. a. Raddo, T.R., "Transition technologies towards 6g networks." *J Wireless Com Network*, vol. 100, 2021.

[3] S. H. A. Kazmi, R. Hassan, F. Qamar, K. Nisar, and A. A. A. Ibrahim, "Security concepts in emerging 6g communication: Threats, countermeasures, authentication techniques and research directions," *Symmetry*, vol. 15, no. 6, 2023. [Online]. Available: https://www.mdpi.com/2073-8994/15/6/1147

[4] B. B. Yousif, E. E. Elsayed, and M. M. Alzalabani, "Atmospheric turbulence mitigation using spatial mode multiplexing and modified pulse position modulation in hybrid rf/fso orbital-angular-momentum multiplexed based on mimo wireless communications system," *Optics Communications*, vol. 436, pp. 197–208, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0030401818310836

[5] P. K. R. Maddikunta, Q.-V. Pham, P. B, N. Deepa, K. Dev, T. R. Gadekallu, R. Ruby, and M. Liyanage, "Industry 5.0: A survey on enabling technologies and potential applications," *Journal of Industrial Information Integration*, vol. 26, p. 100257, 2022.

[6] A. Hazra, P. K. Donta, T. Amgoth, and S. Dustdar, "Cooperative transmission scheduling and computation offloading with collaboration of fog and cloud for industrial iot applications," *IEEE Internet of Things Journal*, vol. 10, no. 5, pp. 3944–3953, 2023.

[7] B. B. Yousif and E. E. Elsayed, "Performance enhancement of an orbital-angular-momentum-multiplexed free-space optical link under atmospheric turbulence effects using spatial-mode multiplexing and hybrid diversity based on adaptive mimo equalization," *IEEE Access*, vol. 7, pp. 84 401–84 412, 2019.

[8] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 5, pp. 649–659, 2008.

[9] A.-C. Enache and V. V. Patriciu, "Intrusions detection based on support vector machine optimized with swarm intelligence," in *2014 IEEE 9th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*, 2014, pp. 153–158.

[10] J. Shun and H. A. Malki, "Network intrusion detection system using neural networks," in *2008 Fourth International Conference on Natural Computation*, vol. 5, 2008, pp. 242–246.

[11] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for iot security based on learning techniques," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2671–2701, 2019.

[12] X. Zhou, W. Liang, W. Li, K. Yan, S. Shimizu, and K. I.-K. Wang, "Hierarchical adversarial attacks against graph-neural-network-based iot network intrusion detection system," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9310–9319, 2022.

[13] C. Liu, Z. Gu, and J. Wang, "A hybrid intrusion detection system based on scalable k-means+ random forest and deep learning," *IEEE Access*, vol. 9, pp. 75 729–75 740, 2021.

[14] Z. A. E. Houda, B. Brik, and L. Khoukhi, ""why should i trust your ids?": An explainable deep learning framework for intrusion detection systems in internet of things networks," *IEEE Open Journal of the Communications Society*, vol. 3, pp. 1164–1176, 2022.

[15] L. Wang and G.-S. G. Kuo, "Mathematical modeling for network selection in heterogeneous wireless networks — a tutorial," *IEEE Communications Surveys Tutorials*, vol. 15, no. 1, pp. 271–292, 2013.

[16] J. B. Rosen, "Existence and uniqueness of equilibrium points for concave n-person games," *Wiley, Econometric Society*, vol. 33, no. 3, p. 520–34, 1965.

[17] "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. 85, p. 2825–2830, 2011.

[18] "Mininet." [Online]. Available: http://mininet.org

[19] "Openvswicth." [Online]. Available: https://www.openvswitch.org/

[20] "Openflow switch." [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

[21] "Floodlight openflow controller." [Online]. Available: https://floodlight.atlassian.net/wiki/spaces/HOME/overview

[22] "sflow-rt." [Online]. Available: http://www.sflow-rt.com

[23] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6.

[24] S. Z. Lin, Y. Shi, and Z. Xue, "Character-level intrusion detection based on convolutional neural networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8.

[25] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," in *Neural Information Processing*. Springer International Publishing, 2017, pp. 858–866.

[26] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21 954–21 961, 2017.

[27] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 2986–2998, 2016.

[28] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with svm for network intrusion detection," *IEEE Access*, vol. 6, pp. 52 843–52 856, 2018.

[29] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, "An adaptive ensemble machine learning model for intrusion detection," *IEEE Access*, vol. 7, pp. 82 512–82 521, 2019.

[30] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Information Sciences*, vol. 378, pp. 484–497, 2017.

[31] K. Singh, S. C. Guntuku, A. Thakur, and C. Hota, "Big data analytics framework for peer-to-peer botnet detection using random forests," *Information Sciences*, vol. 278, pp. 488–497, 2014.

[32] C. D. McDermott and A. Petrovski, "Investigation of computational intelligence techniques for intrusion detection in wireless sensor networks." *International journal of computer networks and communications*, vol. 9, pp. 45–56, 2017.

[33] N. Moustafa, G. Misra, and J. Slay, "Generalized outlier gaussian mixture technique based on automated association features for simulating and detecting web application attacks," *IEEE Transactions on Sustainable Computing*, pp. 1–1, 2018.

[34] N. Moustafa, E. Adi, B. Turnbull, and J. Hu, "A new threat intelligence scheme for safeguarding industry 4.0 systems," *IEEE Access*, vol. 6, pp. 32 910–32 924, 2018.