Evaluating MLP and Autoencoder Models for Zero-Day Attack Detection in 6G Networks

Maria Gabriela Lima Damasceno, Caio Bruno Bezerra de Souza, Andson Marreiros Balieiro

[1]Centro de Informática (CIn) – Universidade Federal de Pernambuco (UFPE)
Recife – PE – Brazil

{mgld, cbbs, amb4}@cin.ufpe.br

Abstract. *Sixth Generation (6G) networks promise a deep integration with Artificial Intelligence (AI), enabling intelligent and efficient communication systems. However, this evolution also introduces new cybersecurity threats, including Zero-Day attacks that exploit previously unknown system vulnerabilities. This paper evaluates two widely adopted models, the Multilayer Perceptron (MLP) and the Autoencoder, for attack detection, with a particular focus on Zero-Day attacks. Using a dataset collected from a real 5G network, the study evaluates each model not only in terms of detection performance but also with respect to computational resource consumption. The results indicate that the MLP model outperforms the Autoencoder in both overall classification accuracy and zero-day attack detection, albeit at the cost of higher computational resource usage.*

1. Introduction

The Sixth Generation of Mobile Networks (6G), expected to be launched by 2030 [Kianpisheh and Taleb 2024], aims to deliver ultra-high data rates, near-zero latency, reduced energy consumption, zero-touch operation, and support for advanced services such as digital twins, holographic communications, and the metaverse. To fulfill these ambitious goals, a wide array of enabling technologies are expected to converge, including continuum cloud computing, terrestrial and aerial communications, integrated communication and sensing, resource and service virtualization, network slicing, and programmability. While this convergence is promising, it also increases network complexity, broadens the attack surface, and poses significant challenges to secure network management and operation.

An important advancement in 6G will be its native integration of Artificial Intelligence (AI) [Ziegler et al. 2020], enabling intelligent network management, automation, optimization, and security across all layers of the architecture. However, this evolution in network operation also brings increased cyber threats [De Azambuja et al. 2023], further exacerbated by advances in AI and processing capabilities that empower attackers to design new types of attacks and exploit emerging vulnerabilities. Among these threats, Zero-Day attacks stand out as a significant risk, since they can render conventional defense mechanisms ineffective by exploiting previously unknown vulnerabilities or taking advantage of attacks that the network has not yet identified [Guo 2023].

Detecting and mitigating cyber-attacks in real time remains a critical challenge in 6G networks. In this context, several AI-based solutions have been proposed, such as those in [Benzaïd et al. 2024], [Kianpisheh and Taleb 2024], [Ali et al. 2022], [Hamid et al. 2022], and [Sarhan et al. 2023]. While these studies offer valuable insights and results, a notable gap persists concerning Zero-Day attacks. For instance, works like [Benzaïd et al. 2024] and [Kianpisheh and Taleb 2024] are limited to known attacks, failing to address the Zero-Day threat. Although other studies do target Zero-Day scenarios, such as [Ali et al. 2022], [Hamid et al. 2022], and [Sarhan et al. 2023], they typically focus only on classification accuracy, neglecting the analysis of resource consumption. This omission is significant, given that sustainability is a core pillar of 6G networks [Imoize et al. 2021], and many detection mechanisms may need to operate on resource-constrained devices. Furthermore, even the accuracy evaluations in some studies, such as [Ali et al. 2022] and [Hamid et al. 2022], are based solely on overall metrics that combine results from known and unknown attacks. This approach may obscure poor performance in detecting Zero-Day attacks, as the high accuracy for known threats can mask deficiencies in identifying new, unseen ones. Moreover, many studies are not based on datasets collected from real mobile networks, which can limit the generalization of their findings to realistic 6G scenarios.

This paper addresses these gaps by evaluating Zero-Day attack detection solutions using a dataset collected from a real 5G network, the 5G Network Intrusion Detection Dataset (5G-NIDD) [Samarakoon et al. 2022]. We consider two widely used models for zero-day attack detection: a Multi-Layer Perceptron (MLP) and an Autoencoder [Ahmad et al. 2023], both applied to classify network traffic as either benign or malicious. In addition to overall accuracy, we assess these models in terms of computational resource consumption, their specific performance in detecting Zero-Day attacks, and time required for both training and testing phases. We also present a methodology that ensures both models are trained without prior exposure to a specific type of attack, treated as the Zero-Day attack, which is only introduced during the testing phase. This methodology enables a proper evaluation of the models' ability to generalize from known attacks to unknown ones, thereby reflecting their effectiveness in real Zero-Day attack scenarios. Results show that the MLP model achieved higher accuracy than the Autoencoder, both in overall classification and in the detection of Zero-Day attacks, albeit at the cost of increased computational resource consumption.

The remainder of this paper is structured as follows. Section 2 discusses the related work. Section 3 lists the AI applications in 6G networks and the functionality of the MLP Machine Learning (ML) technique. Section 4 describes the data processing, model construction, and performance evaluation. Section 5 presents the results of the model evaluation and resource allocation. Finally, Section 6 concludes the paper and suggests future research.

2. Related Work

Artificial intelligence-based solutions for security challenges in 6G networks have been widely explored in the literature. For example, in [Kianpisheh and Taleb 2024], the authors adopt collaborative Federated Learning (FL) to detect Distributed Denial of Service (DDoS) attacks, enabling intelligent services at the core, edge, and end device levels. The proposed solution is evaluated using CICDDoS 2019 data set, which simulates realistic traffic, including legitimate and DDoS attack flows, over various protocols such as HTTP, FTP, and SSH in a wired network. Similarly, FL is employed in [Benzaïd et al. 2024] for

DDoS attack detection in Open Radio Access Network (O-RAN)-based 5G and Beyond (B5G) networks. The authors incorporate Continual Learning (CL) into FL to address the catastrophic forgetting (CF) problem, wherein previously learned knowledge is lost as new attack patterns are sequentially acquired from a data stream. They utilize a subset of DDoS attacks from 5G-NIDD dataset to demonstrate the performance gains achieved through the integration of CL with FL. Although they present important insights and results, both works focus exclusively on known attacks and do not consider the time and resource consumption involved in deploying the proposed solutions.

Regarding zero-Day attacks, a sandbox-based solution for Zero-Day attack detection and prevention in Software-Defined Networks (SDN) is proposed in [Al-Rushdan et al. 2019]. The approach analyzes memory dumps, files created by malware, and registry information to make appropriate decisions, such as isolating the client or blocking all incoming traffic. This mechanism aims to protect both the SDN controller and the client node from Zero-Day threats. The authors use the Mininet simulator to evaluate the solution under different malware sizes, reporting a detection rate of 97.78%. The authors in [Sameera and Shashi 2020] propose a deep transductive transfer learning framework for Zero-Day attack detection. They utilize two datasets: NSL-KDD, which includes 147,907 packet-based samples across four attack categories (DoS, Probe, R2L, and U2R) and CIDD, a cloud intrusion detection dataset comprising 5,274 samples of normal and DoS attack instances. The proposed framework achieves an accuracy of 91.75% on the NSL-KDD dataset and 78.85% on the CIDD dataset. In contrast, [Redino et al. 2022] employ the MAWI, NCCDC, DIF, and Codex datasets and propose a double Autoencoder (AE) approach. In the first workflow, the AE is trained exclusively with benign network traffic to identify potentially malicious flows. In the second, it is trained using only known attacks to detect new attacks, known as Zero-Day attacks. Our research, when using the Autoencoder model, follows a similar workflow to this second AE but applies it to a Mobile Networks Dataset.

Comparative studies on different solutions for zero-day attack detection have also been presented in the literature, such as in [Ali et al. 2022], [Hamid et al. 2022], [Redino et al. 2022] and [Lu et al. 2024]. However, these works often assess only the overall accuracy of the models, without specifically evaluating their effectiveness in detecting zero-day attacks, i.e., samples belonging to classes not present during the training phase. In contrast, although [Sarhan et al. 2023] analyzes metrics for zero-day attack detection, it neglects the computational consumption and does not rely on datasets collected from real mobile networks. Highlighting the need for computationally efficient solutions in resource-constrained IoT environments, the authors in [Agbedanu et al. 2025] propose an adaptive and memory-efficient algorithm for detecting Zero-Day attacks in IoT systems. Their solution achieves high detection rates for simulated zero-day attacks, with overall accuracy exceeding 99% and low computational overhead.

As previously discussed, existing studies explore the use of Artificial Intelligence (AI) and Machine Learning (ML) approaches for attack detection in 6G networks, but many focus solely on known attacks. Among those targeting zero-day attacks, few take into account the computational or time resource consumption of the proposed solutions, an essential factor in resource-constrained IoT environments and in light of sustainability, which is a key pillar of 6G networks. Furthermore, some of these studies are not even based on mobile network environments, limiting their applicability and generalization of their findings. To fill these gaps, we evaluate two widely-adopted models for detecting zero-day attacks in 6G networks. Our analysis leverages a dataset that includes different attacks collected from a real 5G network [Samarakoon et al. 2022], and examines performance metrics for both overall and zero-day attacks. Additionally, we assess the computational and time resource consumption of each model, providing a comprehensive evaluation of their effectiveness and feasibility in practical scenarios. Table 1 summarizes the related studies in comparison to the proposed one.

Table 1. Comparison between related works

Reference	6G/B5G Focus	Model(s)/Approach	Zero-Day Detection	Mobile Network Dataset	Resource Evaluation
[Kianpisheh and Taleb 2024]	Yes	Federated Learning	No	Yes	No
[Benzaïd et al. 2024]	Yes	Federated + Continual Learning	No	Yes	No
[Ali et al. 2022]	No	Multiple ML models (comparison)	Partial*	No	No
[Hamid et al. 2022]	No	Multiple ML models (comparison)	Partial*	No	No
[Lu et al. 2024]	No	Transfer Learning	Partial*	No	No
[Sarhan et al. 2023]	No	Zero-Day Detection Models	Yes	No	No
[Al-Rushdan et al. 2019]	No	Cuckoo Sandbox	Yes	No	No
[Sameera and Shashi 2020]	No	Deep transductive transfer learning	Yes	No	No
[Guo 2023]	Yes	ML-based approaches (review)	Yes (Conceptual)	No	No
[Agbedanu et al. 2025]	No	STM and LTM	Yes	Yes	Yes
[Redino et al. 2022]	No	Autoencoder	Yes	No	No
This work	**Yes**	**MLP + Autoencoder**	**Yes**	**Yes**	**Yes**

*Mentions Zero-Day but lacks specific metrics.

3. Background

3.1. Artificial Intelligence in 6G Networks

The sixth generation of mobile networks is expected to be born with native support for AI/ML techniques, i.e., the integration of AI/ML across various layers, segments, and functions of the network is considered essential in 6G networks [Ziegler et al. 2020]. This embedding aims to enable efficient, flexible, secure, and zero-touch operation and management of 6G networks. Several studies, such as [Wu et al. 2021] [Jain et al. 2022], have already proposed the design of a new intelligent plan with emphasis on the orchestration, management and operation of end-to-end AI workflows. In parallel with network operation and management challenges, security issues are also expected to escalate in 6G networks due to the diversity of services, devices, and technologies envisioned to coexist. For instance, the growing number of Internet of Things (IoT) devices—often constrained in memory and power to support robust security mechanisms—makes ensuring network security challenging [Zhang et al. 2021]. In this context, the use of Artificial Intelligence becomes essential to provide security and trust services in 6G networks.

3.2. Multi-Layer Perceptron (MLP)

The Multi-Layer Perceptron (MLP) is a type of artificial neural network classified under Deep Neural Networks (DNNs). An MLP consists of multiple layers of neurons, including an input layer, one or more hidden layers, and an output layer. Neurons in each layer are fully connected to those in the subsequent layer through adjustable weights. As one of the main approaches in supervised machine learning, its architecture enables it to learn complex patterns and perform tasks such as classification, regression, and pattern recognition [Taud and Mas 2018]. MLPs are widely employed for anomaly detection

[Teoh et al. 2018], including the identification of malicious traffic such as malware and Zero-Day attacks [Abri et al. 2019]. By applying optimization algorithms such as back-propagation [Lawrence and Giles 2000], an MLP can iteratively enhance its anomaly detection capabilities. In an MLP model, each neuron in a given layer l computes an output based on input weights W, input features h and a bias term b. This operation can be defined as shown in Eq. 1 [Taud and Mas 2018].

$$h^{(l)} = f\left(W^{(l)}h^{(l-1)} + b^{(l)}\right) \tag{1}$$

where:

$W^{(l)}$ and $b^{(l)}$ are the weights and bias of the l-th layer,

f is a nonlinear activation function (e.g., Rectified linear unit, and sigmoid),

$h^{(l)}$ is the output of the l-th layer,

$h^{(l-1)}$ the output of the previous layer. Serves as the input to the current layer,

$h^{(0)} = x$ is the input vector.

3.3. Autoencoders

Autoencoders are a type of artificial neural network trained to replicate their input at the output [Goodfellow et al. 2016], thereby learning compact and informative representations of the input data. Because the input and output are identical [Hindy et al. 2020], autoencoders are considered self-supervised learning techniques. They are commonly used for tasks such as dimensionality reduction and anomaly detection. Autoencoders consist of two main components: the encoder and the decoder. The encoder maps the input data into a compressed representation, focusing on the most relevant features and revealing the underlying patterns of the data. The decoder then reconstructs the original input from this compressed representation, attempting to replicate the input data as closely as possible, thereby retaining the essential information. This architecture is particularly interesting for Zero-Day attack detection, as it can capture complex and previously unseen attack patterns [Mohamed et al. 2025]. The Mean Squared Error (MSE) is used to measure the difference between the input and its reconstruction [Yoshioka et al. 2024]. Higher MSE values indicate greater discrepancy, which may suggest that the sample is an anomaly. In Eq. 2, x_i denotes the original value and $\hat{x}_i$ the value reconstructed by the autoencoder for the i-th feature of the sample.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \tag{2}$$

4. Methodology

This section presents the main aspects related to the dataset, the preprocessing stage, and the organization of samples to define Zero-Day attacks. Additionally, it describes the configurations and metrics used to evaluate the models in terms of attack detection performance, as well as computational and time resource consumption.

4.1. Data Pre-Processing

The 5G Network Intrusion Detection Dataset (5G-NIDD) [Samarakoon et al. 2022] is a dataset that addresses the issue of security in 5G networks, with an emphasis on real-time attack detection. The data is labeled as benign and malicious and comprises attacks as Hypertext Transfer Protocol (HTTP) Flood, Internet Control Message Protocol (ICMP) Flood, Synchronize (SYN) Flood, Synchronize (SYN) Scan, Slowrate Denial of Service (DoS), Transmission Control Protocol (TCP) Connect Scan, User Datagram Protocol (UDP) Flood, and User Datagram Protocol (UDP) Scan.

In this study, samples that contained missing values (null) were removed from the dataset to ensure that the models were trained with complete and valid data. Furthermore, the dataset was balanced in Python using the Synthetic Minority Over-sampling Technique (SMOTE) library. It creates synthetic samples of the minority class to balance the distribution of the classes. This procedure aims to prevent the models from becoming biased toward one class over another. To train and evaluate the MLP and Autoencoders models, the data was divided into two groups, 80% for training and 20% for testing, with 243,178 samples to be used.

4.2. Zero-Day Processing

To incorporate Zero-Day attacks into the model's detection, a different attack type from the dataset was selected as the Zero-Day in each test iteration. The selected attack was excluded from the training set during the data partitioning process to evaluate how the model would perform when encountering previously unseen attacks during the testing phase. By removing the Zero-Day attack samples from the training set and including them only in the test set, we ensured that the model had no prior exposure to these specific attacks. For instance, in one scenario, the model was not trained on ICMP Flood attacks but was required to classify them during testing. Fig. 1 illustrates this process of defining zero-day attacks using the dataset 5G-NIDD [Samarakoon et al. 2022]. During the data Pre-Processing stage, a specific type of attack is purposely removed from the training set (D_{train}) and reserved exclusively for the test set (D_{test}). This will be the attack referred to as zero-day in the evaluation.

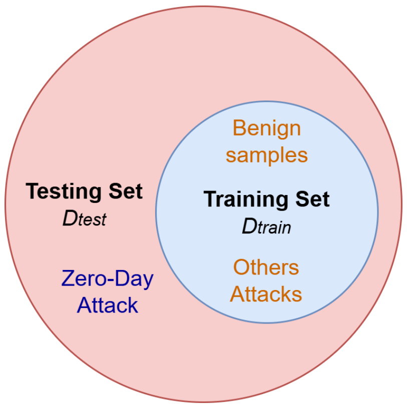

Figure 1. Zero-Day Processing

The training set includes only benign samples and known attack types, enabling the model to learn predefined normal and malicious patterns, excluding the zero-day attacks. During the testing phase, the model is evaluated using benign samples, known attack samples, and samples containing the unseen zero-day attack. The goal is to assess the model's generalization capability and its effectiveness in identifying novel threats, an essential requirement for 6G networks, where new attack types are expected to emerge.

4.3. Model Flow

Fig. 2 illustrates the flow used by both models to evaluate the detection of Zero-Day attacks. Firstly, the process begins with the input of a dataset (D), which contains labeled samples of various types of attacks. After that, one of the attack classes is defined as the representative of the Zero-Day scenario, such as the ICMPFlood attack. Then, the class defined as Zero-Day is intentionally removed from the training process.

When the (D_{train}) and (D_{test}) sets are initialized, 80% of the samples are selected for training in general, but within the training set, there is a reallocation of the samples that belong to the Zero-Day class to the test set. In other words, the model is tested with 20% of the total samples with the addition of all the samples from the Zero-Day class.

After partitioning the samples, the model is trained with the (D_{train}) set, which does not contain any of the Zero-Day samples. Subsequently, the model is evaluated using the (D_{test}) set, which contains the previously unpublished samples. Finally, the model's performance is assessed using metrics for attack detection such as accuracy, precision, recall, and F1-score, as well as the Zero-Day Detection Rate (ZD_{DR}). Additionally, computational resource metrics such as CPU usage, memory consumption, and execution time are also considered.

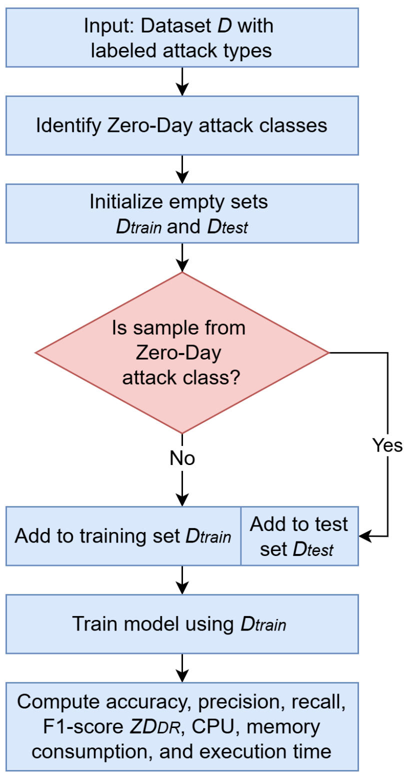
Input: Dataset *D* with labeled attack types
Identify Zero-Day attack classes
Initialize empty sets *Dtrain* and *Dtest*
Is sample from Zero-Day attack class?
No
Yes
Add to training set *Dtrain*
Add to test set *Dtest*
Train model using *Dtrain*
Compute accuracy, precision, recall, F1-score *ZDDR*, CPU, memory consumption, and execution time

Figure 2. Model Flow

4.4. Model Configuration

Zero-day attack detection models were built using MLP and Autoencoder algorithms. The models used throughout this research are already well-known and well-established in the literature, being widely used in evaluating anomaly detection as seen in [Ali et al. 2022], [Hamid et al. 2022] and [Guo 2023]. However, the main contribution of this work is the evaluation methodology that aims to include both the analysis of the model's performance from the point of view of commonly evaluated metrics, such as Accuracy, F1-Score, etc., as well as metrics that are still little explored, such as those related to the consumption of computational resources. The input layer of the models receives data based on the number of features in the training set. Similar to [Samarakoon et al. 2022], this paper takes into account 24 features as inputs to the models.

The MLP model consists of three dense layers with 128, 64, and 32 neurons, respectively. All neurons use the Rectified Linear Unit (ReLU) activation function, and L2 regularization is applied to penalize large weights and prevent overfitting. Between the dense layers, dropout layers with a rate of 40% are included to randomly deactivate 40% of the connections during training, encouraging the network to learn more generalizable representations. The output layer is a dense layer with a single neuron and a sigmoid activation function, suitable for binary classification tasks, as it maps the output to a value between 0 and 1. In this context, it is used to classify samples as either malicious attacks or benign traffic. The MLP model was trained for 10 epochs using a batch size of 128, the Adam optimizer, and a learning rate of 0.001. Table 2 summarizes the MLP model configuration, which was set following the work in [Kianpisheh and Taleb 2024].

Table 2. MLP Model Architecture

Layer	Type	Neurons	Activation	Additional Features
1	Dense	128	ReLU	L2 Regularization
2	Dropout	–	–	40% dropout
3	Dense	64	ReLU	L2 Regularization
4	Dropout	–	–	40% dropout
5	Dense	32	ReLU	L2 Regularization
6	Output (Dense)	1	Sigmoid	Binary classification
Training Parameters				
Optimizer		Adam		
Learning Rate		0.001		
Batch Size		128		
Epochs		10		

The autoencoder model consists of three dense layers in the encoder and three dense layers in the decoder. In the first stage, the encoder reduces the data dimensionality by successively passing through layers with 128, 64, and 32 neurons, all using the ReLU activation function. The last encoder layer defines the latent space and represents the compressed encoding of the input data. In the second stage, the decoder reconstructs the data from this latent representation, also using ReLU in the 64 and 128-neuron layers, and a sigmoid activation function in the final layer, which matches the original input dimension. The model is compiled using the Adam optimizer, and the Mean Squared Error (MSE) is adopted as the loss function, as it is well-suited for reconstruction tasks.

In the model configuration, the threshold was set at the 70th percentile of the MSE values calculated on the validation data. Training was performed with a batch size of 32 samples and 10 epochs. The same number of epochs was also used for the MLP model. Table 3 presents the Autoencoder model architecture, developed following [Ali et al. 2022].

Table 3. Autoencoder Model Architecture

Layer	Type	Neurons	Activation	Additional Features
1	Dense	128	ReLU	–
2	Dense	64	ReLU	–
3	Dense	32	ReLU	–
4	Dense	64	ReLU	–
5	Dense	128	ReLU	–
6	Output (Dense)	–	Sigmoid	Reconstruction
Training Parameters				
Optimizer		Adam		
Loss Function		MSE 70th percentile		
Batch Size		32		
Epochs		10		

The implementation of the proposed models and experiments is publicly available at our GitHub repository: https://github.com/gabrieladamasceno/MLP-and-Autoencoder-Models-for-Zero-Day-Attack-Detection.

4.5. Metrics for Attack Detection Analysis

In binary classification problems, such as attack detection, the terms True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) always appear. TP occurs when the model correctly classifies a positive instance, while TN (True Negative) occurs when it correctly classifies a negative instance. FP is an error where the model classifies a negative instance as positive, and finally, FN is when the model fails to identify a positive instance, classifying it as negative. In our research, a positive instance refers to any type of attack and a negative instance refers to benign network traffic. In this work, we consider the following metrics to evaluate the solutions regarding the attack detection.

a) Accuracy: it is the ratio between the number of correct predictions and the total number of predictions made [Sarhan et al. 2023]. It is computed via Eq. 3, which considers the TP, TN, FP, and FN values.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{3}$$

b) Precision: it is defined as the ratio between true positives (TP) and the sum of true positives and false positives (FP), as shown in Eq. 4. It refers to the percentage of correctly identified positives out of all the results predicted as positive [Ali et al. 2022].

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4}$$

c) Recall: it is defined as the ratio of true positives to the sum of true positives and false negatives (FN), as exhibited in Eq. 5. It represents the percentage of correctly identified positives relative to all actual positive samples [Ali et al. 2022].

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5}$$

d) F1-Score: this metric is defined by the harmonic mean of precision and recall, considering both false negatives and false positives in its computation (see Eq. 6). It is particularly useful when dealing with imbalanced data sets [Sarhan et al. 2023]:

$$\text{F1-Score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} \tag{6}$$

e) Zero-day attack detection ratio (ZD_{DR}): it is the percentage of correctly classified zero-Day attack samples in the test set [Sarhan et al. 2023], computed via Eq. 7.

$$ZD_{DR} = \frac{TP_{ZD}}{TP_{ZD} + FN_{ZD}} \tag{7}$$

4.6. Metric for Computational Resource Analysis

In addition to evaluating the detection performance, it is essential to consider the computational cost of the models. This paper considers the following main resource usage metrics.

a) CPU Usage: this metric refers to the average percentage of central processing unit (CPU) resources used during model execution, both in training and testing. Lower CPU usage indicates a more efficient model in terms of processing power. This metric is relevant for deploying models on devices that have limited resources, such as mobile devices or IoT ones [Agbedanu et al. 2025].
b) Memory Consumption: memory usage is measured in megabytes (MB) and represents the average amount of Random Access Memory (RAM) required during model operation. It is best suited for real-time applications or devices with limited memory to keep memory consumption as low as possible [Agbedanu et al. 2025].
c) Execution Time: it refers to the total time in seconds required to complete the training phase or the testing phase. Shorter execution times are desirable, especially in security scenarios that require fast or real-time processing [Agbedanu et al. 2025].

5. Results

This section discusses the results obtained by the MLP and Autoencoder models, covering both overall detection performance and the detection of zero-day attacks, as well as their memory usage, CPU consumption, and execution time.

5.1. Simulation Environment

The experiments were performed using Google Colaboratory (Colab) [Google 2025], a cloud computing platform provided by Google that allows the development and execution of Jupyter [Team 2025] notebooks. Additionally, the environment featured an Intel® Xeon® CPU at 2.20 GHz. Colab was chosen because the platform provides native support for several libraries in the Python [Foundation 2025] language that were employed in the analysis, such as TensorFlow [Abadi et al. 2015], Keras [Chollet 2015], PyTorch [Paszke et al. 2019], Scikit-learn [Pedregosa et al. 2011], Pandas [The pandas development team 2024], etc.

5.2. Zero-day attack and overall detection

Table 4 presents the results for precision, recall, and F1 score. Consistent with the previous findings, the MLP model achieved superior overall detection performance across various attack types. For most attacks, the MLP attained a precision above 0.99, with the exception of the Benign class in the UDPFlood and HTTPFlood scenarios. Similar results can be observed for recall and F1 score, although the MLP model shows degraded performance when facing UDPFlood and HTTPFlood attacks. In generall, MLP presents a robust and reliable performance, especially important for critical environments such as 6G.

In contrast, the Autoencoder demonstrates severely limited effectiveness, with recall and F1-score values below 0.1, indicating that it fails to detect actual attacks, compromising the fundamental purpose of a security system in 6G networks. Even the seemingly reasonable precision of 0.62 for HTTPFlood attacks is misleading, as its extremely low recall reveals that correct detections are rare. Overall, the MLP model consistently identifies a significantly greater proportion of malicious traffic and achieves higher F1-scores across both classes. The Autoencoder has demonstrated limited performance in several zero-day attack scenarios, especially for certain types of attacks such as UDPFlood, detecting only 10% of the threats, meaning that 90% of attacks would go unnoticed. While not ideal as a standalone solution, it can still be used in hybrid or clustered systems.

Table 4. Overall Attack Detection Results

Attack Type	Class	MLP			Autoencoder		
		Precision	Recall	F1-Score	Precision	Recall	F1-Score
ICMPFlood	Benign	0.9959	0.9989	0.9974	0.3867	0.9327	0.5468
	Malicious	0.9993	0.9973	0.9983	0.4697	0.0387	0.0716
UDPFlood	Benign	0.5107	0.9964	0.6753	0.3687	0.8892	0.5213
	Malicious	0.9938	0.3795	0.5493	0.1268	0.0105	0.0193
SYNFlood	Benign	0.9948	0.9980	0.9964	0.3868	0.9328	0.5468
	Malicious	0.9987	0.9966	0.9976	0.4706	0.0388	0.0717
SYNScan	Benign	0.9995	0.9982	0.9988	0.3869	0.9331	0.5470
	Malicious	0.9988	0.9997	0.9993	0.4726	0.0390	0.0720
HTTPFlood	Benign	0.8619	0.9982	0.9251	0.3949	0.9523	0.5583
	Malicious	0.9987	0.8961	0.9446	0.6241	0.0515	0.0951
UDPScan	Benign	0.9984	0.9996	0.9990	0.3869	0.9331	0.5470
	Malicious	0.9997	0.9990	0.9994	0.4732	0.0390	0.0721
TCPConnectScan	Benign	0.9971	0.9983	0.9977	0.3867	0.9326	0.5467
	Malicious	0.9989	0.9981	0.9985	0.4688	0.0387	0.0715
SlowrateDoS	Benign	0.9966	0.9996	0.9981	0.3865	0.9322	0.5465
	Malicious	0.9997	0.9978	0.9988	0.4655	0.0384	0.0710

Fig. 3 presents the overall accuracy of the models, considering both known and zero-day attacks. It can be observed that, for all attack types, the MLP approach outperforms the Autoencoder, achieving a gain of 0.6073 in detecting UDPScan attacks and maintaining high accuracy, above 0.99, for most attack types, except for UDPFlood. In contrast, the Autoencoder exhibited poor performance, with limited accuracy about 0.40, failing to effectively detect attacks in 6G network scenarios. The MLP achieved its highest accuracy with the UDPScan attack (0.9992), whereas the Autoencoder performed best on HTTPFlood attacks, reaching only 0.4063. These results reinforce the reliability of the MLP for critical applications such as threat detection in 6G networks, whereas the Autoencoder's limited performance highlights its unsuitability for such use cases.

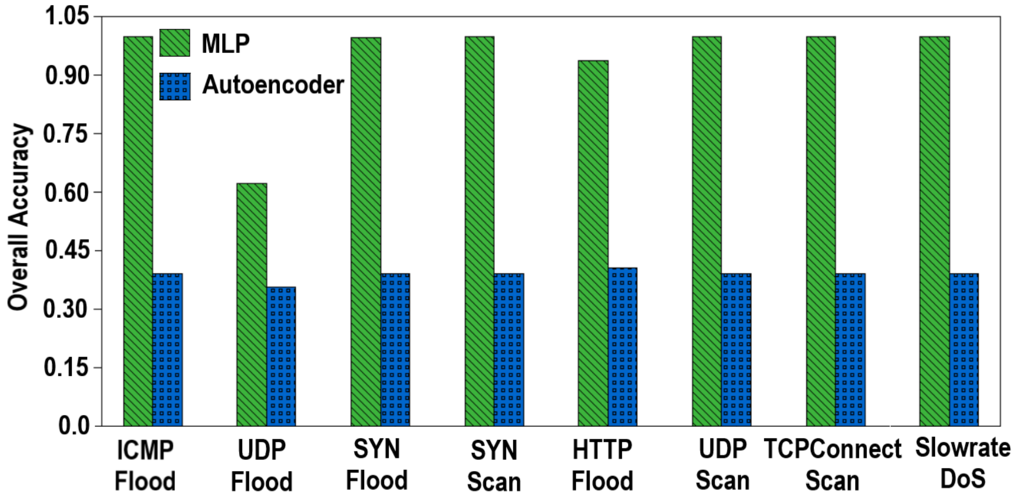

Figure 3. Overall Accuracy of the Models

Table 5 presents the zero-day attack detection ratios achieved by the models. The MLP achieved high detection ratios for attacks such as SYNFlood, SYNScan, HTTPFlood, and SlowrateDoS. However, the model struggled to detect UDPFlood and UDPScan as zero-day attacks. The Autoencoder model, in contrast, exhibited a less consistent performance overall compared to the MLP. Nevertheless, it achieved noteworthy results in certain cases, such as a detection ratio of 0.9900 for UDPFlood and 0.6300 for ICMPFlood attacks.This suggests that the Autoencoder is more sensitive to specific types of attacks, but it fails to maintain a consistently high detection ratio across all attack types.

Table 5. Zero-Day Detection Ratio Obtained by MLP and Autoencoder

Zero-Day Attack	Samples	MLP ZD_{DR}	Autoencoder ZD_{DR}
ICMPFlood	238	0.9370	0.6300
UDPFlood	91421	0.0000	0.9900
SYNFlood	1919	1.0000	0.0000
SYNScan	3894	1.0000	0.0010
HTTPFlood	28163	1.0000	0.0000
UDPScan	3193	0.7269	0.4300
TCPConnectScan	4053	0.9280	0.1000
SlowrateDoS	14503	1.0000	0.0037

5.3. Resource Consumption

The analysis of computational resource consumption during the training and testing phases of the MLP and Autoencoder models revealed notable differences. As shown in

Table 6, the MLP model generally exhibited lower CPU usage during training compared to the Autoencoder (Table 7), maintaining CPU utilization below 0.40% in scenarios such as ICMPFlood, SYNFlood, and SYNScan. In contrast, the Autoencoder exceeded 0.50% CPU usage in these same attack scenarios. However, for attacks like UDPScan and HTTPFlood, the MLP's CPU usage increased to approximately 2.50%, whereas the Autoencoder maintained lower usage, ranging between 0.10% and 0.20%. In the testing phase, both models had similar CPU consumption considering each zero-Day attack.

Regarding memory usage, a notable contrast was observed between the two models. The Autoencoder (Table 7) exhibited a more consistent memory consumption pattern across all attack types, with values ranging from 3126.04 MB to 4407.25 MB during training and testing. In contrast, the MLP (Table 6) showed significantly higher memory peaks, particularly in scenarios involving HTTPFlood and UDPScan attacks—reaching 14,800.30 MB in training and 45,985.87 MB in testing for HTTPFlood, and 26,482.63 MB and 90,370.23 MB in training and testing, respectively, for UDPScan. These values indicate potential model overload under these conditions, which also corresponded with elevated CPU usage.

In terms of training time, the MLP model (Table 6) was more efficient, exhibiting shorter execution times in most of the evaluated scenarios. In contrast, the Autoencoder model (Table 7) required significantly longer training times, exceeding 660 seconds for the SYNScan and UDPScan attacks. Testing times, however, remained relatively similar between the two models, averaging around 10 to 11 seconds in most cases—except for the MLP during HTTPFlood and UDPScan scenarios, where elevated CPU and memory usage contributed to increased test times of 16.50 and 16.57 seconds, respectively. These results indicate that CPU consumption remained relatively similar between the two models; however, the MLP outperformed the Autoencoder in terms of training execution time, while the Autoencoder exhibited more stable memory usage across both training and testing phases.

Table 6. MLP Resources Usage During Training and Testing

Attack Type	Train CPU	Train Memory (MB)	Train Time (s)	Test CPU	Test Memory (MB)	Test Time (s)
ICMPFlood	0.36%	4165.18	269.12	0.53%	4116.70	10.79
UDPFlood	0.13%	3009.26	110.37	0.43%	2997.40	10.22
SYNFlood	0.30%	4079.59	186.56	0.33%	4021.74	10.63
SYNScan	0.37%	4008.03	237.94	0.49%	3952.58	10.72
HTTPFlood	2.59%	14800.30	206.81	1.40%	45985.87	16.50
UDPScan	2.56%	26482.63	408.90	1.40%	90370.23	16.57
TCPConnectScan	0.55%	3270.74	257.10	0.47%	3226.94	10.52
SlowrateDoS	0.37%	3232.84	213.55	0.32%	3232.50	9.68

Table 7. Autoencoder Resources Usage During Training and Testing

Attack Type	Train CPU	Train Memory (MB)	Train Time (s)	Test CPU	Test Memory (MB)	Test Time (s)
ICMPFlood	0.60%	4341.87	592.72	0.37%	4407.25	10.32
UDPFlood	0.25%	3126.04	381.10	0.53%	3169.65	10.30
SYNFlood	0.70%	3356.22	658.94	0.45%	3413.56	10.66
SYNScan	0.50%	3334.38	669.76	0.48%	3411.24	10.73
HTTPFlood	0.10%	3242.04	470.96	0.38%	3305.47	10.10
UDPScan	0.20%	3309.71	669.98	0.35%	3353.18	11.09
TCPConnectScan	0.05%	3271.10	643.06	0.54%	3310.90	10.76
SlowrateDoS	0.15%	3297.73	606.62	0.51%	3362.32	10.69

6. Conclusion and Future Works

This study evaluated the performance of MLP and Autoencoder models for Zero-Day attack detection in 6G networks. A dataset comprising various types of attacks collected from a real 5G network was used, with data preprocessing performed to define Zero-Day attack samples. The results indicate that the MLP model outperformed the Autoencoder in detecting Zero-Day attacks, achieving high precision, recall, and F1-score, even without prior exposure to the attack samples, demonstrating its capacity to generalize and identify anomalous patterns.

In contrast, although the Autoencoder exhibited more consistent computational resource usage, particularly in terms of memory and CPU consumption, its effectiveness in detecting zero-day attacks was significantly lower. Additionally, its longer training time can be a disadvantage in mobile network security scenarios, as it depends on the solution being employed. Although training is typically performed offline, in some solutions for dynamic environments that adopt online or continuous training, training time may be considered relevant as it is a real-time task, where timely detection is crucial due to the real-time nature of potential threats.

In general, the results denoted that MLP fulfills its objective of detecting unknown threats, thus contributing to strengthening the security posture of emerging 6G networks. As future work, we recommend exploring other machine learning techniques and evaluating the model's performance across different datasets and attack scenarios. In addition, we also plan to extend the evaluation to environments with limited resources, such as IoT and Edge Computing scenarios, since, being the solution employed in this type of context, models with lower attack detection performance, but which require fewer computational resources, may be the most suitable.

Acknowledgment

This study was funded by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Code 001 and by the UFPE/Propesqi via Edital Nº 06/2024.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Abri, F., Siami-Namini, S., Khanghah, M. A., Soltani, F. M., and Namin, A. S. (2019). Can machine/deep learning classifiers detect zero-day malware with high accuracy? In *2019 IEEE international conference on big data (Big Data)*, pages 3252–3259. IEEE.

Agbedanu, P. R., Yang, S. J., Musabe, R., Gatare, I., and Rwigema, J. (2025). A scalable approach to internet of things and industrial internet of things security: Evaluating adaptive self-adjusting memory k-nearest neighbor for zero-day attack detection. *Sensors*, 25(1):216.

Ahmad, R., Alsmadi, I., Alhamdani, W., and Tawalbeh, L. (2023). Zero-day attack detection: a systematic literature review. *Artificial Intelligence Review*, 56(10):10733–10811.

Al-Rushdan, H., Shurman, M., Alnabelsi, S. H., and Althebyan, Q. (2019). Zero-day attack detection and prevention in software-defined networks. In *2019 International Arab Conference on Information Technology (ACIT)*, pages 278–282.

Ali, S., Rehman, S. U., Imran, A., Adeem, G., Iqbal, Z., and Kim, K.-I. (2022). Comparative evaluation of ai-based techniques for zero-day attacks detection. *Electronics*, 11(23):3934.

Benzaïd, C., Hossain, F. M., Taleb, T., Gómez, P. M., and Dieudonne, M. (2024). A federated continual learning framework for sustainable network anomaly detection in o-ran. In *2024 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE.

Chollet, F. (2015). Keras. GitHub repository: https://github.com/keras-team/keras.

De Azambuja, A. J. G., Plesker, C., Schützer, K., Anderl, R., Schleich, B., and Almeida, V. R. (2023). Artificial intelligence-based cyber security in the context of industry 4.0—a survey. *Electronics*, 12(8):1920.

Foundation, P. S. (2025). Python documentation. https://docs.python.org/3/.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Google (2025). Google colaboratory. https://colab.research.google.com/.

Guo, Y. (2023). A review of machine learning-based zero-day attack detection: Challenges and future directions. *Computer communications*, 198:175–185.

Hamid, K., Iqbal, M. W., Aqeel, M., Liu, X., and Arif, M. (2022). Analysis of techniques for detection and removal of zero-day attacks (zda). In *International Conference on Ubiquitous Security*, pages 248–262. Springer.

Hindy, H., Atkinson, R., Tachtatzis, C., Colin, J.-N., Bayne, E., and Bellekens, X. (2020). Utilising deep learning techniques for effective zero-day attack detection. *Electronics*, 9(10):1684.

Imoize, A. L., Adedeji, O., Tandiya, N., and Shetty, S. (2021). 6G enabled smart infrastructure for sustainable society: Opportunities, challenges, and research roadmap. *Sensors*, 21(5).

Jain, P., Gupta, A., and Kumar, N. (2022). A vision towards integrated 6G communication networks: Promising technologies, architecture, and use-cases. *Physical Communication*, 55:101917.

Kianpisheh, S. and Taleb, T. (2024). Collaborative federated learning for 6G with a deep reinforcement learning based controlling mechanism: A ddos attack detection scenario. *IEEE Transactions on Network and Service Management*.

Lawrence, S. and Giles, C. L. (2000). Overfitting and neural networks: conjugate gradient and backpropagation. In *Proceedings of the IEEE-INNS-ENNS International Joint*

Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, volume 1, pages 114–119. IEEE.

Lu, H., Zhao, Y., Song, Y., Yang, Y., He, G., Yu, H., and Ren, Y. (2024). A transfer learning-based intrusion detection system for zero-day attack in communication-based train control system. *Cluster Computing*, 27(6):8477–8492.

Mohamed, A. A., Al-Saleh, A., Sharma, S. K., and Tejani, G. G. (2025). Zero-day exploits detection with adaptive wavepca-autoencoder (awpa) adaptive hybrid exploit detection network (ahednet). *Scientific Reports*, 15(1):4036.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.

Redino, C., Nandakumar, D., Schiller, R., Choi, K., Rahman, A., Bowen, E., Shaha, A., Nehila, J., and Weeks, M. (2022). Zero day threat detection using graph and flow based security telemetry. In *2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pages 655–662. IEEE.

Samarakoon, S., Siriwardhana, Y., Porambage, P., Liyanage, M., Chang, S.-Y., Kim, J., Kim, J., and Ylianttila, M. (2022). 5G-NIDD: A comprehensive network intrusion detection dataset generated over 5G wireless network. *arXiv preprint arXiv:2212.01298*.

Sameera, N. and Shashi, M. (2020). Deep transductive transfer learning framework for zero-day attack detection. *ICT Express*, 6(4):361–367.

Sarhan, M., Layeghy, S., Gallagher, M., and Portmann, M. (2023). From zero-shot machine learning to zero-day attack detection. *International Journal of Information Security*, 22(4):947–959.

Taud, H. and Mas, J.-F. (2018). Multilayer perceptron (MLP). *Geomatic approaches for modeling land change scenarios*, pages 451–455.

Team, J. D. (2025). Jupyter documentation. https://jupyter.org/documentation.

Teoh, T., Chiew, G., Franco, E. J., Ng, P., Benjamin, M., and Goh, Y. (2018). Anomaly detection in cyber security attacks on networks using MLP deep learning. In *2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*, pages 1–5. IEEE.

The pandas development team (2024). pandas-dev/pandas: Pandas.

Wu, J., Li, R., An, X., Peng, C., Liu, Z., Crowcroft, J., and Zhang, H. (2021). Toward native artificial intelligence in 6G networks: System design, architectures, and paradigms. *arXiv preprint arXiv:2103.02823*.

Yoshioka, P., Damasceno, M., Balieiro, A., Swain, S. N., and Alves, E. (2024). A decision-tree solution for the outdated CQI feedback problem in 5G networks. In

2024 XIV Brazilian Symposium on Computing Systems Engineering (SBESC), pages 1–6. IEEE.

Zhang, Z., Cao, Y., Cui, Z., Zhang, W., and Chen, J. (2021). A many-objective optimization based intelligent intrusion detection algorithm for enhancing security of vehicular networks in 6G. *IEEE Transactions on Vehicular Technology*, 70(6):5234–5243.

Ziegler, V., Viswanathan, H., Flinck, H., Hoffmann, M., Räisänen, V., and Hätönen, K. (2020). 6G architecture to connect the worlds. *IEEE Access*, 8:173508–173520.