From Vulnerability to Resilience: Adversarial Training and Real-Time Detection for AI Security

Georgios Ziras[a], Aristeidis Farao[a,b], Apostolis Zarras[a,c] and Christos Xenakis[a]

[a]Department of Digital Systems, University of Piraeus, Piraeus, Greece
[b]InQbit Innovations SRL., Bucharest, Romania
[c]Foundation for Research and Technology - Hellas, Heraklion, Greece

ARTICLE INFO

Keywords:
Adversarial Attacks
Machine Learning Models
Adversarial Training

ABSTRACT

The growing integration of Artificial Intelligence systems into critical infrastructure, such as cybersecurity, healthcare, and finance, has raised significant concerns regarding model robustness in the presence of adversarial attacks. This study examines the vulnerability of various machine learning models to adversarial manipulations and evaluates effective detection and mitigation strategies to improve model resilience. Leveraging the CIC-IDS2017 and CICIoT2023 datasets, we train and evaluate a suite of ML classifiers, including Decision Tree, Random Forest, Logistic Regression, XGBoost, Recurrent Neural Networks, Convolutional Neural Networks, and a custom PyTorch-based Neural Network, under a spectrum of adversarial evasion attacks. These include the Fast Gradient Sign Method, Projected Gradient Descent, DeepFool, Carlini–Wagner, and transfer attacks. We assess classifier robustness against those attacks and examine their defensive behavior through adversarial training, as well as binary input and activation-based detection mechanisms. Our findings indicate that adversarial training provides a more effective and consistent defense compared to detection-based methods.

1. Introduction

Cybersecurity attacks fundamentally aim to compromise a system's confidentiality, integrity, or availability by causing it to behave in unintended ways. Similarly, adversarial Machine Learning (ML) attacks target Artificial Intelligence (AI) systems by manipulating them to produce incorrect or misleading outputs. These attacks typically involve the creation of carefully crafted malicious inputs (a.k.a. adversarial examples) that exploit model vulnerabilities, leading to misclassification and undermining the model's reliability and accuracy.

Adversarial ML attacks can occur at any stage of an AI model's lifecycle, including the training, testing, and deployment phases. They can be broadly categorized into evasion, poisoning, and privacy-based attacks. Moreover, adversarial attacks may be either *targeted*, wherein the attacker aims to induce a specific erroneous behavior or output, or *untargeted*, where the objective is to cause general misclassification without a predefined outcome.

As AI systems become increasingly integrated into mission-critical domains such as healthcare, finance, defense, and autonomous systems—where security and reliability are paramount—their susceptibility to adversarial attacks poses a significant threat. These subtle manipulations can compromise system functionality and lead to severe real-world consequences. Therefore, understanding, detecting, and mitigating such threats has become an essential area of research.

This work investigates the susceptibility of AI models to adversarial perturbations while measuring their baseline accuracy and robustness. The CIC-IDS2017 [33] and CICIoT2023 [22] datasets are employed to train a variety of classification algorithms, including Decision Tree (DT), Random Forest (RF), Logistic Regression (LR), eXtreme Gradient Boosting (XGBoost), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and PyTorch MLP model. This dataset was selected for its realistic representation of regular network traffic and contemporary cyber threats. It consists of labeled network flow features extracted from packet captures (PCAPs) using CICFlowMeter.

*Corresponding author
**Principal corresponding author
georgios_ziras@ssl-unipi.gr (G. Ziras)
ORCID(s): 0000-0001-0000-0000 (G. Ziras); 0000-0001-0000-0000 (A. Farao); 0000-0001-0000-0000 (A. Zarras); 0000-0001-0000-0000 (C. Xenakis)
[1]Ziras et.al

Following the initial training and evaluation of these models, a suite of adversarial attacks, including Fast Gradient Sign Method (FGSM), Projected Gradient Descent (PGD), DeepFool, and Carlini & Wagner (C&W), is applied to assess the models' resilience against adversarial perturbations. Additionally, this study explores adversarial transferability, wherein adversarial examples generated for one model are tested against others with different architectures. This aspect provides insights into shared vulnerabilities and model generalizability. To enhance robustness, adversarial training is employed by augmenting the training dataset with examples specifically crafted to exploit model weaknesses. This iterative process aims to improve the model's ability to detect and resist adversarial inputs. The retrained models are then re-evaluated to assess improvements in accuracy and resilience. Furthermore, this research investigates real-time adversarial detection methods. One such approach involves feature squeezing, which reduces input complexity to highlight discrepancies between original and perturbed inputs. Another technique, Gradient-Based Anomaly Detection, analyzes the distribution of gradients, with adversarial examples typically exhibiting less peaked gradients than benign samples.

In summary, we make the following main contributions:

- We train various AI models and comprehensively assess their vulnerabilities through carefully designed adversarial attacks.
- We develop and implement robust AI models by retraining them using adversarial training with clean and adversarially perturbed datasets.
- We present, to the best of our knowledge, the first practical implementation and evaluation of real-time adversarial attack detection methods.
- We analyze and quantify improvements in robustness and accuracy of AI models resulting from adversarial training compared to traditional models.

The remainder of this article is organized as follows. Section 2 presents background knowledge, including vulnerabilities in AI systems, an analysis of adversarial ML attacks, and an overview of detection and mitigation techniques. Section 3 describes the methodology for conducting adversarial attacks on ML models, as well as the procedures for adversarial training, integration of defenses, and the performance evaluation metrics used. Section 4 discusses the study's limitations and outlines future research directions, while Section 5 reviews related work. Finally, Section 6 concludes the article.

2. Background

This section outlines the key vulnerabilities of AI-based systems, the main types of adversarial AI attacks, and current mitigation and detection strategies designed to counter such threats.

2.1. Vulnerabilities in AI-based Systems

AI-based systems are increasingly deployed in critical domains such as healthcare, finance, autonomous vehicles, defense, and cybersecurity. These systems leverage complex ML models to analyze data, make decisions, and adapt to new inputs. Despite their capabilities, AI systems remain inherently vulnerable due to their dependence on data-driven learning and generalization processes [2, 6, 13]. The key challenges are described below.

A primary vulnerability stems from the *dependence of AI models on the quality and representativeness of their training data*. Biases, imbalances, or incompleteness in the training dataset can severely hinder the model's generalization ability, making it susceptible to adversarial manipulation. Moreover, the non-linearity and high dimensionality of modern Deep Learning (DL) models make them particularly sensitive to adversarial perturbations, where minimal input alterations can lead to significant changes in output.

Another key vulnerability is the *lack of robust defense mechanisms* in many existing AI systems. Most ML models are optimized for accuracy on clean datasets and are not inherently resilient to adversarial inputs, creating opportunities for attackers to exploit these weaknesses.

Furthermore, AI systems operating in real-world settings often *function in dynamic and uncertain environments*. This variability enables adversaries to craft context-aware adversarial examples that evade detection and disrupt operations. In complex tasks such as multiclass classification [23, 35], these manipulations can lead to misclassification into plausible but incorrect classes, making detection even more challenging.

2.2. Adversarial AI Attacks

Adversarial attacks involve the deliberate design of inputs intended to deceive ML and AI models by exploiting their vulnerabilities. These inputs, known as adversarial examples, are crafted to induce incorrect predictions, thus compromising the model's integrity, reliability, and performance [23]. A defining characteristic of adversarial attacks is their ability to remain nearly imperceptible to human observers while being highly effective in deceiving AI systems.

The motivations behind adversarial attacks range from malicious intent, such as disrupting critical systems, to ethical hacking aimed at uncovering and fixing vulnerabilities. The consequences of such attacks can be severe, particularly in high-stakes environments like healthcare, finance, or transportation.

Adversarial attacks are typically classified based on the adversary's knowledge and access level:

- ***White-Box Attacks:*** The attacker has complete knowledge of the model, including its architecture, parameters, and training data. This allows for highly effective input manipulations based on the model's internal structure [6].
- ***Gray-Box Attacks:*** The adversary has partial information, such as knowledge of the model architecture, but not its parameters or training data. Hybrid strategies combine limited internal knowledge with external probing [6].
- ***Black-Box Attacks:*** The attacker has no internal access to the model and relies solely on querying it to infer behavior. These attacks often use surrogate models or exploit the transferability property of adversarial examples [6].

Adversaries employ various methods to craft adversarial inputs tailored to the level of access and knowledge available. Gradient-based techniques are among the most commonly used, leveraging the model's gradients to identify directions that maximize the adversarial impact. Examples include the FGSM, which adds perturbations in the gradient's direction, and the PGD, which iteratively refines adversarial examples for stronger attacks.

Beyond gradient-based approaches, other methods include: (*i*) *C&W:* an optimization-based method that minimizes perturbations while ensuring misclassification [5]; (*ii*) *DeepFool:* an iterative algorithm that computes the minimum perturbation required to cross the decision boundary [21]; (*iii*) *Zeroth-Order Optimization (ZOO):* a black-box attack that estimates gradients using model queries [7]; (*iv*) *Boundary Attack:* a decision-based method that begins with a large perturbation and gradually reduces it [3]; (*v*) *Transfer-based Attack:* exploits the ability of adversarial examples to generalize across different models [32]. These methods underscore the versatility of adversarial attacks, as they can adapt to different levels of system access and defenses. Understanding these techniques is essential for designing robust defenses against adversarial threats.

Adversarial attacks may also be classified based on their methodology and the stage of the ML pipeline they target [6]:

- ***Evasion Attacks:*** Executed during the inference phase, these attacks bypass the model's defenses by modifying test inputs without altering the training data.
- ***Poisoning Attacks:*** Conducted during training, these attacks inject malicious data to compromise the learning process and embed vulnerabilities.
- ***Exploratory Attacks:*** These attacks probe the system to gather insights without modifying training data, often serving as a precursor to evasion or poisoning.

2.3. Mitigation Techniques

Mitigation strategies aim to strengthen ML models against adversarial threats by improving robustness. These techniques include modifying training procedures, augmenting input data, or altering model architectures [2].

Adversarial Training [16, 29, 34] is one of the most widely studied and effective mitigation strategies. In this approach, the model is trained on a mixture of clean and adversarial perturbed examples. The inclusion of adversarial examples in the training process forces the model to learn how to classify both clean and perturbed inputs correctly, thus improving its robustness to future attacks. For example, adversarial training typically involves generating adversarial examples using methods like FGSM or PGD and adding these examples to the training data. By repeatedly exposing the model to adversarial perturbations, the model learns to resist such attacks by recognizing patterns indicative of adversarial perturbations.

Input Transformation and Data Augmentation techniques modify the input data in ways that reduce the impact of adversarial perturbations, often by removing the noise introduced by adversarial attacks. These transformations can include techniques such as feature squeezing, image cropping, or random transformations [18].

Certified Defenses [25, 30] focus on providing formal guarantees that a model will be robust to adversarial attacks within specific bounds. These methods use mathematical reasoning and optimization techniques to compute provable robustness guarantees. However, certified defenses are often computationally expensive and challenging to implement for complex models.

Ensemble Learning [9, 10, 27] is a powerful strategy that combines the predictions of multiple models to make more robust decisions. The underlying idea is that while individual models may be vulnerable to specific adversarial attacks, an ensemble of models, each trained on slightly different data or with varying architectures, can reduce the overall vulnerability of the system. By aggregating the predictions of multiple models, ensemble methods help mitigate the impact of adversarial perturbations, making it more difficult for an attacker to deceive all models in the ensemble simultaneously.

Regularization Methods [28], such as weight decay and dropout, are designed to improve the generalization capabilities of a model. Regularization prevents the model from overfitting to adversarial examples by penalizing overly complex decision boundaries, making it more difficult for small, imperceptible perturbations to drastically alter model predictions.

2.4. Detection Techniques

Detection techniques are critical for identifying adversarial inputs before they can compromise model performance. The goal is to detect adversarial perturbations at the earliest possible stage, thereby enabling corrective measures to be applied in real time.

Gradient-based Detection involves analyzing the gradients of the model's loss function with respect to its inputs. Adversarial examples tend to exhibit large or unusual gradients compared to clean inputs due to their perturbations. By examining the gradient behavior, one can identify anomalies indicative of adversarial manipulation.

Feature-based Detection focuses on detecting adversarial perturbations in the learned features of a model. Since adversarial attacks are designed to exploit learned representations, these attacks often cause discrepancies in the feature space. Detection can thus be achieved by analyzing the activation values of intermediate layers in the neural network.

Input Preprocessing and Transformation approach involves preprocessing the input data to remove or mitigate the effects of adversarial perturbations. Techniques such as image denoising or data transformation are used to filter out noise and reduce the impact of adversarial manipulation.

Statistical Methods for adversarial detection focus on analyzing the statistical properties of the input data. These methods examine distributions of features such as mean, variance, skewness, and kurtosis, which may shift due to adversarial perturbations. Adversarial inputs typically lead to anomalies that can be detected through statistical outlier detection techniques. Statistical Outlier Detection involves the analysis of input features to detect statistical anomalies. By evaluating the statistical properties of the data, this method can detect adversarial examples that deviate from the expected data distribution.

On the detection side, statistical feature analysis offers a promising method for identifying adversarial examples. This approach leverages the statistical properties of the input data, such as its mean, variance, and higher-order moments, to detect anomalies indicative of adversarial manipulation. Adversarial perturbations often cause noticeable shifts in the statistical distribution of the data, and by examining these shifts, statistical methods can efficiently identify adversarial inputs. Moreover, statistical feature analysis is computationally efficient and can be implemented in real-time detection systems [2].

In parallel, statistical analysis offers an efficient and scalable method for adversarial detection; analyzing shifts in the statistical distribution of features enables the real-time identification of manipulated inputs. Combined, these techniques offer a robust and complementary defense mechanism: adversarial training reinforces model resilience, while statistical analysis ensures rapid detection of adversarial threats, enhancing the security and reliability of AI systems.

3. Methodology

This section outlines the methodology employed to evaluate the robustness of ML models against adversarial AI attacks.

Table 1
Number of samples per category found in used datasets

Category	Samples	Category	Samples	Category	Samples
CIC-IDS2017					
Benign	872,105	Port Scanning	12,843	SSH-Patator	5,897
DoS Hulk	231,073	DoS GoldenEye	10,293	DoS Slowloris	5,796
Normal Network Traffic	84,980	FTP-Patator	7,938	DoS Slowhttptest	5,499
CICIoT2023					
Benign	16,577	DDOS-ICMP_FLOOD	5,227	**DDOS-UDP_FLOOD**	3,962
DDOS-TCP_FLOOD	3,227	DDOS-RSTFINFLOOD	3,019	DDOS-PSHACK_FLOOD	3,003
DDOS-SYN_FLOOD	2,954	DDOS-SYNONYMOUSIP_FLOOD	2,620	DOS-UDP_FLOOD	2,490
DOS-TCP_FLOOD	1,907				

3.1. System Overview

This study adopts a comprehensive approach to evaluate and enhance the robustness of ML models in the context of network intrusion detection. All experiments were conducted on a system equipped with an AMD Ryzen 5 2600 Six-Core processor (12 threads), 32GB of RAM, and AMD-V virtualization support. The environment runs on Ubuntu 24.04.2 LTS, offering a stable, high-performance platform suitable for intensive model training and adversarial evaluation tasks. At this point we have to mention that all experiments performed 10 times.

The CIC-IDS2017 dataset [33] and CICIoT2023 [22] were selected for this research, as being widely recognized. CIC-IDS2017 addresses several limitations found in earlier datasets by offering comprehensive feature sets, realistic traffic patterns, and detailed labeling. The dataset includes both benign traffic and diverse cyberattack categories, such as denial-of-service (DoS), distributed DoS (DDoS), brute-force login attempts, infiltration, web-based attacks, port scans, and botnet activities. While, CICIoT2023 is a real-time dataset and benchmark for large-scale attacks in IoT environment.It includes the execution of 33 attacks in an IoT topology composed of 105 devices. These attacks are classified into seven categories, namely DDoS, DoS, Recon, Web-based, Brute Force, Spoofing, and Mirai. Finally, all attacks are executed by malicious IoT devices targeting other IoT devices. Initial analysis revealed the following distribution of network traffic categories as depicted in Table 1

Data preprocessing included handling values by replacing them with *NaN*, addressing missing values, and removing duplicate entries. After duplicate removal, the dataset was reduced from an initial 1,236,424 rows to 1,122,397. Further preprocessing involved encoding categorical variables and scaling numerical data. Later on, the dataset split into a training dataset that includes 785,677 samples with 78 features and a testing dataset that includes 336,720 samples with 78 features, for ensuring the same time reliability and accuracy of subsequent intrusion detection model evaluations.

The preprocessing phase involved the following actions to ensure data quality and readiness for modeling: (*i*) *Data Cleaning* where the dataset was initially inspected for missing values and infinite values—missing or corrupted data were identified and replaced accordingly to ensure dataset integrity; (*ii*) *Encoding* where all categorical features in the dataset were encoded numerically using label encoding, converting textual information into numerical representations suitable for machine learning algorithms; (*iii*) *Data Scaling* where numerical features were scaled using the StandardScaler method to standardize feature ranges, improving model training efficiency and performance consistency; (*iv*) *Dataset Splitting* where the final preprocessed dataset was split into training and testing subsets (commonly 70–30) to facilitate unbiased evaluation of the model's performance—this splitting ensured reproducibility and robustness in performance assessment. Overall, through this preprocessing, the resulting dataset provided a reliable foundation for training ML models capable of accurately detecting and responding to adversarial attacks, thereby ensuring the relevance and applicability of the experimental results [12].

In this study, we trained and evaluated multiple ML models to detect adversarial attacks using the CIC-IDS2017 and CICIoT2023 datasets. The models selected include DT, RF, LR, XGBoost, and Deep Neural Network (DNN), CNN and RNN and PyTorch MLP model. DT was chosen for its simplicity, interpretability, and computational efficiency, making it suitable for baseline evaluation. It effectively identifies clear decision rules and highlights the most significant features contributing to adversarial detection. RF was selected due to its robustness against overfitting, high accuracy across diverse datasets, and its ability to handle noisy or complex data by aggregating predictions from multiple trees to capture more intricate patterns. LR offered a straightforward probabilistic framework appropriate for binary classification tasks, providing a transparent and interpretable baseline for comparison. XGBoost was employed for its superior predictive performance, fast execution, and capability to address class imbalance (i.e., an issue commonly

Table 2
Baseline performance metrics per model

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
CIC-IDS2017				
RF	99.94	99.95	99.96	99.95
LR	97.56	96.58	97.01	97.55
XGBoost	99.97	99.81	99.77	99.79
DT	99.99	99.95	96.80	99.99
PyTorch MLP model	99.44	99.89	98.52	99.69
RNN	99.71	99.71	99.71	99.71
CNN	99.81	99.81	99.81	99.81
CICIoT2023				
RF	84.51	84.47	84.51	84.48
LR	87.06	87.96	87.06	85.65
XGBoost	86.18	85.96	86.18	85.82
DT	86.74	86.77	86.74	85.72
PyTorch MLP model	86.57	88.87	86.57	84.21
RNN	86.08	88.14	86.08	84.25
CNN	86.92	88.19	86.92	85.21

encountered in cybersecurity datasets) through its gradient-boosting approach, which enables iterative optimization and model refinement. Finally, a DNN was implemented using PyTorch MLP model to leverage its capacity for learning complex, nonlinear relationships in high-dimensional network traffic data. Neural networks are particularly advantageous in extracting hierarchical features and detecting subtle adversarial patterns that traditional models may not capture.

For performance evaluation, we employed several standard classification metrics to assess the effectiveness of the proposed models. Specifically, we used *accuracy* to measure the overall proportion of correctly classified instances, including benign and adversarial samples. *Precision* was utilized to evaluate the model's ability to correctly identify adversarial examples among all instances predicted as adversarial, a critical factor in reducing false positives. *Recall* (or *sensitivity*) was used to assess the model's capacity to detect actual adversarial inputs, essential for minimizing the likelihood of undetected attacks. Finally, the *F1-Score* was calculated as a harmonic mean of precision and recall, offering a balanced assessment of the model's performance, particularly in class imbalance scenarios.

Table 2 presents the baseline performance metrics for each assessed model. Regarding, the metrics occurred in CIC-IDS2017 dataset it is observed, DT achieved the highest accuracy, reaching 99.99%, whereas LR recorded the lowest accuracy at 97.56%. The remaining models all attained accuracy scores exceeding 99%. Regarding, the metrics occured in CICIoT2023 dataset it is observed (see Table 2), LR achieved the highest accuracy, reaching 87.06%, whereas RF recorded the lowest accuracy at 84.51%. The remaining models all attained accuracy scores exceeding 86%. These baseline metrics are reference points against which the performance improvements from adversarial training will be compared.

3.2. Robustness Evaluation Against Adversarial AI

Several adversarial attacks were performed to thoroughly evaluate the robustness of the aforementioned ML models. The *DT Attack*, a specialized adversarial technique tailored for DT-based models, was applied to the DT model. This attack exploits the hierarchical structure of decision trees by identifying and modifying critical feature values that influence the classification outcome. The *FGSM*, a fast, single-step attack, was executed against both the LR model and the PyTorch MLP model. This method perturbs input data by adjusting feature values in the direction of the gradient sign, generating minimal yet effective perturbations that mislead the model. The *PGD* attack was also applied to the LR and PyTorch MLP model. Unlike FGSM, PGD operates iteratively, applying small perturbations over multiple steps to refine adversarial modifications, thereby increasing the attack's effectiveness and making it more difficult to defend against. *DeepFool*, another iterative attack, was used on the same models. It computes the minimal perturbation required to shift an input sample across the decision boundary, leading to misclassification with minimal changes to the input. Finally, the *C&W* attack was conducted on the LR and PyTorch MLP model. This optimization-based technique aims to produce highly effective adversarial examples by minimizing the perturbation's perceptibility while maximizing its adversarial impact through a refined optimization process. The following setup is used for the aforementioned attacks. FGSM is applied with a perturbation magnitude ϵ ranging from 0.4 to 0.7,

Table 3
Model evaluation metrics after adversarial attacks

Performed Attack	Target Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
		CIC-IDS2017			
DT Attack	DT	0.27	22.29	0.18	0.07
FGSM	LR	1.99	2.85	2.29	2.65
PGD	LR	0.65	26.81	0.65	0.50
DeepFool	LR	1.62	3.24	1.62	1.61
C&W	LR	2.09	4.82	2.09	2.09
FGSM	PyTorch MLP model	78.54	31.69	22.64	23.98
PGD	PyTorch MLP model	72.00	18.76	11.04	22.55
DeepFool	PyTorch MLP model	65.92	18.76	11.04	10.55
C&W	PyTorch MLP model	55.94	33.18	17.83	18.76
FGSM	CNN	10.43	33.98	10.43	10.11
FGSM	RNN	18.94	33.18	18.94	23.87
PGD	CNN	0.85	2.55	0.85	1.21
PGD	RNN	0.24	0.67	0.24	0.34
DeepFool	CNN	0.15	0.43	0.15	0.18
DeepFool	RNN	0.38	1.29	0.38	0.58
		CICIoT2023			
DT Attack	DT	0.02	78.59	0.02	0.01
FGSM	LR	17.85	43.81	17.85	11.63
PGD	LR	11.24	88.96	11.24	2.63
DeepFool	LR	13.78	21.11	13.78	9.90
C&W	LR	12.90	24.07	12.90	6.94
FGSM	PyTorch MLP model	27.13	24.41	27.13	21.49
PGD	PyTorch MLP model	84.43	84.04	84.43	82.14
DeepFool	PyTorch MLP model	41.53	36.03	41.53	35.30
C&W	PyTorch MLP model	86.57	88.87	86.57	84.21
FGSM	CNN	17.12	19.69	17.12	17.07
PGD	CNN	20.76	20.16	20.76	18.02
DeepFool	CNN	2.09	48.23	2.09	3.29
FGSM	RNN	13.87	29.93	13.87	6.94
PGD	RNN	11.53	5.26	11.53	6.75
DeepFool	RNN	13.42	6.28	13.42	7.29

suitable for one-step attacks on structured data. PGD, a stronger iterative variant, uses ϵ values between 0.6 and 0.8, with a step size of 0.01 over 50 to 100 iterations to craft more effective perturbations. DeepFool automatically adjusts its perturbations based on the decision boundary, requiring no predefined ϵ. Lastly, the C&W attack is performed as a targeted optimization-based method, using a confidence parameter of 0.3 to 0.4 and 10 optimization steps to generate high-confidence adversarial examples with minimal visibility.

Table 3 summarizes the evaluation metrics of the models following the application of various adversarial attacks. On the one hand, in CIC-IDS2017 dataset (see Table 3), we can observe that the DT attack had a catastrophic impact on the DT model, reducing its accuracy to just 0.27%. This drastic performance degradation reveals the model's extreme sensitivity to adversarial perturbations and highlights its lack of robustness in defending against even minimal adversarial noise. Similarly, both FGSM and PGD attacks severely compromised the performance of the LR model, with accuracies dropping to 1.99% and 0.65%, respectively. In contrast, the PyTorch MLP model demonstrated a markedly stronger resistance to these attacks, maintaining accuracies of 78.54% under FGSM and 72.00% under PGD. These results suggest that while linear models are highly susceptible to gradient-based adversarial methods, deeper architectures may incorporate features that inherently mitigate such vulnerabilities, at least to a certain extent. Under the DeepFool attack, LR again showed a dramatic drop in performance, with accuracy falling to 1.62%. Although the PyTorch MLP model performed better in this case, its accuracy still dropped to 65.92%, indicating that even more robust deep learning models remain vulnerable to well-optimized adversarial perturbations designed to subtly manipulate decision boundaries. The C&W attack emerged as the most damaging adversarial method across both models. It reduced the accuracy of the LR model to 2.09%, while the PyTorch MLP model experienced a substantial decline to 55.94%. This significant impact underscores the effectiveness of optimization-based attacks like C&W, which can exploit even robust architectures through targeted and precise adversarial noise. Across all attack types, both CNN and RNN models experience a dramatic drop in accuracy, often below 1% in the case of PGD and DeepFool, indicating that the adversarial inputs effectively deceive the models. Precision, recall, and F1-scores are similarly low, especially for PGD and DeepFool, with F1-scores dropping to as low as 0.18% for DeepFool on the CNN and 0.34% for PGD on the RNN. The FGSM attack shows slightly better detection metrics, particularly for the RNN (F1-score

of 23.87%), but still reflects poor overall performance. These metrics clearly highlight that both CNNs and RNNs, without adversarial defenses, are highly susceptible to even basic adversarial attacks, resulting in severe degradation of their classification capabilities.

On the other hand, in CICIoT2023 dataset (see Table 3), we can observe that DT Attack achieved again to dramatically decrease accuracy of DT model at 0.02%. Then, LR's accuracy was also decreased being attacked from FGSM, PGD, C&W and Deepfool. While, we mention that the biggest impact was noted against PGD reaching 11.24%. Next, PyTorch MLP model was evaluated against all attacks FGSM, PGD, C&W and Deepfool. All achieved to impact the model. The biggest impact achieved by FGSM attack while the less from the C&W achieving 27.13% and 86.57% accuracy correspondingly. CNN assessed against FGSM, PGD and DeepFool. The latter achieved the biggest impact with 2.09%, while PGD the less at 20.76% accuracy. Finally, RNN evaluated against FGSM, PGD and DeepFool. PGD achieved the biggest impact with 11.53% and FGSM the less impact at 13.87% accuracy. The relatively strong performance of the PyTorch MLP model under adversarial conditions (e.g., 78.54% accuracy under FGSM and 55.94% under C&W) can be attributed to its architectural capacity to learn complex, non-linear feature representations. Unlike traditional models with rigid or shallow decision boundaries, deep networks extract hierarchical abstractions that enable more flexible and robust classification. This allows the model to partially resist subtle perturbations, particularly in high-dimensional feature spaces such as those present in network traffic data.

Overall, it is observed that DeepFool performed the most severe impact on the CNN model across both CIC-IDS2017 and CICIoT2023 datasets. This can be reasoned due to the attack's unique approach and the inherent vulnerability of CNNs to small, well-targeted perturbations. DeepFool computes the minimal perturbation required to push an input across the decision boundary. As a high-capacity model, CNN often develops sharp decision boundaries in high-dimensional feature spaces, making it particularly susceptible to minimal perturbations that are carefully aligned with the gradients of the network, as DeepFool does. Moreover, since the used CNN is more sensitive to feature-level distortions in such data, DeepFool's ability to exploit subtle vulnerabilities with precision leads to drastic performance degradation, as seen by near-zero accuracy and recall in both datasets. This illustrates that CNNs, while powerful, can be critically destabilized by attacks that finely adapt to the model's geometry, as DeepFool does. Also, PyTorch MLP model consistently exhibits the highest resilience to adversarial attacks, regardless of the attack type or dataset. Generally speaking, PyTorch MLP model likely has a simpler and more regularized structure than CNN. This makes it less prone to overfitting, reducing its sensitivity to small adversarial perturbations. Simpler models often generalize better in the presence of noise, particularly in structured data domains like these. Also, PyTorch MLP model treats all input features in a flat, uniform manner. This homogeneous feature processing reduces the chance that perturbations targeting localized dependencies (like in CNNs) or sequential dependencies (like in RNNs) will drastically alter the output.

Table 4
Transfer Attack Evaluation Metrics in CIC-IDS2017

Performed Attack	Target Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
DT Attack (DT)	DT	0.27	22.29	0.18	0.07
DT Attack (DT)	RF	99.28	99.28	98.25	98.75
DT Attack (DT)	XGBoost	90.52	65.84	66.57	66.16
DT Attack (DT)	LR	92.8	78.14	76.64	75.14
DT Attack (DT)	PyTorch MLP model	98.1	97.96	83.72	85.21
FGSM (LR)	DT	1.09	22.51	1.18	0.3
FGSM (LR)	RF	28.33	83.63	5.01	5.08
FGSM (LR)	XGBoost	0.63	15.7	0.26	0.32
FGSM (LR)	LR	1.99	2.85	4.29	2.65
FGSM (LR)		12.41	19.9	4.06	4.28
PGD (LR)	DT	1.14	55.85	1.26	0.32
PGD (LR)	RF	27.94	83.62	5.07	5.05
PGD (LR)	XGBoost	0.6	18.14	0.47	0.71
PGD (LR)	LR	1.98	2.78	4.02	2.53
PGD (LR)	PyTorch MLP model	11.54	22.07	4.29	4.64
DeepFool (LR)	DT	4.34	25.63	6.32	5.82
DeepFool (LR)	RF	87.06	94.24	25.83	30
DeepFool (LR)	XGBoost	12.24	34.87	11.2	14.25
DeepFool (LR)	LR	1.62	3.24	3.81	3.01
DeepFool (LR)	PyTorch MLP model	98.85	97.57	94.04	95.66
C&W (LR)	DT	3.3	36.69	13.73	7.75
C&W (LR)	RF	74.39	91.16	13.19	12.98
C&W (LR)	XGBoost	2.21	34.32	11.29	5.91
C&W (LR)	LR	2.09	4.82	4.52	3.51
C&W (LR)	PyTorch MLP model	98.22	98.51	94.49	96.41

FGSM (PyTorch MLP model)	DT	48.95	29.27	9.31	11.02
FGSM (PyTorch MLP model)	RF	73.22	86.09	11.2	9.63
FGSM (PyTorch MLP model)	XGBoost	52.37	9.08	8.75	8.84
FGSM (PyTorch MLP model)	LR	32.92	22.79	44.82	20.65
FGSM (PyTorch MLP model)	PyTorch MLP model	78.57	33.7	22.71	24.15
PGD (PyTorch MLP model)	DT	24.75	15.17	10.84	8.09
PGD (PyTorch MLP model)	RF	73.21	86.04	11.11	9.41
PGD (PyTorch MLP model)	XGBoost	47.74	10.89	17.46	10.17
PGD (PyTorch MLP model)	LR	26.41	21.88	34.86	16.49
PGD (PyTorch MLP model)	PyTorch MLP model	78.09	37.86	20.92	22.45
DeepFool (PyTorch MLP model)	DT	42.8	15.26	13.13	11.29
DeepFool (PyTorch MLP model)	RF	73.23	86.39	11.11	9.4
DeepFool (PyTorch MLP model)	XGBoost	63.53	18.65	10.56	10.72
DeepFool (PyTorch MLP model)	LR	18.77	15.08	23.59	10.89
DeepFool (PyTorch MLP model)	PyTorch MLP model	69.93	20.01	11.05	9.87
C&W (PyTorch MLP model)	DT	34.25	14.76	13.43	9.5
C&W (PyTorch MLP model)	RF	74.78	92.09	14.46	15.76
C&W (PyTorch MLP model)	XGBoost	57.17	23.11	17.42	14.12
C&W (PyTorch MLP model)	LR	49.7	14.31	19.53	13.17
C&W (PyTorch MLP model)	PyTorch MLP model	55.94	33.18	17.83	18.76
DT Attack	CNN	0.74	65.63	0.74	0.73
FGSM (LR)	CNN	0.89	14.88	0.89	1.00
PGD (LR)	CNN	0.79	15.16	0.79	0.85
DeepFool (LR)	CNN	0.14	22.12	0.14	0.04
C&W (LR)	CNN	0.14	18.19	0.14	0.04
FGSM (PyTorch MLP model)	CNN	0.94	24.21	0.94	0.74
PGD (PyTorch MLP model)	CNN	0.57	31.48	0.57	0.44
DeepFool (PyTorch MLP model)	CNN	1.14	16.75	1.14	1.95
C&W (PyTorch MLP model)	CNN	3.33	64.11	3.33	5.60
FGSM (CNN)	CNN	1.41	36.53	1.41	0.76
PGD (CNN)	CNN	0.21	4.02	0.21	0.09
DeepFool (CNN)	CNN	17.12	73.71	17.12	27.35
FGSM (RNN)	CNN	0.79	71.26	0.79	1.09
PGD (RNN)	CNN	1.14	72.93	1.14	1.63
DT Attack	(RNN)	97.5	97.71	97.5	97.54
FGSM (LR)	(RNN)	15.13	28.74	15.13	19.24
PGD (LR)	(RNN)	15.25	28.63	15.25	19.27
DeepFool (LR)	(RNN)	80.24	78.59	80.24	78.71
C&W (LR)	(RNN)	87.04	85.75	87.04	85.34
FGSM (PyTorch MLP model)	(RNN)	64.9	71.41	64.9	67.13
PGD (PyTorch MLP model)	(RNN)	63.59	72.95	63.59	65.83
DeepFool (PyTorch MLP model)	(RNN)	28.53	69.24	28.53	38.3
C&W (PyTorch MLP model)	(RNN)	53.16	63.31	53.16	55.46
FGSM (CNN)	(RNN)	50.07	55.08	50.07	52.26
PGD (CNN)	(RNN)	53.97	57.5	53.97	55.56
DeepFool (CNN)	(RNN)	3.44	10.24	3.44	4.37
FGSM (RNN)	(RNN)	18.94	33.18	18.94	23.87
PGD (RNN)	(RNN)	0.24	0.67	0.24	0.34

Table 5
Transfer Attack Evaluation Metrics in CICIoT2023

Performed Attack	**Target Model**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-Score (%)**
DT Attack	DT	0.02	78.59	0.02	0.01
DT Attack	XGBoost	85.98	86.05	85.98	83.96
DT Attack	RF	82.02	83.69	82.02	81.79
DT Attack	LR	86.06	88.44	86.06	85.01
DT Attack	PyTorch MLP model	86.24	88.48	86.24	83.65
DT Attack	CNN	8.42	72.61	8.42	1.89
DT Attack	RNN	85.84	88.16	85.84	83.61
FGSM (LR)	DT	25.74	15.19	25.74	18.32
FGSM (LR)	XGBoost	34.88	23.55	34.88	25.31
FGSM (LR)	RF	60.89	63.79	60.89	57.57
FGSM (LR)	LR	17.85	43.81	17.85	11.63
FGSM (LR)	PyTorch MLP model	51.99	59.11	51.99	50.67
FGSM (LR)	CNN	10.74	66.37	10.74	2.62
FGSM (LR)	RNN	28.84	51.8	28.84	23.85
PGD (LR)	DT	25.85	15.29	25.85	18.51
PGD (LR)	XGBoost	37.48	27.09	37.48	28.42
PGD (LR)	RF	65.26	71.09	65.26	61.06
PGD (LR)	LR	13.28	41.08	13.28	7.02
PGD (LR)	PyTorch MLP model	58.65	67.7	58.65	56.89
PGD (LR)	CNN	10.9	72.99	10.9	2.6
PGD (LR)	RNN	34.95	55.2	34.95	33.49
DeepFool (LR)	DT	28.61	19.61	28.61	20.77
DeepFool (LR)	XGBoost	35.84	32.79	35.84	30.63

DeepFool (LR)	RF	40.66	47.12	40.66	37.14
DeepFool (LR)	LR	13.78	21.11	13.78	9.9
DeepFool (LR)	PyTorch MLP model	86.18	88.32	86.18	83.68
DeepFool (LR)	CNN	7.25	72.63	7.25	1.73
DeepFool (LR)	RNN	84.56	87.81	84.56	82.17
C&W (LR)	DT	24.03	52.07	24.03	15.05
C&W (LR)	XGBoost	37.02	64.61	37.02	28.11
C&W (LR)	RF	68.21	76.98	68.21	67.02
C&W (LR)	LR	12.9	24.07	12.9	6.94
C&W (LR)	PyTorch MLP model	85	85.88	85	83.14
C&W (LR)	CNN	7.34	72.67	7.34	1.84
C&W (LR)	RNN	71.91	80.37	71.91	71.16
FGSM (PyTorch MLP model)	DT	25.64	15.89	25.64	15.75
FGSM (PyTorch MLP model)	XGBoost	36.87	24.43	36.87	27.02
FGSM (PyTorch MLP model)	RF	37.57	63.23	37.57	34.76
FGSM (PyTorch MLP model)	LR	37.44	53.46	37.44	29.11
FGSM (PyTorch MLP model)	PyTorch MLP model	27.13	24.41	27.13	21.49
FGSM (PyTorch MLP model)	CNN	7.62	72.57	7.62	1.49
FGSM (PyTorch MLP model)	RNN	28.49	41.65	28.49	18.27
PGD (PyTorch MLP model)	DT	65.58	69.65	65.58	65.36
PGD (PyTorch MLP model)	XGBoost	67.52	69.15	67.52	64.23
PGD (PyTorch MLP model)	RF	78.59	83.77	78.59	73.16
PGD (PyTorch MLP model)	LR	84.36	83.4	84.36	83.1
PGD (PyTorch MLP model)	PyTorch MLP model	84.43	84.04	84.43	82.14
PGD (PyTorch MLP model)	CNN	2.9	66.27	2.9	0.58
PGD (PyTorch MLP model)	RNN	77.2	76.47	77.2	72.02
DeepFool (PyTorch MLP model)	DT	36.92	32.15	36.92	33.12
DeepFool (PyTorch MLP model)	XGBoost	21.03	21.96	21.03	18.61
DeepFool (PyTorch MLP model)	RF	36.74	37.33	36.74	31.55
DeepFool (PyTorch MLP model)	LR	29.46	36.14	29.46	30.12
DeepFool (PyTorch MLP model)	PyTorch MLP model	41.53	36.03	41.53	35.3
DeepFool (PyTorch MLP model)	CNN	7.98	48.41	7.98	2.24
DeepFool (PyTorch MLP model)	RNN	38.4	32.53	38.4	34.16
C&W (PyTorch MLP model)	DT	86.74	86.77	86.74	85.72
C&W (PyTorch MLP model)	XGBoost	86.18	85.96	86.18	85.82
C&W (PyTorch MLP model)	RF	84.51	84.47	84.51	84.48
C&W (PyTorch MLP model)	LR	87.06	87.96	87.06	85.65
C&W (PyTorch MLP model)	PyTorch MLP model	86.57	88.87	86.57	84.21
C&W (PyTorch MLP model)	CNN	7.32	66.97	7.32	1.67
C&W (PyTorch MLP model)	RNN	86.08	88.14	86.08	84.25
FGSM (CNN)	DT	33.64	58.19	33.64	26.6
FGSM (CNN)	XGBoost	32.05	60.21	32.05	29.12
FGSM (CNN)	RF	77.59	71.22	77.59	72.45
FGSM (CNN)	LR	71.61	72.49	71.61	67.57
FGSM (CNN)	PyTorch MLP model	79.04	80.22	79.04	75.86
FGSM (CNN)	CNN	1.62	66.17	1.62	0.39
FGSM (CNN)	RNN	67.74	67.41	67.74	63.85
PGD (CNN)	DT	33.93	62.27	33.93	25.95
PGD (CNN)	XGBoost	41.34	66.08	41.34	35.87
PGD (CNN)	RF	78.13	76.99	78.13	72.92
PGD (CNN)	LR	75.03	74.43	75.03	69.52
PGD (CNN)	PyTorch MLP model	76.83	78.58	76.83	71.28
PGD (CNN)	CNN	5.95	72.32	5.95	1.42
PGD (CNN)	RNN	74.38	72.28	74.38	70.45
DeepFool (CNN)	DT	38.37	43.37	38.37	38.32
DeepFool (CNN)	XGBoost	34.55	45.5	34.55	30.79
DeepFool (CNN)	RF	50.23	54.07	50.23	46.09
DeepFool (CNN)	LR	29.68	42.44	29.68	34.17
DeepFool (CNN)	PyTorch MLP model	42.77	37.52	42.77	37.56
DeepFool (CNN)	CNN	7.18	65.36	7.18	1.16
DeepFool (CNN)	RNN	42.81	37.72	42.81	35.87
FGSM (RNN)	DT	31.1	18	31.1	21.84
FGSM (RNN)	XGBoost	39.23	26.58	39.23	28.88
FGSM (RNN)	RF	54.31	67.48	54.31	52.78
FGSM (RNN)	LR	26.58	50.95	26.58	18.84
FGSM (RNN)	PyTorch MLP model	56.12	57.13	56.12	54.4
FGSM (RNN)	CNN	7.51	65.78	7.51	1.6
FGSM (RNN)	RNN	13.87	29.93	13.87	6.94
PGD (RNN)	DT	31.6	19.14	31.6	22.14
PGD (RNN)	XGBoost	42.01	35.37	42.01	32.4
PGD (RNN)	RF	59.56	69.95	59.56	57.2
PGD (RNN)	LR	17.71	40.06	17.71	13.03
PGD (RNN)	PyTorch MLP model	55.92	57.18	55.92	54.38
PGD (RNN)	CNN	5.95	66.68	5.95	1.29
PGD (RNN)	RNN	11.53	5.26	11.53	6.75
DeepFool (RNN)	DT	3.79	5.96	3.79	3.85
DeepFool (RNN)	XGBoost	4.52	7.51	4.52	5.4
DeepFool (RNN)	RF	5.37	16.45	5.37	6.75
DeepFool (RNN)	LR	13.63	5.69	13.63	6.41

DeepFool (RNN)	PyTorch MLP model	7.05	3.46	7.05	3.79
DeepFool (RNN)	CNN	7.65	72.84	7.65	3.08
DeepFool (RNN)	RNN	13.42	6.28	13.42	7.29

Beyond direct attacks, we also investigated the transferability of adversarial examples across models (see Table 4 and Table 5). In this setting, adversarial samples crafted for a source model were evaluated on different target models to assess the cross-model generalization of adversarial perturbations. This phenomenon of transferability is especially relevant in black-box scenarios, where attackers do not have access to the target model's parameters or architecture but can still compromise its integrity using surrogate models. The effectiveness of these transfer attacks raises critical concerns about the general robustness of machine learning models and their exposure to real-world adversarial threats. Evaluating transfer attacks across multiple models offers critical insights into their vulnerability to adversarial perturbations. This analysis investigates the effectiveness of various attack strategies on different target models (i.e., DT, RF, XGBoost, LR, and PyTorch MLP model) using performance metrics such as accuracy, precision, recall, and F1-score (see Table 4 and Table 5).

Regarding the dataset CIC-IDS1017 (see Table 4), when the DT attack is applied to its source model, the DT classifier performance degrades drastically, with accuracy and F1-score dropping to 0.27% and 0.07%, respectively. This outcome indicates that the attack is highly effective at misleading the model it was crafted for. However, when transferred to other models, the impact is less uniform. RF remains highly robust, maintaining 99.28% accuracy, while XGBoost and LR exhibit moderate degradation, with accuracies between 90% and 92%. The PyTorch MLP model is similarly resilient, preserving 98.1% accuracy, suggesting that DL models are less affected by this specific attack when not used as the source. The FGSM attack, crafted using a LR model, significantly degrades the performance of DT (1.09% accuracy), yet its effect on other models varies. RF and XGBoost demonstrate relative robustness with 28.33% and 0.63% accuracy, respectively, while the PyTorch MLP model retains partial robustness at 12.41%. Notably, the LR model itself is heavily compromised (1.99% accuracy), confirming the effectiveness of FGSM when targeted at its source architecture. A similar trend is observed with PGD attacks on LR. The performance of DT and XGBoost declines sharply, with accuracies of 1.14% and 0.6%, respectively. Although RF displays greater resistance (27.94% accuracy), it still experiences a notable drop. The PyTorch MLP model, once again, shows moderate resilience, maintaining 11.54% accuracy under this stronger iterative attack.

DeepFool attacks crafted from the LR model are particularly effective against traditional ML models. DT and XGBoost exhibit significant performance degradation, with accuracies dropping to 4.34% and 12.24%, respectively. The LR model itself is also severely impacted (1.62% accuracy). In contrast, RF and PyTorch MLP model demonstrate strong robustness, achieving 87.06% and 98.85% accuracy, respectively. The C&W attack follows a similar pattern. When generated using LR, it causes a considerable drop in performance for both DT (3.3%) and XGBoost (2.21%). However, PyTorch is only marginally affected, maintaining a high accuracy of 98.22%. These findings reinforce the notion that DL models are generally more resilient to adversarial examples crafted on traditional ML models. Interestingly, when adversarial examples are generated using the PyTorch MLP model and transferred to other models, the overall impact is less severe compared to attacks sourced from classical models. Although DT and XGBoost remain vulnerable, the degradation is reduced. PyTorch MLP model itself shows performance degradation under various attacks (FGSM: 78.57%, PGD: 78.09%, DeepFool: 69.93%, C&W: 55.94%), but remains considerably more robust than its traditional counterparts.

The CNN model exhibits significant vulnerability to most transfer attacks, especially those generated using strong gradient-based methods such as DeepFool and C&W, even when transferred from simple surrogate models like LR or PyTorch MLP model. Accuracy under attack often drops to near-zero levels, such as 0.14% for DeepFool and C&W from LR, indicating the CNN is highly susceptible. Despite the low accuracy, recall values remain high (e.g., 0.89 for FGSM), but this comes at the cost of very low precision, suggesting many false positives. The best-performing attack in terms of detection (high F1) seems to be FGSM from LR, which achieves an F1-Score of 1.00, albeit with poor precision (14.88%), highlighting a detection-heavy, error-prone response. Attacks sourced from CNN (e.g., DeepFool (CNN)) lead to slightly better balance, with an F1-Score of 27.35%, still low, but noticeably higher than others. In contrast to the CNN, the RNN demonstrates much stronger resilience against most transfer attacks. The DT-based attack fails to degrade RNN performance, with an accuracy of 97.5%, and very high precision and F1-score, indicating it is ineffective as an attack method. Attacks from LR (e.g., FGSM and PGD) do reduce the accuracy to around 15%, but have low precision and F1-scores, meaning the attacks cause misclassifications but are poorly detectable. Interestingly, DeepFool and C&W attacks from LR are more effective here than on the CNN target, maintaining relatively high F1-scores of 78.71% and 85.34%, respectively. Attacks transferred from PyTorch MLP model and CNN vary in effectiveness,

with FGSM and PGD (PyTorch MLP model) showing moderate impact (F1-scores around 65–67%), and DeepFool (PyTorch MLP model) achieving 38.30%, showing partial success. Attacks from the RNN itself, such as FGSM and PGD, are mostly ineffective, with F1-scores as low as 0.34%, likely due to overfitting or lack of transferability between similar architectures.

Overall, as observed from Table 4, the most effective transfer attack is DeepFool (LR) executed on CNN, where the CNN's performance completely collapses: accuracy drops to 0.14%. This demonstrates extremely high transferability from a simple linear model to a complex deep model because CNNs are highly sensitive to finely crafted perturbations, especially those optimized against decision boundaries, as DeepFool does. Similarly, C&W (LR) on CNN and PGD (LR) on CNN also show near-zero accuracy, confirming CNN as the most vulnerable model to transfer attacks. This is attributed to CNNs' local feature dependencies and lack of input-space regularization, making them prone to spatially distributed adversarial noise. On the other hand, the least effective transfer attack (i.e., the one that fails to deceive the target model) is DT Attack (DT) on RF, where the Random Forest preserves 99.28%, indicating almost no impact. This highlights RF as the most robust model for transfer attacks due to its ensemble nature and resistance to gradient-based perturbations (since it is non-differentiable and built on decision boundaries that do not smoothly shift with input changes).

Regarding the dataset CICIoT2023 (see Table 5), the DT Attack has a devastating impact on the DT model itself, reducing its accuracy to just 0.02%, effectively breaking its predictive capability. The same applies to CNN, which reaches an impact equal to 8.42%. In contrast, other models remain virtually unaffected. XGBoost (85.98%), RF (82.02%), LR (86.06%), PyTorch MLP model (86.24%), and RNN (85.84%) maintain high accuracy, highlighting the highly non-transferable nature of this attack. Next, FGSM crafted from a LR model severely impacts DT (25.74% accuracy) and moderately affects other models. RF (60.89%) and PyTorch MLP model (51.99%) show partial robustness, while CNN (10.74%) and RNN (28.84%) see substantial performance drops. Interestingly, the LR model that generated the attack drops to 17.85% accuracy, confirming FGSM's effectiveness on its own architecture. This attack demonstrates moderate transferability. Furthermore, with PGD crafted using LR, DT (25.85%) and LR (13.28%) are heavily compromised, while RF (65.26%) and PyTorch MLP model (58.65%) exhibit stronger resilience. CNN continues to perform poorly (10.90%), and RNN shows a moderate drop to 34.95%. In addition, DeepFool targeting LR produces mixed results. While DT (28.61%) and LR (13.78%) are significantly affected, PyTorch MLP model (86.18%) and RNN (84.56%) remain largely intact, suggesting high robustness in neural architectures. CNN's performance plummets to 7.25%, consistent with its overall vulnerability.

Also, C&W crafted on LR targets its source model effectively (12.90% accuracy) and moderately affects DT (24.03%) and XGBoost (37.02%). However, RF (68.21%), PyTorch MLP model (85.00%), and RNN (71.91%) resist the attack well. CNN drops again to 7.34%, reinforcing its consistent weakness. Overall, this attack appears less transferable. Moreover, FGSM generated from PyTorch MLP model leads to moderate degradation in DT (25.64%) and XGBoost (36.87%). LR and RNN both drop to around 37.44% and 28.49%, respectively. The most surprising result is the low accuracy of the PyTorch MLP model model itself (27.13%), indicating that FGSM remains highly effective on the source model but less so on others. CNN's performance remains dismal (7.62%), consistent across attacks. PGD from PyTorch MLP model causes widespread performance degradation, especially in shallow models. DT falls to 65.58%, XGBoost to 67.52%, while RF (78.59%) and LR (84.36%) fare better. Surprisingly, PyTorch MLP model retains 84.43%, showing strong robustness to its own PGD attack. CNN's accuracy drops sharply (2.90%), while RNN holds at 77.20%. This attack shows moderate transferability. Also, DeepFool targeting PyTorch MLP model produces variable results. While DT (36.92%) and XGBoost (21.03%) show losses, models such as PyTorch MLP model (41.53%) and RNN (38.40%) suffer less sharply. CNN, however, again struggles (7.98%), as expected. LR and RF maintain middling scores (29.46%, 36.74%). C&W for PyTorch MLP model yields minimal degradation, even against its own model: PyTorch retains 86.57% accuracy. LR (87.06%), RNN (86.08%), and XGBoost (86.18%) also hold strong. Even DT (86.74%) resists this attack—a stark contrast from other cases. CNN, in keeping with its trend, collapses to 7.32%. The C&W attack from PyTorch MLP model shows very low transferability. Furthermore, FGSM crafted from CNN surprisingly causes notable drops in LR (71.61%) and DT (33.64%), despite being a simpler attack. CNN itself collapses (1.62%), confirming the attack's strength against its own architecture. RF (77.59%) and PyTorch MLP model (79.04%) remain resilient. The attack shows low transferability.

Additionally, PGD from CNN follows the same pattern: devastating CNN (5.95%), mildly hurting DT (33.93%) and XGBoost (41.34%), while leaving other models like RF (78.13%) and PyTorch MLP model (76.83%) relatively intact. This reflects low transferability but confirms CNN's internal vulnerability to gradient-based iterative attacks. DeepFool on CNN slightly affects all models but is most impactful on CNN itself (7.18%), DT (38.37%), and RF

(50.23%). LR and PyTorch MLP model maintain 29.68% and 42.77%, respectively. RNN scores 42.81%, showing this attack's broader spread but moderate depth, with no catastrophic degradation beyond CNN. Also, FGSM from RNN weakens the RNN itself dramatically (13.87%) and causes visible degradation in DT (31.10%), XGBoost (39.23%), and LR (26.58%). PyTorch MLP model and RF fare better, at 56.12% and 54.31%. CNN's score remains very low (7.51%). This reflects moderate transferability with a stronger impact on simpler models. PGD from RNN is slightly destructive. RNN drops to 11.53%, while DT (31.60%), XGBoost (42.01%), and LR (17.71%) are all impacted. PyTorch MLP model (55.92%) and RF (59.56%) show better resilience. Again, CNN performs poorly (5.95%), confirming PGD's consistent effectiveness against RNN and CNN. Finally, With DeepFool from RNN, 5% accuracy has the CNN (7.65%) and RNN itself (13.42%). PyTorch MLP model and LR maintain 7.05% and 13.63%, respectively. Overall, it is observed that DT and LR are more vulnerable, while models like PyTorch MLP model and RNN show greater resilience. Regarding CNN, it consistently performs poorly across nearly all attacks.

Overall, from Table 5, it is observed that the most vulnerable model is CNN, which repeatedly suffers extreme performance degradation across nearly all transfer attacks. This is likely due to CNN's sensitivity to local perturbations and lack of global feature aggregation, making it highly susceptible to adversarial noise. Meanwhile, RF is the most resilient to transfer attacks because it is an ensemble of multiple DTs trained on randomized feature subsets, making it less sensitive to small, coordinated input perturbations. This diversity and lack of gradient flow reduce the effectiveness of gradient-based adversarial attacks. Regarding attack strength, the most effective transfer attack is DeepFool (RNN), which consistently reduces model performance. This demonstrates DeepFool's capacity to generalize adversarial perturbations across architectures. On the other hand, the DT attack is the least effective transfer attack because its perturbations are highly specific to the structure and thresholds of a single DT, making them fail to generalize across different models with diverse architectures and decision boundaries.

3.3. Model's Robustness Re-Evaluation After Training

To mitigate the effects of adversarial attacks, this study adopts *adversarial training* as the primary defense strategy. Adversarial training involves retraining machine learning (ML) models using a dataset augmented with both original and adversarially perturbed inputs. This method exposes the model to malicious samples during training, thereby enhancing its ability to recognize subtle perturbations and improving its overall robustness against future adversarial manipulations. Specifically, a robust deep neural network, implemented in PyTorch MLP model, was retrained using a combination of clean and adversarial examples generated in prior stages. This robust model was subsequently evaluated against the previously crafted adversarial examples to assess its resilience to such attacks.

Figure 1 illustrates the pipeline employed for training and evaluating model robustness against adversarial AI threats. The process begins with acquiring raw data ①, specifically network traffic data from a cybersecurity dataset. Preprocessing steps include the removal of duplicates and incomplete entries, encoding of categorical features, and normalization or standardization to ensure stable and efficient model convergence ②. The dataset is then partitioned into training (③ & ⑥) and testing (④ & ⑤) subsets. The training set is used for model development, while the testing set—kept unseen during training—is reserved for evaluation, helping to prevent overfitting and support generalization.

Using clean training data, five distinct ML models (i.e., RF, LR, DT, XGBoost, and a PyTorch MLP model) were trained to establish baseline performance and assess robustness under adversarial conditions. Each model was designed to classify network traffic as either benign or representative of various attack types. Models were learned from the data by optimizing appropriate classification loss functions. This multi-model approach comprehensively compares traditional ML methods and deep learning architectures in adversarial settings.

Once trained, models were initially evaluated on clean test data using standard metrics such as accuracy, precision, recall, and F1-score. All models exhibited high performance at this stage, as they had not yet been subjected to adversarial perturbations. The pipeline incorporated adversarial attacks to simulate real-world threat scenarios utilizing subtle input perturbations intended to mislead the models while remaining visually imperceptible. Adversarial samples were generated using several well-known evasion techniques, including FGSM, PGD, C&W, DeepFool, and DT attacks. These methods exploit gradients from the model's loss function to craft deceptive inputs capable of inducing misclassifications.

Beyond direct attacks, we also employed a transfer attack strategy, wherein adversarial examples generated using one model were tested against another. This mirrors practical attack scenarios where adversaries lack access to the target model's internal parameters but exploit the transferability property of adversarial samples. All generated adversarial examples (both direct and transferred) were applied to the testing dataset to comprehensively evaluate each model's resilience and identify vulnerabilities.

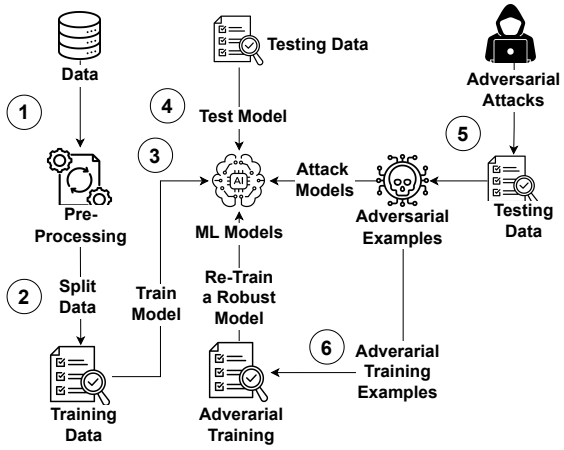

Figure 1: Pipeline for adversarial training

The PyTorch MLP model was retrained in response to these vulnerabilities using adversarial training, incorporating clean and perturbed examples. This retraining process enabled the model to learn to correctly classify adversarial inputs, significantly increasing its robustness. At this point, we have to note that the robust classifier was trained for 30 epochs using Adam optimizer (batch size 128, learning rate 0.001) with cross-entropy loss on true labels (0–8).

Table 6 presents the evaluation metrics of the PyTorch MLP model after adversarial training. Regarding the CIC-IDS2017, As shown, the model maintained high accuracy (above 98%) across most adversarial attacks. Minimal degradation was observed under attacks such as DT, FGSM using LR, PGD (LR), DeepFool (LR), and C&W (LR), with accuracy remaining close to 98.9%. However, the FGSM and PGD attacks implemented in PyTorch MLP model resulted in slightly greater performance deterioration, reducing accuracy to 98.42% and 97.80%, respectively. Next, DeepFool (PyTorch MLP model) was the most effective among all evaluated attacks, causing the model's accuracy to drop to 92.54%. This indicates that DeepFool is particularly adept at generating adversarial examples that compromise model robustness. The C&W (PyTorch MLP model) attack achieved moderate success, reducing accuracy to 97.13%. Although still high, this drop underscores that this variant was more successful in identifying and exploiting vulnerabilities than its LR-based counterpart. Performance metrics for models subjected to adversarial attacks that were crafted using CNN and RNN architectures exhibit high resilience, with accuracy, precision, recall, and F1 scores consistently above 95% across all attack types. Among the CNN-based attacks, FGSM (CNN) has the least impact, preserving 96.2% accuracy, while DeepFool (CNN) is slightly more effective, reducing accuracy to 95.4%. Similarly, for RNN-based attacks, FGSM (RNN) results in the highest performance (96.5% accuracy), while DeepFool (RNN) again causes the largest drop, though marginal, bringing accuracy to 95.6%. Overall, the data suggests that models are highly robust to adversarial examples generated from both CNN and RNN sources, with only minimal degradation observed across all metrics.

Also, regarding the dataset CICIoT2023 (see Table 6) the performance of PyTorch MLP model evaluated under a wide range of adversarial attacks generated from various source architectures, including DT, LR, PyTorch, CNN, and RNN. Across all attacks, the model maintains strong robustness, with accuracy consistently above 85% and F1 scores ranging between 80.82% and 85.57%. The FGSM attack from LR has the highest impact on performance, producing the lowest F1 score (85.57%) among all attacks, despite a relatively high accuracy of 87.02%. In contrast, attacks like PGD (CNN) and DeepFool (CNN) result in slightly lower accuracy (85.49% and 85.08%, respectively), with corresponding drops in F1 scores (81.20% and 80.82%), suggesting marginally greater effectiveness from CNN-based adversarial examples. Attacks from RNN sources (e.g., FGSM (RNN) and PGD (RNN)) are relatively less harmful, with F1 scores reaching up to 84.37%. Overall, the results demonstrate that while adversarial attacks introduce performance degradation, the model shows notable resistance and stability.

All in all, it is observed that the PyTorch MLP model post-adversarial training is robust enough against transfer and itself adversarial attacks regardless of the dataset (see Table 6). The model's robustness stems primarily from the implementation of adversarial training that involves augmenting the training dataset with adversarial examples

Table 6
PyTorch MLP model evaluation after adversarial training

Attack	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
CIC-IDS2017				
DT Attack	98.9	98.91	98.9	98.89
FGSM (LR)	98.83	98.84	98.83	98.82
PGD (LR)	98.85	98.86	98.85	98.84
DeepFool (LR)	98.92	98.93	98.92	98.91
C&W (LR)	98.93	98.95	98.93	98.92
FGSM (PyTorch MLP model)	98.42	98.44	98.42	98.4
PGD (PyTorch MLP model)	97.8	97.83	97.8	97.75
DeepFool (PyTorch MLP model)	92.54	92.18	92.54	91.03
C&W (PyTorch MLP model)	97.13	97.2	97.13	97.06
FGSM (CNN)	96.2	96.0	96.2	96.1
PGD (CNN)	95.8	95.6	95.8	95.7
DeepFool (CNN)	95.4	95.2	95.4	95.3
FGSM (RNN)	96.5	96.3	96.5	96.4
PGD (RNN)	96.0	95.8	96.0	95.9
DeepFool (RNN)	95.6	95.4	95.6	95.5
CICIoT2023				
DT Attack	85.78	86.53	85.78	81.84
FGSM (LR)	87.02	87.89	87.02	85.57
PGD (LR)	86.94	87.79	86.94	85.46
DeepFool (LR)	85.88	88.35	85.88	82.22
CW (LR)	85.89	88.27	85.89	82.27
FGSM (PyTorch MLP model)	86.14	87.50	86.14	84.08
PGD (PyTorch MLP model)	85.51	87.35	85.51	81.38
DeepFool (PyTorch MLP model)	85.17	85.16	85.17	83.22
CW (PyTorch MLP model)	85.91	88.31	85.91	82.27
FGSM (CNN)	85.71	87.47	85.71	82.61
PGD (CNN)	85.49	87.33	85.49	81.20
DeepFool (CNN)	85.08	87.29	85.08	80.82
FGSM (RNN)	86.22	88.22	86.22	83.12
PGD (RNN)	86.14	87.46	86.14	84.37
DeepFool (RNN)	85.86	84.21	85.86	81.96

generated from multiple attack methods. During this process, the model learns to correctly classify clean and perturbed inputs, effectively increasing the decision boundary margin between classes and forcing the model to generalize better. Additionally, training with a cross-entropy loss on true labels helps the model focus on class-specific gradients, and the use of the Adam optimizer with a carefully chosen learning rate (0.001) and batch size (128) supports stable convergence without overfitting to noisy adversarial patterns. Because the adversarial examples used in training come from various sources (e.g., FGSM, PGD, DeepFool), the model develops resistance to diverse gradient-based and optimization-based attacks, including those not explicitly seen during training. This broad exposure helps the model learn more resilient feature representations, ultimately reducing the success rate of transferred and white-box attacks across datasets like CIC-IDS2017 and CICIoT2023.

3.4. Evaluation of the Detection Mechanism

Adversarial attacks against ML models, particularly evasion attacks involving the crafting of malicious inputs, have led to the development of dedicated detection mechanisms. IBM's Adversarial Robustness Toolbox (ART) [14] provides a comprehensive suite of such detection tools. In this work, we focus on two ART detectors designed to identify evasion attacks at runtime: (*i*) *Binary Input Detector*: This is a binary classifier trained to distinguish between clean and adversarial inputs. It uses a CrossEntropyLoss function optimized with the Adam algorithm. The model is trained for 10 epochs with a batch size of 128. The labels for training are binary, where 0 represents clean data and 1 represents adversarial data; (*ii*) Binary Activation Detector: This detector analyzes the activations from an intermediate layer (specifically the 'fc1' layer) of the neural network. A separate, shallow neural network is trained on these intermediate activations, which are approximately 128-dimensional, to detect adversarial behavior based on the network's internal response rather than the raw input. A dedicated neural network is trained using a labeled dataset comprising benign (clean) and malicious (adversarial) samples. Clean inputs are labeled as `0`, while adversarial examples are labeled as `1`. Through this process, the detector learns to recognize statistical patterns that differentiate adversarial inputs from genuine ones. At runtime, the detector examines each incoming input and outputs a binary prediction. If an input is flagged as adversarial (*class 1*), it is blocked or handled via a separate processing pathway, preventing it from reaching

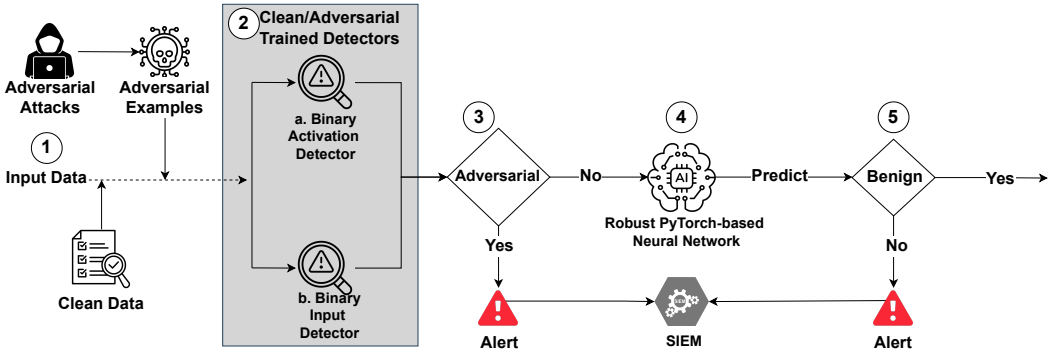

Figure 2: Pipeline after detection implementation

the primary ML model. For example, adversarial images can be intercepted in an image [36] classification scenario before they mislead the classifier.

Binary Activation Detector operates similarly but shifts the detection process from the input space to the internal representation space of the model. Rather than analyzing raw inputs, it utilizes the activation values from one or more hidden layers of the Neural Network (NN) as features for classification. The intuition is that adversarial inputs can induce anomalous activation patterns within the network despite being visually or semantically similar to clean inputs. This detector is trained on activation vectors collected from clean and adversarial examples, labeled accordingly, and used to train a binary classifier [1, 4, 11, 20]. In ART, the user specifies the architecture of the detector and the internal layers to monitor. By leveraging the internal dynamics of the model, this approach can uncover subtle indicators of adversarial behavior that are not observable in the input domain.

Following the initial adversarial robustness evaluation phase (see Figure 1), we extended our ML pipeline to incorporate real-time adversarial detection mechanisms. The resulting system proactively intercepts potential threats before they interact with the core classification model (see Figure 2). Input data ① may originate from two categories: (*i*) *clean* data, representing legitimate network traffic or application behavior, and (*ii*) *adversarial* data, crafted using evasion techniques such as FGSM, PGD, DeepFool, or C&W. To ensure conservative threat mitigation, all incoming inputs are routed through a detection module ② comprising both the Binary Input Detector and the Binary Activation Detector.

The Binary Input Detector processes the raw input features and flags samples that deviate statistically from the distribution of clean data. In parallel, the Binary Activation Detector examines the internal activations elicited by the input and identifies anomalous behavior in the activation space. Both detectors are trained independently using labeled datasets that include clean and adversarial examples. Each outputs a binary decision: *clean* or *adversarial*. If either detector classifies the input as adversarial ③, the system blocks it and triggers an alert. Only inputs classified as clean by both detectors are forwarded to the primary classification model ④, a robust neural network implemented in PyTorch and trained with adversarial training on a mix of clean and adversarial data. Finally ⑤, the model produces a prediction. If the result is categorized as benign, the input is accepted; otherwise, if classified as malicious, an additional alert is generated for security monitoring. This layered defense strategy enhances the resilience of the AI system by intercepting potentially malicious inputs at multiple points, thereby reducing the likelihood of successful evasion attacks.

The performance of two binary detectors (i.e., Input Detector and Activation Detector) was evaluated on two benchmark datasets: CIC-IDS2017 and CICIoT2023, using standard classification metrics: Precision, Recall, and F1-Score (see Table 7). Particularly, for the CIC-IDS2017 dataset, there were 73 total samples, with 12 clean samples and 61 adversarial samples, while for the CICIoT2023 dataset were used 80 total samples, including 8 clean and 72 adversarial samples. On the CIC-IDS2017 dataset, the Input Detector achieved a precision of 73%, recall of 70%, and an F1-score of 72%, indicating a reasonably balanced performance in detecting adversarial inputs. In comparison, the Activation Detector performed slightly worse, with a precision of 67%, recall of 61%, and an F1-score of 64%, suggesting a less reliable detection capability based on intermediate activations for this dataset. For the CICIoT2023 dataset, both detectors showed improved results. The Input Detector achieved a precision of 78%, recall of 80%, and an F1-score of 79%, while the Activation Detector reached a precision of 79%, recall of 71%, and an F1-score of

Table 7
Detectors' performance evaluation

Detector Type	Precision (%)	Recall (%)	F1-Score (%)
CIC-IDS2017			
Binary Input Detector	73	70	72
Binary Activation Detector	67	61	64
CICIoT2023			
Binary Input Detector	78	80	79
Binary Activation Detector	79	71	75

75%. Notably, on this dataset, the Activation Detector showed higher precision but lower recall compared to the Input Detector, indicating it was more conservative in flagging inputs as adversarial but slightly less effective in capturing all adversarial cases. Overall, the Input Detector consistently demonstrated stronger and more balanced performance across both datasets, while the Activation Detector showed promise, particularly on CICIoT2023, though with a trade-off between precision and recall.

To evaluate the effectiveness of each method, the F1 score is used as a metric that considers both precision (the accuracy of positive predictions) and recall (the ability to identify all relevant instances). A higher F1 score indicates better overall performance in correctly detecting adversarial attacks while minimizing false positives and false negatives. When comparing methods, the one with the higher F1 score is generally more effective in maintaining this balance. Thus, adversarial training excels (see Table 6) in building inherent model resilience, while detectors provide a flexible, complementary defense that can be especially valuable in complex or evolving environments.

4. Discussion

This section reflects on the limitations of the current research and outlines future work that could further enhance the resilience of AI-based systems against adversarial attacks.

4.1. Limitations

While this study provides valuable insights into the detection and mitigation of adversarial attacks within AI-based systems, several limitations must be acknowledged. These constraints underscore areas requiring further investigation to enhance such systems' practical applicability and resilience. Spanning dataset scope, model architecture diversity, and the breadth of adversarial techniques considered, these limitations highlight the inherent complexity of developing robust AI defenses in dynamic threat environments. Recognizing these gaps is crucial for guiding future research and advancing adversarial robustness in AI-driven cybersecurity.

This article utilizes the CIC-IDS2017 and CICIoT2023 datasets, both of which are widely recognized benchmarks for intrusion detection and IoT-specific threat modeling. While these datasets offer diverse traffic profiles and attack categories, including DoS, DDoS, brute-force, and botnet traffic, their synthetically generated nature and limited capture environments limit their validity. Real-world network environments often exhibit greater traffic patterns, protocol usage, and user behavior heterogeneity. Additionally, emerging zero-day attacks and complex multi-stage threats prevalent in modern threat landscapes are underrepresented. As a result, the models trained and tested within this controlled scope may not generalize effectively to operational deployments across varied enterprise networks, cloud-based infrastructures, or mobile and edge computing contexts.

Further, the study evaluates a range of ML classifiers, including traditional algorithms (DT, RF, LR, and XGBoost) and DL architectures (RNN, CNN, and a custom PyTorch-based neural network). However, while representative of major algorithmic families, the model selection is not exhaustive. Notably absent are advanced transformer-based models (e.g., Vision Transformers or BERT variants for sequence modeling), hybrid ensembles, and graph-based neural networks, which have shown promise in capturing spatio-temporal patterns and complex dependencies in cybersecurity contexts. Furthermore, ensemble strategies and stacking techniques, which could enhance robustness through model diversity, were not explored due to time and resource constraints. A broader and more nuanced exploration of architectural variants could yield deeper insights into robustness differentials under adversarial stress.

Next, the adversarial evaluation was conducted using four prominent evasion attack techniques: FGSM, PGD, DeepFool, and CW, each representing different gradients of attack complexity and perturbation constraints. However,

the scope did not include more recent attacks tailored for network traffic manipulation (e.g., protocol-aware perturbations or adversarial packet injection). These newer methods often exploit domain-specific vulnerabilities and adaptive strategies that could bypass static defenses. Moreover, poisoning attacks, where adversarial samples are injected during training, were not investigated despite their growing relevance in federated and distributed learning settings. The omission of these attack vectors presents an incomplete picture of the full adversarial landscape deployed systems face.

Adversarial training, while shown to enhance model robustness in this article, incurs a significant computational burden due to the necessity of generating perturbations for each training instance and iteratively updating gradients. The experiments were conducted under constrained hardware conditions, limiting the batch sizes, model complexity, and extent of hyperparameter optimization achievable. For instance, deeper neural networks or transformer-based architectures, which may require extensive GPU resources and prolonged training time, were excluded from evaluation. Additionally, comprehensive grid searches or Bayesian optimization for learning rate, dropout, and weight decay parameters were impractical. These constraints may have hindered the models' ability to achieve optimal generalization performance and resilience under adversarial pressure.

Finally, one possible explanation for the disparity in model robustness lies in the nature of their decision boundaries and feature extraction capabilities. Simpler models like DT or LR rely on linear or axis-aligned splits, which are easily bypassed by small, targeted perturbations. In contrast, deep neural networks develop smoother, more abstract decision boundaries through multi-layer feature transformations. This enables them to preserve inter-class separation more effectively when processing adversarial inputs, as observed in the PyTorch MLP model's consistent performance across multiple attack types.

4.2. Future Work

The following directions outline promising avenues for enhancing the resilience and trustworthiness of AI systems against adversarial manipulation in real-world deployments. The need for more robust and adaptive systems becomes increasingly vital as the cybersecurity landscape evolves. A primary avenue for future work lies in expanding the scope of training data. By incorporating additional datasets that represent a broader variety of network traffic patterns, attack methodologies, and environmental conditions, we can significantly improve the generalizability and robustness of intrusion detection models across diverse real-world environments. This comprehensive data expansion would allow models to better capture the complexities and subtleties inherent in network traffic, thereby strengthening defenses against emerging attack vectors.

Furthermore, exploring the efficacy of more advanced ML architectures is crucial. Ensemble methods, where multiple models collaborate to make predictions, may provide higher accuracy and robustness than single-model approaches. Additionally, deep neural networks (DNNs), with their capacity to model complex, high-dimensional data, present a promising avenue for tackling adversarial manipulation. The strength of deep learning lies in its ability to uncover hidden patterns within large datasets, potentially leading to novel strategies for increasing resistance to adversarial perturbations. Future research should focus on evaluating these advanced architectures' effectiveness, how they can be fine-tuned to resist adversarial attacks, and identifying techniques to enhance their stability and reliability in real-world applications.

Implementing dynamic defense mechanisms that adapt to evolving adversarial tactics in real time remains a critical challenge. As adversaries continue to develop increasingly sophisticated attack strategies, creating systems that evolve in response is essential. Research into online learning algorithms, which enable models to update continuously as new data becomes available, represents a key area for further investigation. When combined with adaptive defense mechanisms, these real-time detection frameworks could significantly enhance a system's ability to recognize and mitigate previously unseen attacks. Moreover, integrating reinforcement learning techniques, where models are trained to adapt based on feedback from their environment, could offer further improvements in detecting and defending against novel threats.

Additionally, the integration of explainable AI (XAI) techniques is critical for improving the interpretability and transparency of AI-based intrusion detection systems, particularly when handling adversarial inputs. As the adoption of AI in security-critical systems continues to grow, developing methods to make these systems more interpretable to human users becomes increasingly important. Gaining insights into model decision-making processes can help uncover vulnerabilities, enabling more effective countermeasures against adversarial attacks. Moreover, improving explainability can foster greater trust in Intrusion Detection Systems (IDS) deployments, as stakeholders will have a

clearer understanding of how decisions are made, an especially important factor when such decisions carry significant security implications.

Finally, a promising direction for future work is the development of hybrid systems that combine multiple defensive strategies, such as integrating traditional security measures with AI-driven models. These systems could provide a multi-layered defense, where each layer complements the others, creating a more robust overall protection mechanism. Such approaches also facilitate the integration of human oversight, enabling security professionals to intervene or adjust system behavior when necessary, further enhancing the adaptability and effectiveness of intrusion detection systems.

5. Related Work

This section presents key scientific contributions and methodologies related to adversarial attacks, focusing on established attack models, detection strategies, and defensive mechanisms.

Chakraborty et al. [6] offered a comprehensive survey on the classification of adversarial attacks (i.e., evasion, poisoning, and exploratory) and corresponding defense techniques such as defensive distillation, input transformation, and adversarial training. The study emphasized ensemble-based defenses to improve model resilience and highlighted the importance of standardized evaluation metrics for ensuring comparability across research efforts. Similarly, Liang et al. [17] provided an overview of foundational adversarial example generation methods, including L-BFGS, FGSM, JSMA, and DeepFool. Their review categorized defense techniques into model-based, data-driven, and auxiliary-network-based approaches. Also, Malik et al. [19] conducted a systematic review emphasizing the risks of minor yet deliberate input perturbations, particularly in critical applications such as autonomous vehicles and medical diagnostics. The authors classified attacks into privacy-based, evasion, and poisoning types, and discussed open-source tools for adversarial robustness evaluation. They proposed countermeasures aimed at enabling the secure deployment of AI systems in sensitive environments. In addition, Bountakas et al. [2] focused on image classifiers and introduced a taxonomy of attacks based on adversary knowledge (white-box, black-box, gray-box), distinguishing between evasion and poisoning techniques. Defense mechanisms were categorized into detection (e.g., feature squeezing, statistical methods) and mitigation (e.g., adversarial training, defensive distillation). The authors discussed evaluation challenges, including limited datasets and key metrics such as Malicious Traffic Evasion Rate and Detection Evasion Rate.

Qiu et al. [24] analyzed the resilience of AI systems by classifying adversarial attacks based on their timing, occurring during either the training or testing phases. Their study spanned various application domains such as physical systems, cybersecurity, computer vision, and natural language processing. Defense strategies were categorized into three main types: model architecture modifications, training/testing data changes, and auxiliary tools. Next, in the context of IDS, Tcydenova et al. [31] proposed a detection framework that integrates explainable AI via LIME with SVM-based classifiers. Their method identifies adversarial traffic by comparing real-time network traffic characteristics against patterns extracted from normal training data. The system effectively detected PGD-generated adversarial examples, demonstrating high accuracy and interpretability. Furthermore, Pantelakis et al. [23] investigated adversarial robustness in IoT networks using machine learning-based multiclass classifiers. Their experiments, conducted on the IoTID20 dataset with attacks including JSMA, FGSM, and DeepFool, revealed that RF models outperformed others in detection accuracy. Adversarial training significantly enhanced classifier robustness, with RF achieving the best post-training performance. Also, [1] proposes an Egyptian Vulture Optimized Adaptive Elman Recurrent Neural Networks (EVO-AERNN) model to assess cybersecurity resilience and compare it with machine learning and deep learning-based classifers. It illustrates the potential of using adversaryaware feature sampling to build more robust classifers and use an optimized algorithm to maintain inherent resilience.

Ryu and Choi [26] introduced an entropy-based method for detecting adversarial attacks on deep neural networks. Their approach, which measures entropy changes after bit-depth reduction, offers a computationally efficient alternative to resource-intensive defenses. Experiments on CIFAR-10 and ImageNet achieved over 98% detection accuracy with low false positive rates. Ciolino et al. [8] explored adversarial attack detection in image classification by identifying the presence of an attack, the targeted model, and the attack type from a single model output. Extending their prior work, they proposed a defense framework tested across six white-box attacks and five pretrained models. While the framework achieved up to 70% accuracy in identifying the attacked model, distinguishing the exact attack type remained challenging (approximately 37% accuracy). Furthermore, [35] studies the challenges posed by the next-generation decision support systems in the era of 5G and big data. To build trust in AI, a saliency map is adopted as a visualization method to reveal the vulnerability of neural networks. The visualization method is further taken to identify imperceptible adversarial samples and reasons for the misclassification of high-accuracy models.

Also, Jia et al. [15] introduced a novel attack detection framework based on the MDATA model leveraging AI for enhanced cybersecurity in smart cities. The proposed framework, ACAM, aims to detect and respond to cyber threats, fortifying the resilience of smart city cyber infrastructures. Also, [16] proposes an AI-based control framework leveraging adversarial deep learning to enhance STVSA in power systems. Its primary motivation was to strengthen existing methods against adversarial cyber-attacks, thereby ensuring the reliability and stability of power system operations.

6. Conclusion

This article investigated the integration of adversarial detection mechanisms into ML pipelines to enhance the robustness of intrusion detection systems against advanced evasion strategies. Using the CIC-IDS2017 and CICIoT2023 datasets, we trained and evaluated a variety of ML models, including DT, RF, LR, XGBoost, CNN, RNN, and a custom neural network implemented in PyTorch. While these models exhibited strong baseline performance, they demonstrated notable vulnerabilities when exposed to adversarial perturbations generated by FGSM, PGD, DeepFool, and C&W attacks as well as transfer adversarial attacks across models. To mitigate these vulnerabilities, adversarial training was employed by augmenting the training data with adversarial examples derived from the aforementioned attack methods. This approach significantly enhanced the resilience of the models, with the PyTorch MLP model maintaining an accuracy exceeding 92.54% under most adversarial conditions in the CIC-IDS2017 dataset and more than 85.08% in the CICIoT2023 dataset. Nevertheless, specific attacks, particularly DeepFool and C&W, continued to cause performance degradation, even in the robust PyTorch MLP model. Apart from the adversarial training, we deployed and trained binary input and activation detectors to detect adversarial attacks. The results highlighted that the binary input detector performed better regardless of the dataset. Overall, the adversarial training outperformed detectors regardless of datasets for defending against adversarial attacks. The presented results underscore the persistent challenge of defending against adaptive adversarial techniques and highlight the critical need for continued research into more effective and generalizable defense mechanisms for ML-based intrusion detection systems.

Acknowledgements

This research has received funding from the European Commission's Horizon Europe and DIGITAL research and innovation programs under grant agreements No. 101131292 (AIAS), No. 101183162 (ANTIDOTE), No. 101168407 (cPAID), No. 101145872 (NITRO), and No. 101120962 (RESCALE).

Declarations

Competing Interest Statement: The authors declare no conflict of interest.

Research Data/Code Availability: The source code will be available upon request.

Compliance with Ethical Standards: This article does not contain any studies with human participants or animals performed by any of the authors.

References

[1] Barik, K., Misra, S., Fernandez-Sanz, L., 2024. A model for estimating resiliency of ai-based classifiers defending against cyber attacks. International Journal of Computational Intelligence Systems 17, 290.

[2] Bountakas, P., Zarras, A., Lekidis, A., Xenakis, C., 2023. Defense strategies for adversarial machine learning: A survey. Computer Science Review 49, 100573.

[3] Brendel, W., Rauber, J., Bethge, M., 2017. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. arXiv preprint arXiv:1712.04248 .

[4] Carlini, N., Wagner, D., 2017a. Adversarial examples are not easily detected: Bypassing ten detection methods, in: Proceedings of the 10th ACM workshop on artificial intelligence and security, pp. 3–14.

[5] Carlini, N., Wagner, D., 2017b. Towards evaluating the robustness of neural networks, in: 2017 ieee symposium on security and privacy (sp), Ieee. pp. 39–57.

[6] Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., Mukhopadhyay, D., 2021. A survey on adversarial attacks and defences. CAAI Transactions on Intelligence Technology 6, 25–45.

[7] Chen, P.Y., Zhang, H., Sharma, Y., Yi, J., Hsieh, C.J., 2017. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models, in: Proceedings of the 10th ACM workshop on artificial intelligence and security, pp. 15–26.

[8] Ciolino, M., Kalin, J., Noever, D., 2021. Fortify machine learning production systems: Detect and classify adversarial attacks. arXiv preprint arXiv:2102.09695 .
[9] Dietterich, T.G., et al., 2002. Ensemble learning. The handbook of brain theory and neural networks 2, 110–125.
[10] Dong, X., Yu, Z., Cao, W., Shi, Y., Ma, Q., 2020. A survey on ensemble learning. Frontiers of Computer Science 14, 241–258.
[11] Feinman, R., Curtin, R.R., Shintre, S., Gardner, A.B., 2017. Detecting adversarial samples from artifacts. arXiv preprint arXiv:1703.00410 .
[12] Felix, E.A., Lee, S.P., 2019. Systematic literature review of preprocessing techniques for imbalanced data. Iet Software 13, 479–496.
[13] Huang, H.J., Iskandarov, B., Rahman, M., Otal, H.T., Canbaz, M.A., 2024. Federated learning in adversarial environments: Testbed design and poisoning resilience in cybersecurity. arXiv preprint arXiv:2409.09794 .
[14] IBM, 2023. Adversarial Robustness Toolbox (ART. https://research.ibm.com/projects/adversarial-robustness-toolbox. Online; Last Accessed: 04/2025.
[15] Jia, Y., Gu, Z., Du, L., Long, Y., Wang, Y., Li, J., Zhang, Y., 2023. Artificial intelligence enabled cyber security defense for smart cities: A novel attack detection framework based on the mdata model. Knowledge-Based Systems 276, 110781.
[16] Li, Y., Zhang, S., Li, Y., 2025. Ai-enhanced resilience in power systems: Adversarial deep learning for robust short-term voltage stability assessment under cyber-attacks. Chaos, Solitons & Fractals 196, 116406.
[17] Liang, H., He, E., Zhao, Y., Jia, Z., Li, H., 2022. Adversarial attack and defense: A survey. Electronics 11, 1283.
[18] Maharana, K., Mondal, S., Nemade, B., 2022. A review: Data pre-processing and data augmentation techniques. Global Transitions Proceedings 3, 91–99.
[19] Malik, J., Muthalagu, R., Pawar, P.M., 2024. A systematic review of adversarial machine learning attacks, defensive controls and technologies. IEEE Access .
[20] Metzen, J.H., Genewein, T., Fischer, V., Bischoff, B., 2017. On detecting adversarial perturbations. arXiv preprint arXiv:1702.04267 .
[21] Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P., 2016. Deepfool: a simple and accurate method to fool deep neural networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2574–2582.
[22] Neto, E.C.P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., Ghorbani, A.A., 2023. Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment. Sensors 23, 5941.
[23] Pantelakis, V., Bountakas, P., Farao, A., Xenakis, C., 2023. Adversarial machine learning attacks on multiclass classification of iot network traffic, in: Proceedings of the 18th International Conference on Availability, Reliability and Security, pp. 1–8.
[24] Qiu, S., Liu, Q., Zhou, S., Wu, C., 2019. Review of artificial intelligence adversarial attack and defense technologies. Applied Sciences 9, 909.
[25] Raghunathan, A., Steinhardt, J., Liang, P., 2018. Certified defenses against adversarial examples. arXiv preprint arXiv:1801.09344 .
[26] Ryu, G., Choi, D., 2024. Detection of adversarial attacks based on differences in image entropy. International Journal of Information Security 23, 299–314.
[27] Sagi, O., Rokach, L., 2018. Ensemble learning: A survey. Wiley interdisciplinary reviews: data mining and knowledge discovery 8, e1249.
[28] Santos, C.F.G.D., Papa, J.P., 2022. Avoiding overfitting: A survey on regularization methods for convolutional neural networks. ACM Computing Surveys (Csur) 54, 1–25.
[29] Shafahi, A., Najibi, M., Ghiasi, M.A., Xu, Z., Dickerson, J., Studer, C., Davis, L.S., Taylor, G., Goldstein, T., 2019. Adversarial training for free! Advances in neural information processing systems 32.
[30] Steinhardt, J., Koh, P.W.W., Liang, P.S., 2017. Certified defenses for data poisoning attacks. Advances in neural information processing systems 30.
[31] Tcydenova, E., Kim, T.W., Lee, C., Park, J.H., 2021. Detection of adversarial attacks in ai-based intrusion detection systems using explainable ai. Human-Centric Comput Inform Sci 11.
[32] Tramèr, F., Papernot, N., Goodfellow, I., Boneh, D., McDaniel, P., 2017. The space of transferable adversarial examples. arXiv preprint arXiv:1704.03453 .
[33] University of New Brunswick, . Intrusion detection evaluation dataset (CIC-IDS2017). https://www.unb.ca/cic/datasets/ids-2017.html. Online; Last Accessed: 04/2025.
[34] Wong, E., Rice, L., Kolter, J.Z., 2020. Fast is better than free: Revisiting adversarial training. arXiv preprint arXiv:2001.03994 .
[35] Zhang, D., Gu, Z., Ren, L., Shafiq, M., 2023. An interpretability security framework for intelligent decision support systems based on saliency map. International Journal of Information Security 22, 1249–1260.
[36] Zhang, D., Shafiq, M., Srivastava, G., Gadekallu, T.R., Wang, L., Gu, Z., 2024. Stbciot: Securing the transmission of biometric images in customer iot. IEEE Internet of Things Journal 11, 16279–16288.