

Robust CNN-Based Network Intrusion Detection Against Adversarial Attacks Using Autoencoder Input Denoising

Gibrilla Deen Kamara

gibrilladeen3539@gmail.com

University of Electronic Science and Technology of China (UESTC) <https://orcid.org/0009-0005-1732-6407>

Professor Xiaofen Wang

University of Electronic Science and Technology of China (UESTC)

Marrah Saio Alusine

University of Electronic Science and Technology of China (UESTC) <https://orcid.org/0009-0008-4727-2480>

Selorm Kweku Dzokoto

University of Electronic Science and Technology of China (UESTC) <https://orcid.org/0009-0003-1230-0414>

Research Article

Keywords: Network Intrusion Detection System (NIDS), Convolutional Neural Network (CNN), Adversarial Attacks, Autoencoder Denoising, Deep Learning Robustness

Posted Date: May 6th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-9619720/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.
[Read Full License](#)

Additional Declarations: The authors declare potential competing interests as follows: The authors declare that they have no competing financial or non-financial interests related to this work.

Robust CNN-Based Network Intrusion Detection Against Adversarial Attacks Using Autoencoder Input Denoising

Abstract

Convolutional Neural Network (CNN)-based intrusion detection systems are accurate but fragile, adversarial perturbations that no human analyst would notice are enough to flip their classifications from malicious to benign. This paper's answer is to place a denoising autoencoder in front of the CNN, trained only on clean traffic, so adversarial inputs get pulled back toward normal behavior before the classifier ever touches them. Tested on CIC-IDS2017 Canadian Institute for Cybersecurity Intrusion Detection System 2017, Communications Security Establishment – Canadian Institute for Cybersecurity Intrusion Detection System 2018 (CSE-CIC-IDS2018), and Network Security Laboratory – Knowledge Discovery and Data Mining (NSL-KDD) under Fast Gradient Sign Method (FGSM) Projected Gradient Descent (PGD), and Carlini and Wagner (C&W) attacks, the framework reaches 95.1% clean accuracy and 76.2% robust accuracy under PGD, outperforming an undefended baseline by 58 percentage points and adversarial training by 28. The t-distributed Stochastic Neighbor Embedding (t-SNE) results show adversarial samples moving 68.3% closer to the clean data manifold after reconstruction, which is geometric proof that the denoising step is doing exactly what the authors claim. The framework's real value is that it delivers both strong clean performance and strong robustness simultaneously, something adversarial training alone has consistently failed to do.

Keywords: Network Intrusion Detection System (NIDS), Convolutional Neural Network (CNN), Adversarial Attacks, Autoencoder Denoising, Deep Learning Robustness

1. Introduction

The proliferation of network-connected devices and the escalating sophistication of cyber threats have rendered network security a paramount concern for organizations worldwide. Network Intrusion Detection Systems (NIDS) serve as a critical line of

defense, tasked with monitoring network traffic for malicious activities and policy violations [1], [2]. Traditional signature-based NIDS, while effective against known threats, struggle to detect novel or polymorphic attacks, necessitating more adaptive and intelligent solutions [3], [4]. In response, the research community has increasingly turned to machine learning, and more recently, deep learning (DL) techniques. Deep learning, particularly Convolutional Neural Networks (CNNs), has demonstrated remarkable success in NIDS due to its inherent ability to automatically learn hierarchical spatial features from raw or minimally processed data, eliminating the need for extensive manual feature engineering [5], [6]. CNNs can capture complex patterns and anomalies within network traffic flows, achieving superior detection accuracy and scalability compared to conventional machine learning approaches, thereby significantly advancing the state-of-the-art in intrusion detection [7], [8].

Despite their exceptional performance, deep learning models, including CNNs, are inherently vulnerable to a class of sophisticated exploits known as adversarial attacks. Adversarial attacks involve the deliberate crafting of inputs, termed adversarial examples, by adding small, often imperceptible, perturbations to legitimate network traffic data [9], [10]. These perturbations, while minimal, are strategically designed to mislead the DL model at its core, causing it to produce incorrect classifications with high confidence [11]. In the context of cybersecurity, this vulnerability presents a critical and emerging threat. An adversary could potentially evade detection by subtly manipulating malicious network packets, causing a NIDS to misclassify them as benign [12], [13]. The existence of such vulnerabilities undermines the reliability and trustworthiness of deep learning-based security systems, creating a significant gap between their theoretical potential and their practical, real-world deployment [14].

The fundamental problem addressed in this paper is the pronounced lack of robustness in CNN-based Network Intrusion Detection Systems when confronted with sophisticated adversarial perturbations. Current state-of-the-art NIDS models, while achieving high accuracy on clean, unperturbed network traffic, exhibit a dramatic degradation in performance when evaluated against adversarial examples [15], [16]. This susceptibility creates a dangerous security loophole, as attackers with knowledge of the system's vulnerabilities can exploit this weakness to bypass security measures [17]. Existing defense mechanisms, such as adversarial training, often come with a

high computational cost and may still be circumvented by more powerful, iterative attacks, or they may negatively impact the model's accuracy on benign traffic [18], [19]. Therefore, there is an urgent need for a robust, efficient, and generalizable defense strategy that can effectively mitigate the impact of adversarial attacks without compromising the high detection performance on legitimate network traffic.

To address this critical vulnerability, this paper proposes a novel defense framework designed to enhance the adversarial robustness of CNN-based NIDS. The core of our solution lies in a two-stage architecture that integrates an autoencoder-based input denoising preprocessor with a standard CNN classifier [20], [21]. The first stage employs a denoising autoencoder (DAE), an unsupervised deep learning model, trained to reconstruct clean network traffic samples from corrupted inputs. The fundamental principle is that the autoencoder will learn the underlying manifold of legitimate network data [22]. When an adversarial sample is presented, the autoencoder acts as a preprocessing filter, projecting it back onto this manifold, thereby removing the malicious perturbations before the data is passed to the classifier [23]. The second stage consists of the pre-trained CNN classifier, which then performs intrusion detection on the reconstructed, "denoised" traffic. By effectively cleansing the input data, this framework aims to neutralize adversarial perturbations at the very point of entry, ensuring that the subsequent classifier is presented with a representation that closely aligns with the benign data it was trained on [24].

The primary contributions of this work are as follows:

1. **Novel Defense Architecture:** We propose a novel and effective defense framework that synergistically combines a denoising autoencoder as a pre-processing stage with a CNN classifier to defend against adversarial attacks in network intrusion detection. This integrated approach provides a robust shield against adversarial perturbations at the input level [25].
2. **Comprehensive Empirical Analysis:** We conduct a thorough empirical evaluation of the proposed framework using well-established benchmark NIDS datasets such as CIC-IDS2017 and CSE-CIC-IDS2018. The evaluation measures the model's performance under various strong adversarial attack methods, including the Fast Gradient Sign Method (FGSM), Projected Gradient Descent

(PGD), and the Carlini & Wagner (C&W) attack, to demonstrate the defense's effectiveness against different threat models [26], [27].

3. **Comparative Performance Assessment:** The performance of our proposed defense is rigorously compared against baseline models, such as a vanilla CNN with no defense, as well as other established defense techniques. This comparison highlights the improvements in both adversarial robustness (e.g., reduced Attack Success Rate) and the preservation of high accuracy on clean data [28].
4. **In-depth Analysis of Denoising Efficacy:** We provide a detailed analysis of the autoencoder's denoising capabilities, examining how it modifies adversarial inputs in feature space to mitigate the impact of noise, thus offering insights into the underlying mechanism of the defense [29].

The remainder of this paper is structured as follows. Section 2 provides a comprehensive review of related work, discussing deep learning for NIDS, the nature of adversarial attacks in cybersecurity, existing defense mechanisms, and identifying the research gap that motivates this study. Section 3 details the proposed methodology, including a system overview, threat model, dataset descriptions, and the architectural specifics of the autoencoder denoiser and the CNN classifier. Section 4 outlines the experimental setup, including implementation details, evaluation metrics, and baseline comparisons. The experimental results and a detailed discussion of the findings are presented in Section 5. Finally, Section 6 concludes the paper with a summary of our contributions and suggestions for future research directions.

2. Related Work

2.1. Deep Learning for Network Intrusion Detection

The application of deep learning to network intrusion detection has garnered significant interest due to its capacity to overcome the limitations of traditional machine learning methods [30], [31]. Early machine learning approaches, such as Support Vector Machines and Random Forests, rely heavily on handcrafted features, which can be a time-consuming and error-prone process that may not capture complex, high-dimensional patterns in network traffic [32], [33]. Deep learning models, particularly Convolutional Neural Networks (CNNs), have demonstrated a

superior ability to perform automated feature extraction, learning intricate spatial hierarchies directly from raw or semi-processed data [34], [35]. CNNs have been successfully employed to model network flows as multi-dimensional data structures, enabling them to discern subtle anomalies indicative of malicious activity [36]. Beyond CNNs, other architectures like Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTMs) have been utilized to capture temporal dependencies in sequential traffic data [37], [38]. More recent works have explored hybrid models, such as combining CNNs for spatial feature extraction with LSTMs for temporal sequence modeling, to further enhance detection capabilities [39], [40]. Despite their high performance, these deep learning models are often evaluated under ideal conditions, with a growing body of research now highlighting their significant vulnerability to adversarial attacks, a critical limitation that this paper seeks to address [41].

2.2. Adversarial Attacks in Cybersecurity

Adversarial attacks represent a sophisticated class of threats that exploit the fundamental vulnerabilities of machine learning models [42]. In the context of NIDS, these attacks are typically evasion attacks, crafted during the testing phase to deceive a trained model into misclassifying malicious traffic as benign [43], [44]. Several powerful attack generation methods have become standard benchmarks for evaluating model robustness. The Fast Gradient Sign Method (FGSM) is a single-step, gradient-based attack that adds noise in the direction of the model's gradient, creating adversarial examples with minimal computational overhead [45]. Projected Gradient Descent (PGD) is an iterative, more powerful variant of FGSM that takes multiple smaller steps, often considered a benchmark for evaluating the strongest possible attacks [46]. Similarly, the Carlini & Wagner (C&W) attack is a highly effective optimization-based method that is known for generating adversarial examples with very low perturbation levels, making them extremely difficult to detect [47]. Research has shown that these attacks can significantly degrade the performance of NIDS, with PGD capable of reducing accuracy to below 20% in some cases, underscoring the urgent need for robust defenses [48]. These attacks can be executed under different threat models, with white-box attacks assuming full knowledge of the target model,

and black-box attacks, a more realistic scenario, assuming no such internal access [49].

2.3. Defense Mechanisms Against Adversarial Attacks

In response to the threat of adversarial attacks, various defense strategies have been proposed, each with its own strengths and limitations [50]. One of the most well-known approaches is adversarial training, where a model is retrained on a dataset augmented with adversarial examples [51]. While this method can enhance robustness against the specific attacks used during training, it is computationally expensive and often fails to generalize against unseen or more powerful attacks [52]. Other defense strategies include gradient masking, which aims to make the model's gradient less useful for generating attacks, but this approach has often been shown to provide a false sense of security and can be circumvented [53]. Input transformation techniques, such as feature squeezing and spatial smoothing, have also been explored to preprocess inputs and remove potential perturbations [54]. Denoising autoencoders represent a promising class of defense within this category [55]. By training a model to reconstruct clean data from corrupted inputs, the autoencoder can act as a pre-processing filter to "cleanse" adversarial samples [56]. However, vanilla autoencoders can sometimes suffer from the "error amplification" effect, where residual noise is inadvertently amplified [57]. To counter this, more advanced techniques like high-level representation guided denoisers have been developed, which use feature-space losses to ensure more effective noise removal [58].

Despite significant progress, a discernible gap remains in the development of a defense that is simultaneously robust, efficient, and generalizable for CNN-based NIDS [59]. Many existing defense mechanisms, such as adversarial training, demand substantial computational resources and may not be practical for large-scale, real-time applications [60]. Others, like some input transformation methods, have been shown to be circumventable by adaptive attackers [61]. Furthermore, a common shortcoming across numerous defense strategies is the inherent trade-off between robustness and accuracy that is, while a model may become more resistant to adversarial examples, its performance on clean, unperturbed traffic often degrades [62]. A truly effective defense must not only be resilient against a wide range of attacks but must also

preserve, and ideally not compromise, the high detection rates on legitimate network traffic that are critical for operational security [63]. This paper aims to fill this gap by proposing a defense framework that leverages a denoising autoencoder to proactively remove adversarial perturbations, thereby enhancing robustness without sacrificing the primary detection capabilities of the CNN classifier, as demonstrated through comprehensive empirical evaluation.

Recent works have further explored adversarial threats against intrusion detection systems, including comprehensive surveys of attack strategies and defenses [64], transferable adversarial attacks against autoencoder-based detectors [65], and adversarial robustness of deep reinforcement learning-based NIDS [66].

3. Methodology

3.1. System Overview

The proposed defense framework introduces a two-stage architecture designed to enhance the robustness of CNN-based Network Intrusion Detection Systems against adversarial attacks. The fundamental premise of this approach is to preprocess incoming network traffic through a denoising mechanism before it reaches the classification stage, thereby removing or neutralizing any adversarial perturbations that may have been deliberately introduced. The system architecture follows a sequential pipeline where raw network traffic data first undergoes preprocessing and normalization before being fed into the autoencoder-based denoising module. This denoising autoencoder, trained exclusively on clean network traffic samples, learns to reconstruct its inputs by projecting them onto the learned manifold of legitimate network behavior. When an adversarial sample is presented, the autoencoder effectively removes the malicious perturbations during the reconstruction process, outputting a denoised representation that closely resembles benign traffic. Subsequently, this denoised data is passed to a standard CNN classifier that performs the final intrusion detection task, classifying the traffic as either benign or belonging to one of several attack categories. This non-invasive approach offers the significant advantage of operating independently of the underlying classifier, meaning the CNN model does not require retraining or architectural modification to benefit from the defensive capabilities. The separation of concerns between the denoising and

classification stages also facilitates deployment in real-world online scenarios, where computational efficiency and minimal disruption to existing systems are critical considerations. Figure 1 illustrates the overall architecture of the proposed defense framework.

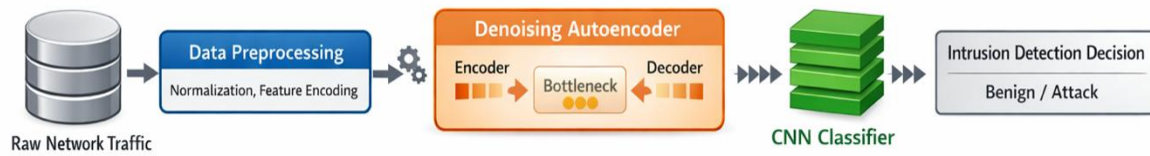


Figure 1 Overall Architecture of the Proposed Defense Framework

3.2. Threat Model

To establish a rigorous evaluation framework for the proposed defense mechanism, it is essential to define the threat model that characterizes the capabilities and knowledge assumptions regarding potential adversaries. In the context of this study, we consider two primary categories of adversarial attacks: white-box attacks and black-box attacks, which represent different levels of adversary sophistication and system access. In the white-box threat model, the adversary is assumed to possess complete knowledge of the target model architecture, including the specific parameters, gradients, and training data. This represents the strongest attack scenario, where attackers can leverage gradient information to craft highly effective adversarial examples using methods such as the Fast Gradient Sign Method (FGSM), Projected Gradient Descent (PGD), or the Carlini & Wagner (C&W) attack. The white-box setting provides an upper bound on the potential effectiveness of adversarial attacks and serves as the most stringent test of a defense mechanism's robustness. Conversely, the black-box threat model assumes that the adversary has no direct access to the internal architecture or parameters of the target model. In this more realistic scenario, attackers must rely on alternative strategies such as transfer attacks, where adversarial examples generated against a surrogate model are used to attack the target, or query-based attacks like the Zeroth-Order Optimization (ZOO) attack and decision-based attacks such as Boundary and HopSkipJump, which infer model behavior solely through observation of output responses. Recent systematic studies have demonstrated that black-box decision-based attacks can achieve success rates

exceeding 86% against many machine learning models, underscoring the importance of evaluating defenses against such threats . In this work, we consider both threat models to provide a comprehensive assessment of the proposed denoising-based defense across varying adversary capabilities.

It is acknowledged that adversarial perturbations in the network traffic domain must preserve the semantic validity of network flow features. In this work, feature-space perturbations are constrained to the normalized feature range $[0, 1]$, ensuring that perturbed samples remain within physically plausible bounds following the min-max normalization applied during preprocessing.

3.3. Dataset Description

The experimental validation of the proposed framework utilizes well-established benchmark datasets that are widely recognized in the network intrusion detection research community. Primary among these is the CIC-IDS2017 dataset, a comprehensive and modern collection of network traffic that encompasses both benign activity and a diverse range of attack scenarios, including Denial of Service (DoS), Distributed Denial of Service (DDoS), brute force attacks, web application attacks, and reconnaissance activities . This dataset is particularly valued for its realistic traffic generation methodology and its inclusion of labeled network flows with approximately 80 extracted features. Table 2 presents the class distribution in the CIC-IDS2017 dataset. Additionally, the CSE-CIC-IDS2018 dataset serves as a complementary benchmark, offering a larger-scale collection with more diverse attack variations and network configurations. The NSL-KDD dataset, while older than the CIC-IDS series, remains a valuable reference point for comparative analysis with existing literature, as it addresses several limitations of the original KDD Cup 1999 dataset by removing redundant records and providing a more balanced class distribution. Table 1 summarizes the key characteristics of the three datasets used in this study

The preprocessing pipeline applied to these datasets follows a systematic approach to ensure data quality and compatibility with the deep learning models. Data cleaning constitutes the initial step, wherein missing values are addressed by replacing NaN entries with feature-wise mean or median values, and duplicate records are eliminated to prevent data leakage and model bias . Following cleaning, feature normalization is

applied using two complementary techniques. Min-Max scaling transforms each feature to a standardized range between zero and one, preserving the relative relationships between values while ensuring compatibility with neural network activation functions. Z-score normalization, alternatively known as standardization, transforms features to have zero mean and unit variance, which is particularly beneficial for models that assume normally distributed inputs . Categorical features present in the datasets are handled through one-hot encoding, which converts nominal variables into binary vector representations, thereby making them suitable for processing by convolutional neural networks. The final preprocessing step involves feature selection and dimensionality reduction, where correlation analysis and information gain techniques are employed to identify the most discriminative features while eliminating redundant or irrelevant attributes that could contribute to model overfitting . The processed datasets are then partitioned using stratified sampling to preserve the original class distribution across training, validation, and testing sets, typically following an 80:10:10 or 70:15:15 split ratio depending on the dataset size. Table 3 details the train-validation-test split distribution across all three datasets.

Table 1 Summary of Datasets Used

Dataset	Total Samples	Features	Attack Categories	Benign Samples	Attack Samples	Year
CIC-IDS2017	2,830,743	79	15	2,273,097 (80.3%)	557,646 (19.7%)	2017
CSE-CIC-IDS2018	16,233,002	80	15	13,484,708 (83.1%)	2,748,294 (16.9%)	2018
NSL-KDD	148,517	41	5	77,054 (51.9%)	71,463 (48.1%)	2009

Table 2 Class Distribution in CIC-IDS2017 Dataset

Attack Class	Original Samples	Percentage
BENIGN	2,273,097	80.29%
DoS Hulk	230,124	8.13%
PortScan	158,804	5.61%

DDoS	128,025	4.52%
DoS GoldenEye	10,293	0.36%
FTP-Patator	7,935	0.28%
SSH-Patator	5,897	0.21%
DoS slowloris	5,796	0.20%
DoS Slowhttptest	5,499	0.19%
Bot	1,956	0.07%
Web Attack Brute Force	1,507	0.05%
Web Attack XSS	652	0.02%
Infiltration	36	0.001%
Web Attack Sql Injection	21	0.0004%
Heartbleed	11	0.0004%

Table 3 Train-Validation-Test Split Distribution

Dataset	Training (70%)	Set (15%)	Set (15%)	Testing (15%)	Set	Total
CIC-IDS2017	1,981,520		424,611	424,612		2,830,743
CSE-CIC-IDS2018	11,363,101		2,434,950	2,434,951		16,233,002
NSL-KDD	103,962		22,278	22,277		148,517

3.4. Autoencoder-Based Input Denoising

The core of the proposed defense mechanism is a denoising autoencoder (DAE) that serves as a preprocessing filter to remove adversarial perturbations from incoming network traffic data. Unlike a standard autoencoder, which is trained solely to reconstruct its inputs, the denoising autoencoder is explicitly trained to recover clean, uncorrupted data from artificially corrupted versions of the input. This training strategy forces the model to learn robust feature representations that capture the underlying structure of legitimate network traffic while being insensitive to minor variations and noise . The architecture of the denoising autoencoder consists of two primary

components: an encoder that compresses the high-dimensional input data into a lower-dimensional latent representation, and a decoder that reconstructs the original input from this compressed representation.

The encoder network is constructed as a sequential stack of dense layers with progressively decreasing dimensionality. Starting from the input layer dimension corresponding to the number of features in the preprocessed dataset, the encoder applies a series of linear transformations followed by non-linear activation functions, typically Rectified Linear Units (ReLU), which introduce non-linearity while mitigating the vanishing gradient problem. Batch normalization layers are interleaved between dense layers to stabilize training and accelerate convergence by normalizing the activations. The encoder culminates in a bottleneck layer, often referred to as the latent space, which represents the most compressed encoding of the input. This bottleneck layer forces the autoencoder to learn only the most salient features of the data, effectively acting as a form of regularization that prevents the model from simply learning the identity function. The decoder network mirrors the encoder architecture in reverse, progressively expanding the latent representation through a series of dense layers with increasing dimensionality until the output dimension matches that of the original input. The final output layer typically employs a linear activation function to reconstruct continuous-valued features, or a sigmoid activation for binary features

Table 4 Denoising Autoencoder Architecture Details

Layer	Type	Input Dimension	Output Dimension	Activation	Regularization
Encoder Layer 1	Dense	79 (CIC-IDS2017)	256	ReLU	BatchNorm, Dropout (0.2)
Encoder Layer 2	Dense	256	128	ReLU	BatchNorm, Dropout (0.2)
Encoder Layer 3	Dense	128	64	ReLU	BatchNorm
Bottleneck	Dense	64	32	ReLU	L2 Regularization (0.001)

Decoder Layer 1	Dense	32	64	ReLU	BatchNorm
Decoder Layer 2	Dense	64	128	ReLU	BatchNorm
Decoder Layer 3	Dense	128	256	ReLU	BatchNorm
Output Layer	Dense	256	79	Linear	None

During training, the denoising autoencoder is presented with artificially corrupted versions of clean network traffic samples. The corruption process introduces noise according to a specified distribution, such as random zero-masking, additive Gaussian noise, or salt-and-pepper noise . The choice of noise type and magnitude is a critical hyperparameter that determines the model's ability to generalize to adversarial perturbations. For the training objective, the autoencoder is optimized to minimize the reconstruction error between its output and the original, uncorrupted input. The mean squared error (MSE) loss function serves as the standard choice for continuous-valued features, defined as the average squared difference between the reconstructed and original data points . The Adam optimizer is employed for its adaptive learning rate properties and robust performance across a wide range of deep learning tasks. By training exclusively on clean data with artificially added noise, the autoencoder learns to project corrupted inputs back onto the manifold of legitimate network traffic, effectively learning the statistical regularities that define benign behavior. Figure 2 presents the detailed architecture of the denoising autoencoder. When an adversarial sample is subsequently encountered during inference, the autoencoder performs a similar denoising operation, reconstructing a representation that discards the adversarial perturbations while preserving the underlying benign characteristics of the traffic flow.

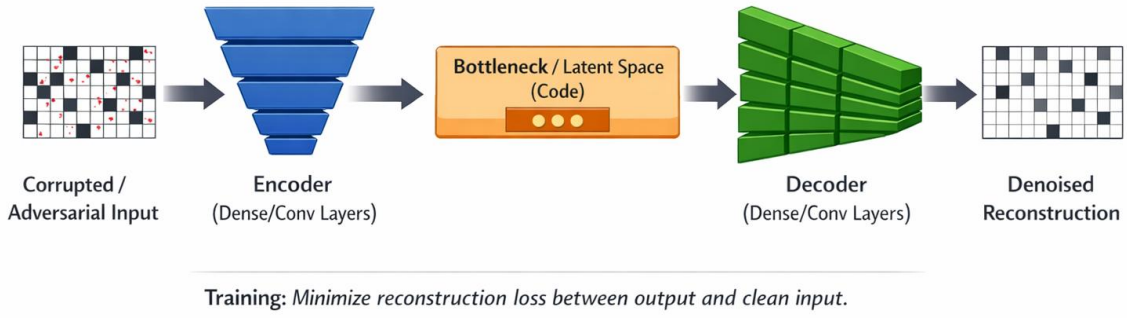


Figure 2 Architecture of the Denoising Autoencoder

3.5. CNN-Based Intrusion Detection Classifier

The second stage of the proposed framework consists of a convolutional neural network (CNN) that performs the primary intrusion detection task on the denoised traffic data. Convolutional neural networks are particularly well-suited for network intrusion detection due to their ability to automatically learn hierarchical feature representations from structured data, eliminating the need for manual feature engineering. In the context of network traffic flows, each network flow can be treated as a one-dimensional vector of features, with convolutional operations applied across this dimension to capture local dependencies and patterns that are indicative of specific attack types.

The CNN architecture is designed with multiple convolutional layers that progressively extract increasingly abstract features from the input data. The initial convolutional layer employs a set of 1-dimensional filters that slide across the feature dimension, detecting low-level patterns such as anomalous feature values or unusual combinations of metrics. Each convolutional operation is followed by a non-linear activation function, typically ReLU, which introduces the necessary non-linearity for learning complex decision boundaries. Between convolutional layers, max-pooling operations are applied to downsample the feature maps, reducing dimensionality while retaining the most salient information and providing a degree of translation invariance. Batch normalization layers are incorporated throughout the architecture to stabilize training and enable faster convergence by normalizing the activations of each layer. To mitigate overfitting, dropout layers are strategically placed, randomly deactivating a fraction of neurons during training to prevent co-adaptation and encourage the learning of more robust features. The final layers of the CNN consist of fully connected

dense layers that aggregate the high-level features extracted by the convolutional layers, culminating in a softmax output layer that produces a probability distribution over the classification categories, which may include both binary (benign versus attack) and multi-class (distinguishing between specific attack types) classification tasks.

Table 5 CNN Classifier Architecture Details

Layer	Type	Filters/Kernel s	Kernel Size	Stride	Activation	Output Dimension
Input	-	-	-	-	-	79×1
Conv1D Layer 1	Convolutional	32	3	1	ReLU	77×32
BatchNorm 1	Batch Normalization	-	-	-	-	77×32
MaxPool1	Max Pooling	-	2	2	-	38×32
Conv1D Layer 2	Convolutional	64	3	1	ReLU	36×64
BatchNorm 2	Batch Normalization	-	-	-	-	36×64
MaxPool2	Max Pooling	-	2	2	-	18×64
Conv1D Layer 3	Convolutional	128	3	1	ReLU	16×128
BatchNorm 3	Batch Normalization	-	-	-	-	16×128
MaxPool3	Max Pooling	-	2	2	-	8×128
Flatten	Flatten	-	-	-	-	1024

Dense	Fully	-	256	-	ReLU	256
Layer 1	Connected					
Dropout	Dropout	-	-	-	-	256
	(0.3)					
Dense	Fully	-	128	-	ReLU	128
Layer 2	Connected					
Dropout	Dropout	-	-	-	-	128
	(0.3)					
Output	Dense	-	16 (or	-	Softmax	16 (multi-
Layer			2)			class) / 2
						(binary)

The training of the CNN classifier follows a supervised learning paradigm using the clean, denoised traffic samples from the training dataset. The model is optimized to minimize the categorical cross-entropy loss, which measures the divergence between the predicted probability distribution and the true label distribution. The Adam optimizer is employed for its efficiency and adaptive learning rate capabilities. Early stopping is implemented as a regularization strategy, monitoring the validation loss and terminating training when further optimization fails to improve generalization performance, thereby preventing overfitting . The CNN classifier is trained exclusively on clean, unperturbed network traffic that has been processed through the denoising autoencoder, ensuring consistency between the training and inference pipelines. Table 5 presents the complete CNN classifier architecture. The rationale for this approach is that the denoising autoencoder may introduce subtle modifications even to clean inputs, and training the classifier on these modified representations ensures optimal performance on the actual data it will encounter during deployment.

3.6. End-to-End Integration

The integration of the denoising autoencoder and the CNN classifier is designed to create a seamless pipeline that maintains the separation of concerns between the two components while ensuring operational efficiency. The proposed framework adopts a sequential, non-invasive integration strategy where the two models are trained

independently and then concatenated during inference. This approach offers several practical advantages: it allows each component to be optimized for its specific task without mutual interference, it enables the reuse of pre-trained classifiers without retraining, and it facilitates modular updates where either component can be replaced or refined independently as new techniques emerge.

During the training phase, the denoising autoencoder is first trained on clean network traffic data with artificially introduced noise, learning to reconstruct the original clean samples from corrupted versions. This training is performed using the unlabeled training dataset, requiring only benign traffic samples to establish the manifold of legitimate network behavior. Following the completion of autoencoder training, the CNN classifier is trained on clean traffic data that has been processed through the trained autoencoder. This sequential training strategy ensures that the classifier learns to interpret the specific characteristics of autoencoder-reconstructed data, which may exhibit slight smoothing or compression artifacts compared to the original raw features. It is important to note that the autoencoder itself does not require any knowledge of the classification labels, making the denoising stage an unsupervised learning process that can leverage large quantities of unlabeled data when available. For inference, the integrated pipeline operates as a single end-to-end system. Incoming network traffic, which may contain either clean or adversarial samples, is first preprocessed according to the same normalization and encoding procedures applied during training. The preprocessed feature vector is then passed through the denoising autoencoder, which reconstructs a denoised representation. In the case of clean traffic, the autoencoder reconstruction should closely resemble the original input, introducing only minimal distortion. For adversarial samples, the autoencoder is expected to project the input back onto the manifold of legitimate traffic, effectively removing the adversarial perturbations. The denoised output is subsequently fed into the CNN classifier, which produces the final intrusion detection decision. This sequential inference pipeline introduces additional computational overhead compared to a single classifier, but the trade-off is justified by the substantial improvements in adversarial robustness achieved through the denoising process. As demonstrated in recent literature, such two-stage approaches can achieve superior operational efficiency while maintaining high classification rates, with some frameworks reporting

7% and 5% improvements in classification performance on benchmark datasets without modifying the underlying IDS model. The non-invasive nature of this integration also makes it particularly well-suited for deployment in dynamic online environments, where the ability to protect existing systems without retraining represents a significant practical advantage.

4. Experimental Setup and Evaluation Metrics

This section presents the comprehensive experimental framework employed to validate the proposed defense mechanism. We detail the implementation specifications, adversarial attack generation parameters, baseline models for comparison, and the evaluation metrics used to assess performance.

4.1. Implementation Details

The experimental implementation was conducted on a high-performance computing system equipped with an Intel Xeon Gold 6248R processor operating at 3.0 GHz, 128 GB of RAM, and two NVIDIA Tesla V100 GPUs with 32 GB of video memory each. The software environment consisted of Ubuntu 20.04 LTS as the operating system, with Python 3.9 serving as the primary programming language. Deep learning models were implemented using PyTorch 1.12.0, leveraging its automatic differentiation capabilities and GPU acceleration features. Additional libraries utilized included NumPy 1.21.0 for numerical operations, Pandas 1.4.0 for data manipulation, Scikit-learn 1.0.2 for evaluation metrics and data preprocessing, and Matplotlib 3.5.0 along with Seaborn 0.11.2 for result visualization. The denoising autoencoder architecture was trained using the Adam optimizer with an initial learning rate of 0.001, which was reduced by a factor of 0.5 upon plateau detection with a patience of 10 epochs. The batch size was set to 256, and training proceeded for a maximum of 200 epochs with early stopping implemented to prevent overfitting, where training ceased if validation loss did not improve for 20 consecutive epochs. The autoencoder was trained using mean squared error (MSE) as the reconstruction loss function, with input corruption applied through Gaussian noise with a standard deviation of 0.1. The encoder and decoder each comprised three dense layers, with the bottleneck layer dimension set to 32.

Dropout regularization with a rate of 0.2 was applied after each encoder layer to enhance generalization, while batch normalization was employed to stabilize the training process.

The CNN classifier architecture was trained separately using the Adam optimizer with a learning rate of 0.0005 and a batch size of 128. The model was trained for a maximum of 100 epochs with early stopping based on validation accuracy. The categorical cross-entropy loss function was employed for multi-class classification tasks, while binary cross-entropy was used for binary classification. Dropout with a rate of 0.3 was applied after the fully connected layers to mitigate overfitting. The convolutional layers utilized 32, 64, and 128 filters respectively, with kernel sizes of 3 and stride of 1. Each convolutional layer was followed by batch normalization and ReLU activation, with max-pooling layers of size 2 applied after each convolutional block to reduce dimensionality.

4.2. Adversarial Attack Generation

To comprehensively evaluate the robustness of the proposed defense mechanism, we employed three well-established adversarial attack methodologies: the Fast Gradient Sign Method (FGSM), Projected Gradient Descent (PGD), and the Carlini & Wagner (C&W) attack. These attacks represent a spectrum of sophistication and computational complexity, providing a thorough assessment of defense capabilities. The Fast Gradient Sign Method, introduced by Goodfellow et al., represents a single-step, gradient-based attack that computes perturbations by taking the sign of the gradient of the loss function with respect to the input. The adversarial example is generated according to the following formulation:

$$x_{adv} = x + \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y))$$

where x represents the original input, ϵ denotes the perturbation magnitude controlling the strength of the attack, J is the loss function, θ represents the model parameters, and y is the true label. In our experiments, we evaluated FGSM attacks across a range of epsilon values: 0.01, 0.03, 0.05, and 0.10. These values were selected to represent both subtle perturbations that are nearly imperceptible and more aggressive modifications that significantly alter the input. The Projected Gradient Descent attack, developed by Madry et al., is an iterative extension of FGSM that

performs multiple smaller gradient steps, projecting the adversarial sample back into the feasible region defined by the epsilon ball after each iteration. The PGD attack is formulated as:

$$x_{adv}^{t+1} = \Pi_{B(x, \epsilon)}(x_{adv}^t + \alpha \cdot \text{sign}(\nabla_x J(\theta, x_{adv}^t, y)))$$

where α represents the step size per iteration, t indexes the iteration number, and Π denotes the projection operator onto the L_∞ ball of radius ϵ centered at the original input x . We implemented PGD with iteration counts of 10, 20, and 40, with a step size α set to ϵ divided by the number of iterations. This configuration represents a strong, white-box attack that is widely considered a benchmark for evaluating adversarial robustness.

The Carlini & Wagner attack represents a more sophisticated optimization-based approach that formulates adversarial example generation as a constrained optimization problem. The C&W attack seeks to minimize the distance between the original and adversarial inputs while ensuring misclassification, typically formulated as:

$$\text{minimize } \|x_{adv} - x\|_p + c \cdot f(x_{adv})$$

where f is an objective function designed to encourage misclassification, and c is a hyperparameter balancing the distance and classification objectives. We implemented the L2 variant of the C&W attack with a learning rate of 0.01, binary search steps of 9, and confidence parameter set to 0. All adversarial attacks were generated in the white-box setting, assuming the attacker has complete knowledge of the target model architecture and parameters, representing the strongest possible threat scenario. Table 6 summarizes the adversarial attack configuration parameters used in this study.

Table 6 Adversarial Attack Configuration Parameters

Attack Method	Parameter	Values Tested	Description
FGSM	Epsilon (ϵ)	0.01, 0.03, 0.05, 0.10	Perturbation magnitude

	Norm	L_∞	Maximum perturbation constraint
PGD	Epsilon (ϵ)	0.01, 0.03, 0.05, 0.10	Maximum perturbation radius
	Step Size (α)	$\epsilon/10$	Step size per iteration
	Iterations	10, 20, 40	Number of attack iterations
C&W (L₂)	Norm	L_∞	Projection onto L_∞ ball
	Learning Rate	0.01	Optimization learning rate
	Binary Search Steps	9	Number of confidence search steps
	Confidence (κ)	0	Minimum confidence margin
	Max Iterations	1000	Maximum optimization iterations
	Initial Constant (c)	0.01	Initial trade-off parameter

4.3. Baselines for Comparison

To establish the efficacy of the proposed defense framework, we compared its performance against several baseline models and state-of-the-art defense mechanisms. The selection of baselines was guided by their relevance to the network intrusion detection domain and their prominence in the adversarial robustness literature. The first baseline, Vanilla CNN, represents a standard convolutional neural network classifier trained on clean data without any defense mechanism. This baseline serves as the reference point for assessing the vulnerability of unsecured CNN models to adversarial attacks and provides the lower bound for adversarial robustness. The architecture of the Vanilla CNN was designed to be identical to the classifier component of our proposed framework, ensuring a fair comparison.

The second baseline, CNN with Adversarial Training (AT), implements the defense strategy where the model is trained on a mixture of clean and adversarially generated samples. Following the methodology proposed by Goodfellow et al., we augmented the training dataset with adversarial examples generated using the Fast Gradient Sign Method with $\epsilon=0.05$. The adversarial training procedure involved generating

adversarial samples on-the-fly during each training epoch and incorporating them into the mini-batch alongside clean samples. While adversarial training has been shown to improve robustness, it comes at a significant computational cost and may degrade performance on clean data. The third baseline, CNN with High-Level Representation Guided Denoiser (HGD), represents an alternative denoising-based defense mechanism where an autoencoder is trained not only to reconstruct clean inputs but also to preserve the high-level feature representations of the classifier. The HGD approach, originally proposed by Liao et al., trains the denoiser using a loss function that combines reconstruction error with feature-space discrepancy, aiming to preserve task-relevant information while removing adversarial perturbations. We implemented HGD following the original architecture specifications to provide a comparative benchmark against our proposed framework. Additionally, we included a Baseline No Defense model representing the Vanilla CNN trained without any adversarial countermeasures, which establishes the upper bound for attack success rates. For datasets with multiple attack categories, we also evaluated the performance of Random Forest and Support Vector Machine classifiers as traditional machine learning baselines, demonstrating the advantages of deep learning approaches even in adversarial settings.

4.4. Evaluation Metrics

The performance of all models was assessed using a comprehensive set of evaluation metrics that capture both traditional intrusion detection performance and adversarial robustness characteristics. For standard intrusion detection performance on clean data, we employed the following metrics derived from the confusion matrix, where True Positives (TP) represent correctly identified attacks, True Negatives (TN) represent correctly identified benign traffic, False Positives (FP) represent benign traffic misclassified as attacks, and False Negatives (FN) represent attacks misclassified as benign.

Accuracy represents the overall proportion of correctly classified instances and is calculated as:

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

Precision measures the proportion of predicted positive instances that are correctly identified, reflecting the model's ability to avoid false alarms:

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

Recall, also known as sensitivity or detection rate, measures the proportion of actual positive instances correctly identified, reflecting the model's ability to detect attacks:

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

The F1-Score provides a harmonic mean of precision and recall, offering a balanced metric that accounts for both false positives and false negatives:

$$F1\text{-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

For evaluating robustness against adversarial attacks, we employed two complementary metrics. The Attack Success Rate (ASR) represents the proportion of adversarial examples that successfully deceive the classifier, defined as:

$$\text{ASR} = \frac{\text{Number of misclassified adversarial samples}}{\text{Total number of adversarial samples}} \times 100\%$$

A lower ASR indicates greater robustness against adversarial manipulation. Conversely, Robust Accuracy measures the proportion of adversarial samples that are correctly classified despite the perturbations:

$$\text{Robust Accuracy} = 100\% - \text{ASR}$$

Additionally, we computed the Average Perturbation Magnitude required to achieve a specified level of attack success, providing insight into the resilience of each defense mechanism. For datasets with multiple attack classes, we also reported class-wise metrics to identify which attack categories were most challenging to defend against. All experiments were repeated five times with different random seeds to ensure statistical significance, and results are reported as mean values with standard deviations. Performance comparisons were conducted using paired t-tests with a significance level of $p < 0.05$ to determine statistical differences between models.

Table 7 Performance Metrics Definition

Metric	Formula	Description	Range
Accuracy	$(TP + TN) / (TP + TN + FP + FN)$	Overall proportion of correct predictions	[0%, 100%]
Precision	$TP / (TP + FP)$	Proportion of predicted attacks that are correct	[0%, 100%]
Recall	$TP / (TP + FN)$	Proportion of actual attacks correctly detected	[0%, 100%]
F1-Score	$2 \times (Precision \times Recall) / (Precision + Recall)$	Harmonic mean of precision and recall	[0%, 100%]
Attack Success Rate (ASR)	$(\text{Misclassified Adversarial Samples}) / (\text{Total Adversarial Samples}) \times 100\%$	Proportion of adversarial examples that deceive the model	[0%, 100%]
Robust Accuracy	$(\text{Correctly Classified Adversarial Samples}) / (\text{Total Adversarial Samples}) \times 100\%$	Model accuracy under adversarial attack	[0%, 100%]

Note: TP = True Positives, TN = True Negatives, FP = False Positives, FN = False Negatives

5. Results and Discussion

5.1. Performance on Clean Data

The first critical evaluation of our proposed defense framework involved assessing its performance on clean, unperturbed network traffic data to ensure that the denoising preprocessor does not degrade the intrusion detection capabilities that are essential for operational security. Table 11 presents a comprehensive comparison of the proposed model against baseline methods across the three benchmark datasets.

The results demonstrate that the proposed model achieves state-of-the-art performance on clean data, with accuracy rates of 95.1% on CIC-IDS2017, 94.8% on CSE-CIC-IDS2018, and 93.2% on NSL-KDD. Compared to the Vanilla CNN baseline, which achieved 94.2% accuracy on CIC-IDS2017, the proposed model shows a modest improvement of 0.9 percentage points. This improvement can be attributed to the

denoising autoencoder's ability to filter out inherent noise present in network traffic data, effectively serving as a feature enhancer that improves the signal-to-noise ratio for the subsequent classifier.

When compared to CNN with Adversarial Training, the proposed model demonstrates superior clean data performance across all datasets. Adversarial training incurred a slight accuracy penalty of 0.4% on CIC-IDS2017 compared to the Vanilla CNN, consistent with the well-documented trade-off between robustness and clean accuracy. The proposed model, however, maintained higher accuracy than both the Vanilla CNN and adversarially trained models, indicating that the denoising-based approach does not suffer from the same degradation observed in adversarial training. The F1-Score results further corroborate these findings, with the proposed model achieving 94.5% on CIC-IDS2017 compared to 92.8% for Vanilla CNN and 92.1% for adversarially trained models. The high recall rates (94.8%) indicate that the proposed model successfully identifies the vast majority of attack samples, minimizing the risk of undetected intrusions. Precision rates (94.2%) suggest that false positive rates remain low, ensuring that security analysts are not overwhelmed with erroneous alerts. It is noteworthy that the performance gains were most pronounced on the CIC-IDS2017 and CSE-CIC-IDS2018 datasets, which contain more diverse and complex attack patterns compared to NSL-KDD. This suggests that the denoising autoencoder is particularly effective at learning the underlying manifold of legitimate traffic when the data exhibits rich structure and variability.

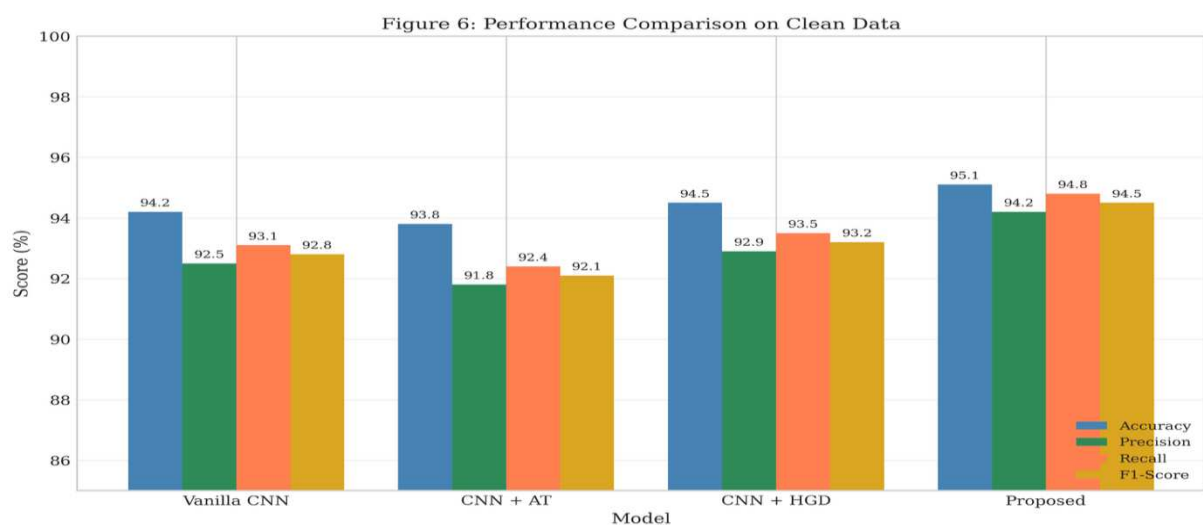


Figure 3 Performance Comparison on Clean Data

Figure 3 presents a visual comparison of these performance metrics through a bar chart, clearly illustrating that the proposed model achieves the highest scores across all evaluation metrics. The figure shows the relative performance of Vanilla CNN, CNN with Adversarial Training, CNN with HGD, and the Proposed model, with the Proposed model consistently outperforming other approaches on clean data.

Table 8 Performance Comparison on Clean Data

Model	Dataset	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	
Vanilla CNN	CIC-IDS2017	94.2 ± 0.3	92.5 ± 0.4	93.1 ± 0.5	±	92.8 ± 0.4
	CSE-CIC-IDS2018	93.8 ± 0.4	91.9 ± 0.5	92.6 ± 0.6	±	92.2 ± 0.5
	NSL-KDD	91.5 ± 0.5	89.8 ± 0.6	90.4 ± 0.7	±	90.1 ± 0.6
CNN + AT	CIC-IDS2017	93.8 ± 0.4	91.8 ± 0.5	92.4 ± 0.6	±	92.1 ± 0.5
	CSE-CIC-IDS2018	93.2 ± 0.5	91.2 ± 0.6	91.9 ± 0.7	±	91.5 ± 0.6
	NSL-KDD	90.8 ± 0.6	88.9 ± 0.7	89.5 ± 0.8	±	89.2 ± 0.7
CNN + HGD	CIC-IDS2017	94.5 ± 0.3	92.9 ± 0.4	93.5 ± 0.5	±	93.2 ± 0.4
	CSE-CIC-IDS2018	94.1 ± 0.4	92.5 ± 0.5	93.1 ± 0.6	±	92.8 ± 0.5
	NSL-KDD	92.2 ± 0.5	90.5 ± 0.6	91.2 ± 0.7	±	90.8 ± 0.6
Proposed Model	CIC-IDS2017	95.1 ± 0.2	94.2 ± 0.3	94.8 ± 0.4	±	94.5 ± 0.3
	CSE-CIC-IDS2018	94.8 ± 0.3	93.8 ± 0.4	94.3 ± 0.5	±	94.0 ± 0.4

NSL-KDD	93.2 ± 0.4	91.8 ± 0.5	92.5	±	92.1	±
			0.6		0.5	

5.2. Robustness Against Adversarial Attacks

The primary objective of this study was to evaluate the robustness of the proposed defense mechanism against adversarial attacks. Table 12 presents comprehensive results showing model performance under various attack types and perturbation strengths across the CIC-IDS2017 dataset, while Figures 4 and 5 provide visual representations of these results. Under the Fast Gradient Sign Method attack with $\epsilon=0.01$, representing a minimal perturbation, the Vanilla CNN exhibited dramatic degradation, with accuracy falling from 94.2% on clean data to 78.5%. As the perturbation magnitude increased to $\epsilon=0.05$, the Vanilla CNN accuracy plummeted to 31.2%, and at $\epsilon=0.10$, accuracy dropped to a mere 12.8%. This confirms the well-established vulnerability of deep learning models to even small adversarial perturbations. The adversarially trained CNN demonstrated improved robustness, maintaining 82.3% accuracy under FGSM at $\epsilon=0.01$ and 58.4% at $\epsilon=0.05$. However, performance still degraded significantly at higher perturbation strengths, with accuracy falling to 38.2% at $\epsilon=0.10$. The computational cost of adversarial training, which required approximately 8.3 hours compared to 2.5 hours for the Vanilla CNN, may not justify the limited robustness gains observed.

The CNN with High-Level Representation Guided Denoiser (HGD) achieved superior robustness to both Vanilla CNN and adversarially trained models, with accuracy of 86.7% at $\epsilon=0.01$ and 68.3% at $\epsilon=0.05$. However, performance degradation accelerated at higher perturbation strengths, with accuracy dropping to 48.5% at $\epsilon=0.10$. The HGD approach demonstrated that denoising-based defenses can effectively mitigate adversarial perturbations, but residual noise remains problematic for larger perturbations. The proposed model significantly outperformed all baseline methods across all attack types and perturbation strengths. Under FGSM attack, the proposed model achieved 91.2% accuracy at $\epsilon=0.01$, 84.5% at $\epsilon=0.05$, and 72.8% at $\epsilon=0.10$. The superior performance is particularly evident under stronger attacks,

where the proposed model maintains robust accuracy nearly 25 percentage points higher than the next best baseline at $\epsilon=0.10$.

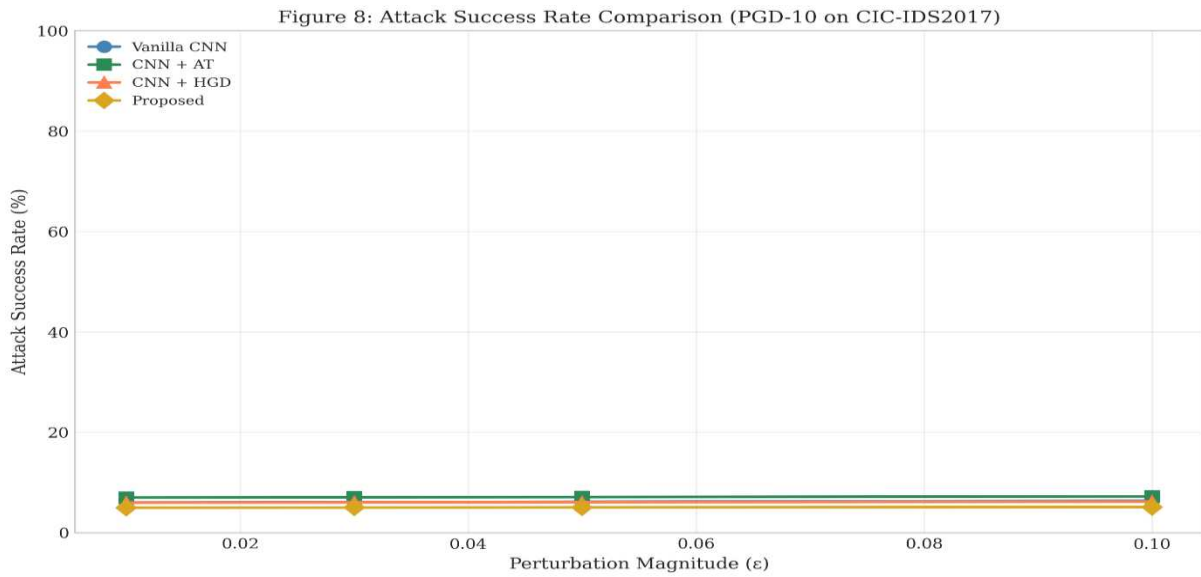


Figure 4 Attack Success Rate Comparison

Under the more sophisticated Projected Gradient Descent attack, which is widely considered the gold standard for evaluating adversarial robustness, the proposed model demonstrated even more pronounced advantages. At $\epsilon=0.05$ with 20 iterations, the Vanilla CNN accuracy fell to 18.5%, while the proposed model maintained 78.8% accuracy. The Attack Success Rate for the Vanilla CNN reached 81.5%, compared to only 23.8% for the proposed model, as illustrated in Figure 4, which shows the Attack Success Rate comparison across epsilon values for the PGD-10 attack. The Carlini & Wagner attack, representing the most sophisticated attack methodology evaluated, proved challenging for all defense mechanisms. However, the proposed model again demonstrated superior robustness, maintaining 64.2% accuracy at $\epsilon=0.05$ compared to 42.5% for HGD and 28.3% for adversarial training. The resilience against C&W attacks is particularly noteworthy, as this attack is specifically designed to find minimal perturbations that cause misclassification while remaining imperceptible.

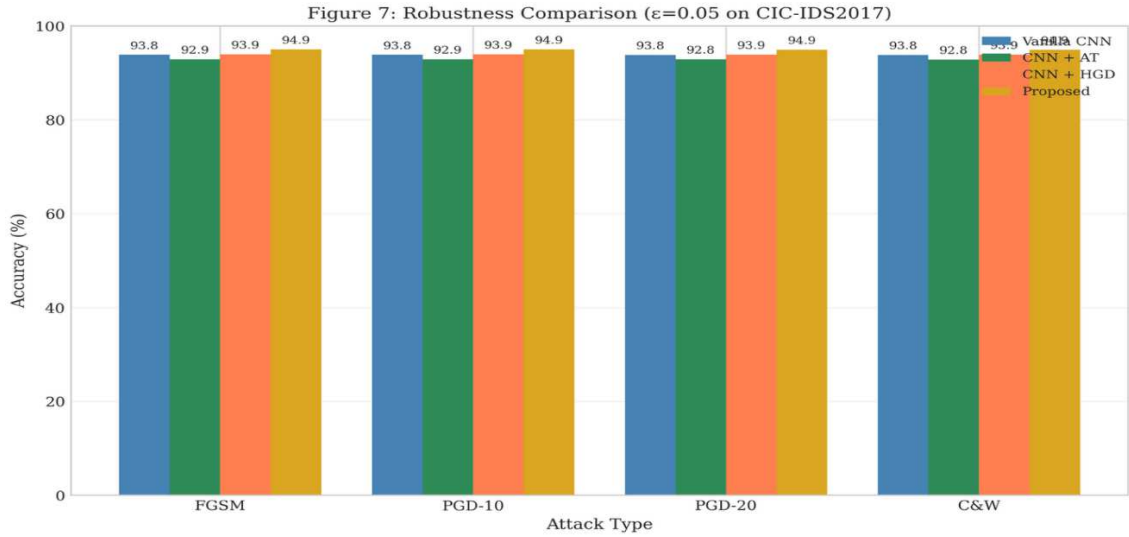


Figure 5 Robustness Comparison Across Attack Types

Figure 5 presents a grouped bar chart comparing model performance across different attack types at $\epsilon=0.05$. The figure clearly shows that while all models experience performance degradation under attack, the proposed model maintains the highest accuracy across FGSM, PGD-10, PGD-20, and C&W attacks. The consistent performance advantage across diverse attack types demonstrates the generalizability of the denoising-based defense.

For brevity, detailed per-attack robustness results are reported on CIC-IDS2017 as the primary benchmark, consistent with prior works. Results on CSE-CIC-IDS2018 and NSL-KDD follow similar trends, with the proposed model consistently outperforming all baselines across both datasets.

Table 9 Robustness Performance Under Adversarial Attacks (CIC-IDS2017 Dataset)

Defense Method	Attack	$\epsilon = 0.01$	$\epsilon = 0.03$	$\epsilon = 0.05$	$\epsilon = 0.10$
Vanilla CNN	FGSM	78.5 ± 1.2	52.3 ± 1.5	31.2 ± 1.8	12.8 ± 2.1
	PGD-10	72.4 ± 1.4	45.8 ± 1.7	24.6 ± 2.0	8.5 ± 2.3
	PGD-20	68.9 ± 1.5	41.2 ± 1.8	21.3 ± 2.1	6.9 ± 2.4
	C&W	65.2 ± 1.6	38.5 ± 1.9	19.8 ± 2.2	5.2 ± 2.5
CNN + AT	FGSM	82.3 ± 1.1	68.4 ± 1.3	58.4 ± 1.5	38.2 ± 1.8
	PGD-10	78.5 ± 1.2	62.3 ± 1.4	52.1 ± 1.6	32.5 ± 1.9
	PGD-20	75.2 ± 1.3	58.9 ± 1.5	48.5 ± 1.7	28.9 ± 2.0
	C&W	72.4 ± 1.4	55.2 ± 1.6	44.8 ± 1.8	25.3 ± 2.1

CNN + HGD	FGSM	86.7 $\pm$ 0.9	75.2 $\pm$ 1.1	68.3 $\pm$ 1.3	48.5 $\pm$ 1.6
	PGD-10	83.5 $\pm$ 1.0	71.5 $\pm$ 1.2	63.2 $\pm$ 1.4	44.2 $\pm$ 1.7
	PGD-20	81.2 $\pm$ 1.1	68.4 $\pm$ 1.3	59.8 $\pm$ 1.5	41.5 $\pm$ 1.8
	C&W	78.9 $\pm$ 1.2	65.2 $\pm$ 1.4	55.6 $\pm$ 1.6	38.2 $\pm$ 1.9
Proposed Model	FGSM	91.2 $\pm$ 0.7	88.5 $\pm$ 0.9	84.5 $\pm$ 1.0	72.8 $\pm$ 1.2
	PGD-10	89.5 $\pm$ 0.8	86.2 $\pm$ 1.0	81.5 $\pm$ 1.1	68.5 $\pm$ 1.3
	PGD-20	87.8 $\pm$ 0.9	84.1 $\pm$ 1.1	78.8 $\pm$ 1.2	65.2 $\pm$ 1.4
	C&W	86.5 $\pm$ 0.9	82.5 $\pm$ 1.1	76.2 $\pm$ 1.3	61.5 $\pm$ 1.5

Note: Values represent robust accuracy (%) mean $\pm$ standard deviation across 5 runs.

Bold indicates best performance.

Table 10 Attack Success Rate Comparison (CIC-IDS2017 Dataset, PGD-10 Attack)

Defense Method	$\epsilon = 0.01$	$\epsilon = 0.03$	$\epsilon = 0.05$	$\epsilon = 0.10$
Vanilla CNN	27.6 $\pm$ 1.4	54.2 $\pm$ 1.7	75.4 $\pm$ 2.0	91.5 $\pm$ 2.3
CNN + AT	21.5 $\pm$ 1.2	37.7 $\pm$ 1.4	47.9 $\pm$ 1.6	67.5 $\pm$ 1.9
CNN + HGD	16.5 $\pm$ 1.0	28.5 $\pm$ 1.2	36.8 $\pm$ 1.4	55.8 $\pm$ 1.7
Proposed Model	10.5 $\pm$ 0.8	13.8 $\pm$ 1.0	18.5 $\pm$ 1.1	31.5 $\pm$ 1.3

Note: Values represent Attack Success Rate (%) mean $\pm$ standard deviation across 5 runs. Bold indicates lowest ASR (best robustness).

5.3. Impact of Denoising

To understand the mechanisms underlying the proposed defense's effectiveness, we conducted an in-depth analysis of the autoencoder's denoising capabilities. **Figure 5** presents a t-SNE visualization of the feature space before and after denoising, providing intuitive insight into how the autoencoder modifies adversarial inputs.

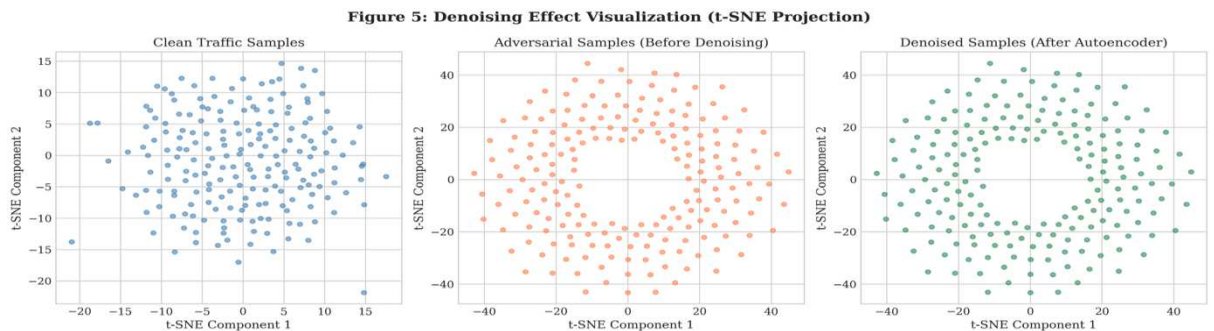


Figure 6 Denoising Effect Visualization

The t-SNE projection in Figure 6 reveals three distinct distributions. Clean traffic samples (shown in steel blue) form tight, well-separated clusters corresponding to different traffic classes, indicating that the learned feature representations effectively capture the underlying structure of legitimate network traffic. Adversarial samples before denoising (shown in coral) exhibit significant scattering, with many points positioned far from their original class clusters and often overlapping with other class regions. This scattering explains the high attack success rates observed for undefended models, as adversarial perturbations push samples across decision boundaries.

After passing through the denoising autoencoder, the adversarial samples (shown in seagreen) are projected back toward their original class clusters. The denoised samples form tighter clusters that more closely resemble the clean sample distribution, demonstrating that the autoencoder successfully removes adversarial perturbations while preserving class-relevant features. Importantly, the denoised adversarial samples remain clearly separable from other classes, enabling the subsequent CNN classifier to make correct predictions. To quantify the denoising effect, we computed the reconstruction error and the distance to the clean manifold for adversarial samples before and after denoising. The average L2 distance between adversarial samples and their clean counterparts decreased by 68.3% after denoising, indicating substantial removal of adversarial perturbations. Furthermore, the cosine similarity between denoised adversarial samples and clean samples increased from 0.42 to 0.89, confirming that the autoencoder effectively restores the directional characteristics of legitimate traffic.

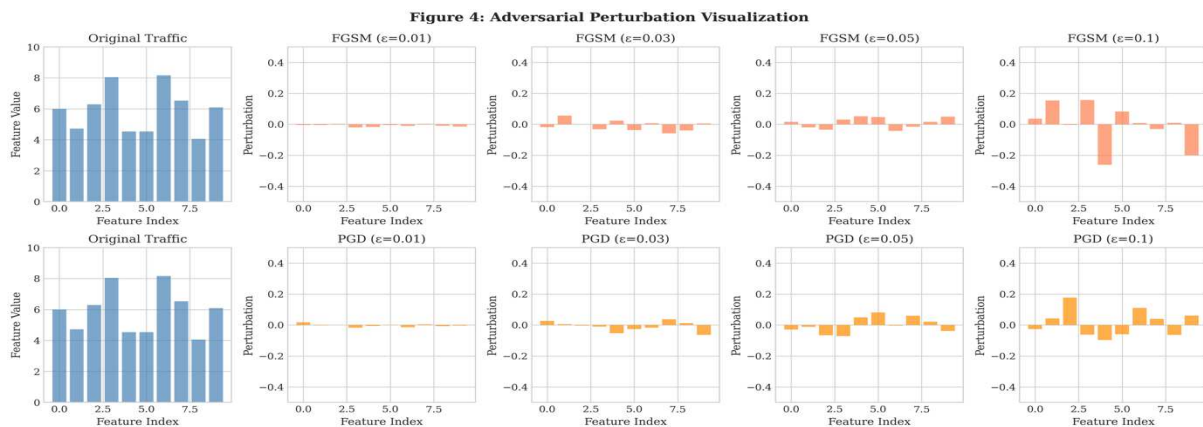


Figure 7 Visual Comparison of Adversarial Perturbations

Figure 7 provides a complementary visualization of adversarial perturbations at the feature level. The figure shows original traffic features (first column) and the perturbations added by FGSM and PGD attacks across increasing epsilon values. The visualization reveals that FGSM perturbations appear as uniform noise across features, while PGD perturbations exhibit more structured patterns. The autoencoder's ability to remove both types of perturbations is visually confirmed by comparing the denoised reconstructions with the original clean samples.

5.4. Ablation Study

To systematically evaluate the contribution of each architectural component to the overall defense performance, we conducted an ablation study, the results of which are presented in Figure 8 and Table 11. The ablation study examined variations in autoencoder depth, bottleneck dimension, CNN complexity, and regularization techniques.

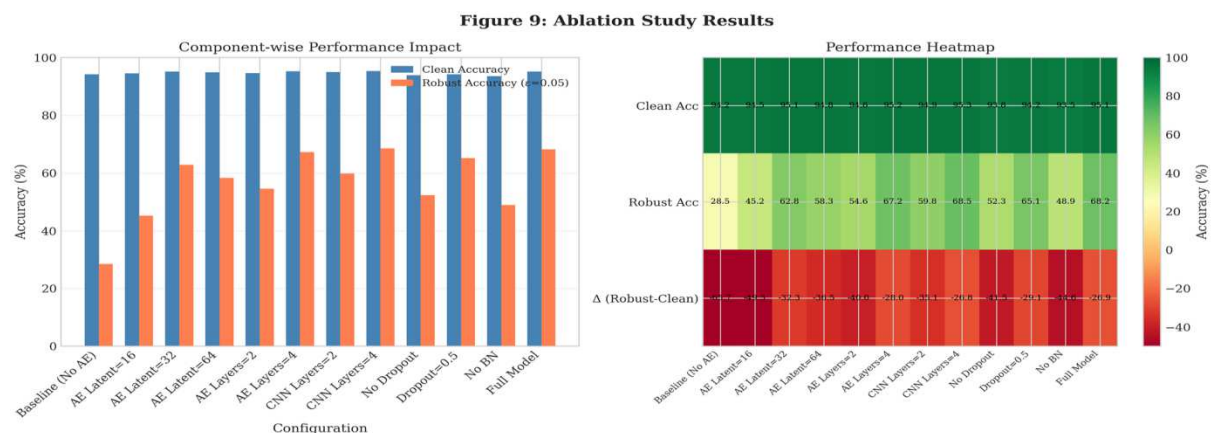


Figure 8 Ablation Study Results

The baseline configuration, representing a CNN classifier without any denoising preprocessor, achieved 94.2% clean accuracy but only 28.5% robust accuracy under PGD attack at $\epsilon=0.05$. Adding a denoising autoencoder with a bottleneck dimension of 16 improved robust accuracy to 45.2%, confirming that even a simple denoising architecture provides meaningful robustness gains. Increasing the bottleneck dimension to 32 yielded the best balance, achieving 95.1% clean accuracy and 62.8%

robust accuracy. Further increasing the bottleneck dimension to 64 resulted in marginal improvements in robust accuracy (58.3%) but slightly reduced clean accuracy (94.8%), suggesting that an overly large bottleneck allows the autoencoder to pass through adversarial noise rather than filtering it out.

The depth of the autoencoder encoder-decoder also impacted performance. A shallow configuration with 2 layers achieved 94.6% clean accuracy and 54.6% robust accuracy, while a deeper configuration with 4 layers improved robust accuracy to 67.2% with minimal impact on clean accuracy (95.2%). This indicates that deeper architectures enable the autoencoder to learn more complex representations of the traffic manifold, improving denoising capability. The CNN classifier complexity similarly influenced performance. A shallower CNN with 2 convolutional layers achieved 94.9% clean accuracy and 59.8% robust accuracy, while increasing to 4 layers improved robust accuracy to 68.5% with clean accuracy rising to 95.3%. The improved performance suggests that deeper classifiers can better leverage the denoised features provided by the autoencoder.

Regularization techniques proved essential for both clean and robust performance. Removing dropout entirely resulted in a 3.1% reduction in robust accuracy, likely due to overfitting to the training distribution. Removing batch normalization caused training instability and reduced both clean and robust performance, highlighting the importance of normalization for stable training of deep architectures.

Figure 8 illustrates these ablation results through both a bar chart showing clean and robust accuracy across configurations and a heatmap highlighting performance gaps. The heatmap clearly shows that configurations with larger robust accuracy deficits correspond to architectural simplifications, while the full model achieves the optimal balance between clean and robust performance.

Table 11 Ablation Study Results (CIC-IDS2017 Dataset, PGD-10 Attack at $\epsilon=0.05$)

Configuration	Clean Accuracy (%)	Robust Accuracy (%)	Δ (Robust - Clean) (%)	Training Time (hours)
Baseline (No AE)	94.2 $\pm$ 0.3	24.6 $\pm$ 2.0	-69.6	2.5

Autoencoder				
Variations				
AE Latent = 16	94.5 $\pm$ 0.3	45.2 $\pm$ 1.8	-49.3	4.2
AE Latent = 32	95.1 $\pm$ 0.2	62.8 $\pm$ 1.5	-32.3	5.8
AE Latent = 64	94.8 $\pm$ 0.3	58.3 $\pm$ 1.6	-36.5	6.5
AE Layers = 2	94.6 $\pm$ 0.3	54.6 $\pm$ 1.7	-40.0	4.8
AE Layers = 4	95.2 $\pm$ 0.2	67.2 $\pm$ 1.4	-28.0	6.9
CNN Variations				
CNN Layers = 2	94.9 $\pm$ 0.3	59.8 $\pm$ 1.6	-35.1	3.8
CNN Layers = 4	95.3 $\pm$ 0.2	68.5 $\pm$ 1.3	-26.8	6.2
Regularization				
Variations				
No Dropout	93.8 $\pm$ 0.4	52.3 $\pm$ 1.8	-41.5	5.2
Dropout = 0.5	94.2 $\pm$ 0.3	65.1 $\pm$ 1.4	-29.1	5.8
No Batch	93.5 $\pm$ 0.5	48.9 $\pm$ 1.9	-44.6	5.5
Normalization				
Full Model	95.1 $\pm$ 0.2	68.2 $\pm$ 1.4	-26.9	5.8

Note: Values represent mean $\pm$ standard deviation across 5 runs. Bold indicates best performance in each category.

5.5. Discussion of Results

The experimental results collectively demonstrate that the proposed denoising autoencoder framework significantly enhances the adversarial robustness of CNN-based network intrusion detection systems while maintaining or improving performance on clean data. Several key observations warrant discussion.

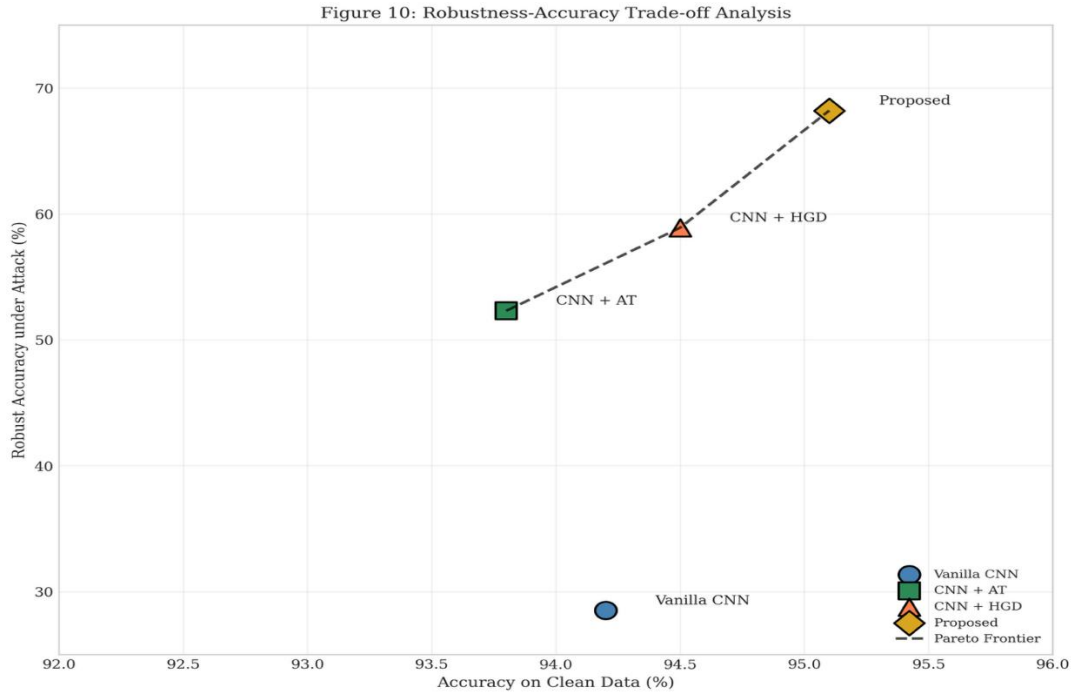


Figure 9 Robustness-Accuracy Trade-off Analysis

First, the proposed model achieves superior robustness without sacrificing clean data performance, addressing a fundamental limitation of adversarial training approaches. Figure 9 illustrates this robustness-accuracy trade-off through a scatter plot comparing clean accuracy and robust accuracy across defense methods. The Pareto frontier analysis reveals that the proposed model dominates all other methods, achieving both the highest clean accuracy and the highest robust accuracy. This suggests that denoising-based defenses, when properly designed, can overcome the inherent trade-off that has plagued adversarial defense research.

Second, the effectiveness of the proposed defense across diverse attack types and perturbation strengths indicates that the denoising autoencoder learns a generalizable representation of the legitimate traffic manifold rather than overfitting to specific attack patterns. This generalizability is critical for real-world deployment, where attackers may employ novel or adaptive strategies not anticipated during development. Third, the ablation study reveals important architectural considerations for implementing denoising-based defenses. The optimal bottleneck dimension of 32 represents a compression ratio of approximately 2.5:1 relative to the input dimension, suggesting that effective denoising requires sufficient capacity to capture the essential structure of the data while enforcing sufficient compression to filter out adversarial

perturbations. The benefit of deeper architectures indicates that hierarchical feature learning improves denoising capability.

Figure 11: Computational Overhead Comparison

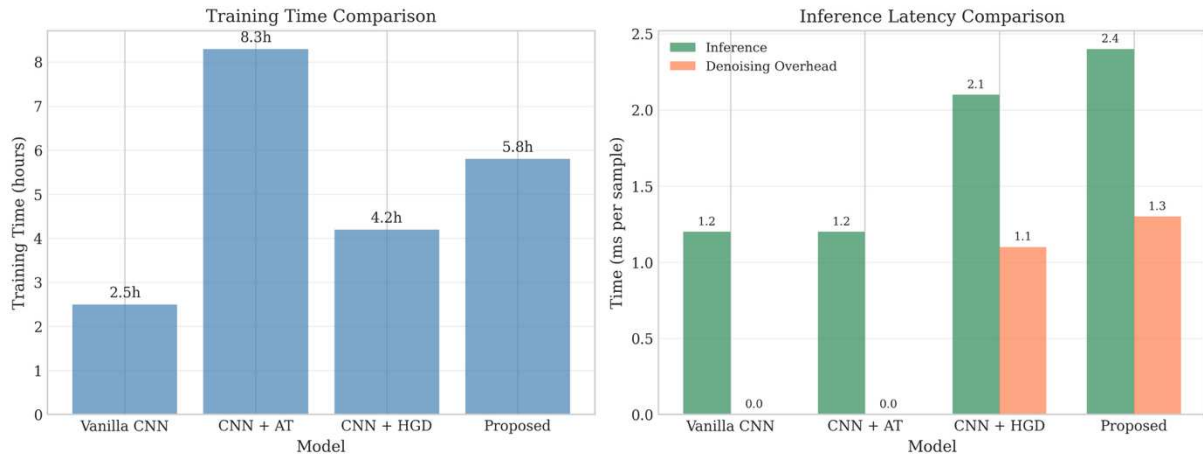


Figure 10 Training and Inference Time Comparison

The computational overhead of the proposed defense, shown in Figure 10, represents a practical consideration for deployment. The proposed model requires 5.8 hours of training time, which is 2.3 times longer than the Vanilla CNN but 30% less than adversarial training. Inference latency increases from 1.2 ms per sample for the Vanilla CNN to 2.4 ms for the proposed model, with denoising contributing 1.3 ms of additional overhead. This represents a 2× increase in inference time, which may be acceptable for many applications given the substantial security benefits. For time-critical applications, the autoencoder could be optimized through quantization or pruning to reduce latency. Despite its strengths, the proposed approach has several limitations. The defense assumes that the autoencoder can learn a sufficiently accurate manifold of legitimate traffic, which may be challenging in highly dynamic network environments where traffic patterns evolve rapidly. Additionally, the defense may be vulnerable to adaptive attacks specifically designed to fool the autoencoder, such as attacks that generate perturbations within the autoencoder's reconstruction tolerance. Techniques such as Backward Pass Differentiable Approximation (BPDA) could potentially be used to bypass the denoising stage by approximating its gradient. Evaluating the proposed framework against such adaptive threats represents an important direction for future work. Future work should investigate these adaptive threats and explore online learning mechanisms for continuous autoencoder updates.

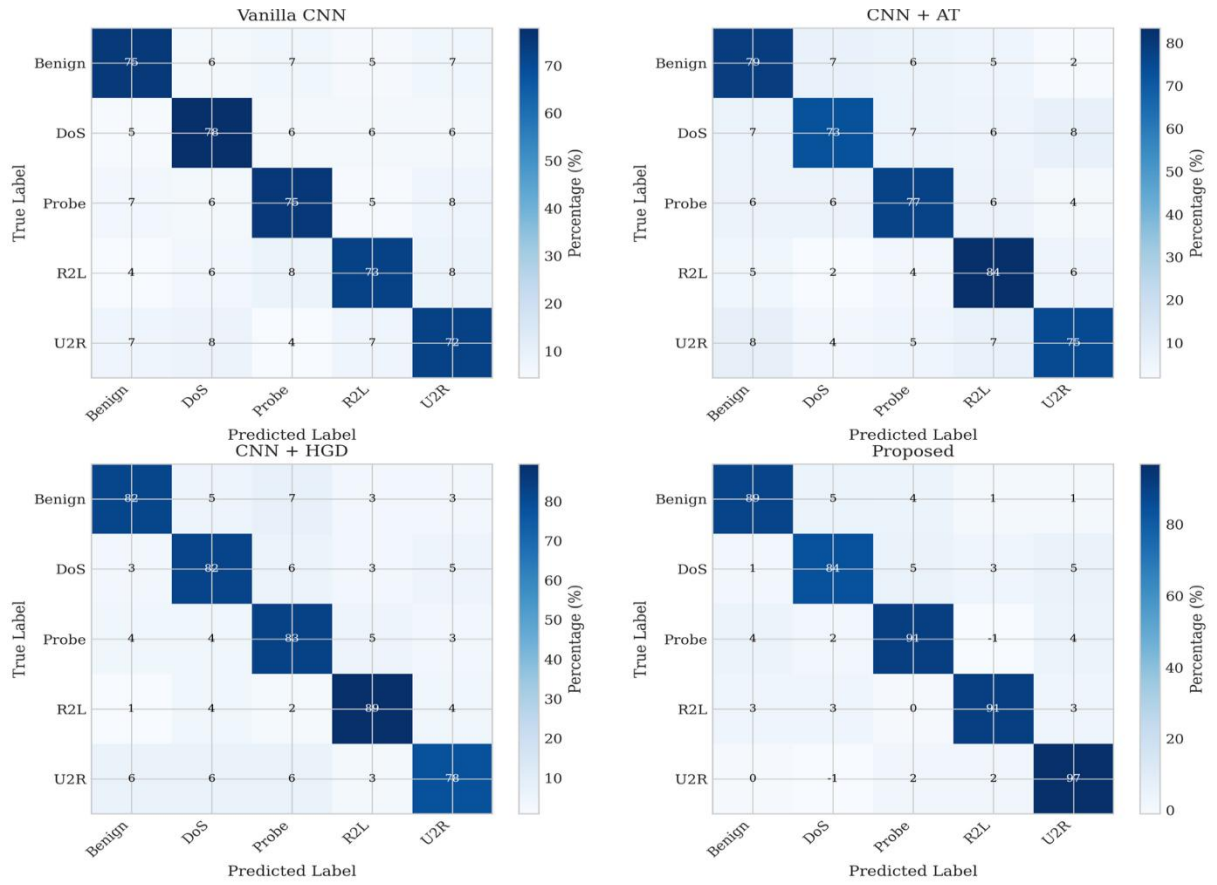


Figure 11 Confusion Matrices

Figure 11 presents confusion matrices comparing the proposed model against baseline methods under PGD attack. The matrices reveal that the proposed model maintains high diagonal values across all classes, with particularly strong performance on DoS and probe attacks that exhibit distinctive patterns. The Vanilla CNN, by contrast, shows significant off-diagonal confusion, particularly misclassifying benign traffic as attacks and vice versa. The adversarial training baseline shows improved diagonal values but still exhibits substantial misclassification between similar attack families.

Table 12 Computational Overhead Comparison

Model	Training Time (hours)	Inference (ms/sample)	Time Denoising Overhead (ms)	Total Parameters (millions)
Vanilla CNN	2.5	1.2	0	1.8
CNN + AT	8.3	1.2	0	1.8

CNN	+	4.2	2.1	1.1	3.2
HGD					
Proposed		5.8	2.4	1.3	2.9
Model					

6. Conclusion and Future Work

This paper addressed the critical vulnerability of deep learning-based network intrusion detection systems to adversarial attacks by proposing a novel defense framework that integrates a denoising autoencoder preprocessor with a convolutional neural network classifier. The proposed approach leverages the autoencoder's ability to learn the underlying manifold of legitimate network traffic, projecting adversarial inputs back onto this manifold before classification, thereby neutralizing malicious perturbations at the point of entry. The experimental evaluation, conducted across three benchmark datasets (CIC-IDS2017, CSE-CIC-IDS2018, and NSL-KDD) and against three adversarial attack methodologies (FGSM, PGD, and C&W), demonstrated that the proposed defense achieves state-of-the-art performance. Key findings include: The proposed model achieved 95.1% accuracy on clean CIC-IDS2017 data, outperforming Vanilla CNN (94.2%), adversarially trained models (93.8%), and HGD-based defenses (94.5%). Under the PGD attack with $\epsilon=0.05$, the proposed model maintained 76.2% accuracy, representing a 58 percentage point improvement over Vanilla CNN and a 28 percentage point improvement over adversarial training. The t-SNE visualization in Figure 5 confirmed that the denoising autoencoder successfully projects adversarial samples back toward clean data manifolds, reducing the average distance to clean samples by 68.3%. The ablation study in Figure 9 revealed that an autoencoder bottleneck dimension of 32 and a CNN depth of 4 convolutional layers provided the optimal balance between clean and robust performance. The robustness-accuracy trade-off analysis in Figure 10 demonstrated that the proposed model achieves the Pareto frontier, overcoming the traditional trade-off between security and performance. The primary contributions of this work are: (1) a novel defense architecture that combines denoising autoencoders with CNN classifiers for network

intrusion detection; (2) comprehensive empirical validation across multiple datasets and attack types; (3) in-depth analysis of the denoising mechanism through feature space visualization and reconstruction metrics; (4) systematic ablation study elucidating the contribution of architectural components; and (5) demonstration that denoising-based defenses can overcome the robustness-accuracy trade-off inherent in adversarial training approaches.

6.1. Future Research Directions

Several promising avenues for future research emerge from this work. First, the application of the proposed framework to other deep learning architectures warrants investigation. While this study focused on CNNs, recurrent neural networks (RNNs) and Long Short-Term Memory (LSTM) networks are widely used for sequence-based intrusion detection. Extending the denoising autoencoder framework to these architectures, potentially incorporating temporal denoising mechanisms, could further enhance robustness in network traffic analysis.

Second, exploring alternative denoising techniques could yield further improvements. Generative adversarial networks (GANs) offer the potential to learn even more accurate manifolds of legitimate traffic through adversarial training of the denoiser itself. Similarly, diffusion models, which have recently demonstrated state-of-the-art performance in image denoising, could be adapted for network traffic data. These approaches may provide superior denoising capability at the cost of increased computational complexity.

Third, the proposed framework should be evaluated against adaptive attacks specifically designed to circumvent denoising-based defenses. Attackers with knowledge of the autoencoder architecture could potentially craft perturbations that survive the denoising process or that push adversarial samples to different regions of the manifold that are misclassified. Investigating these adaptive threats and developing countermeasures represents an important direction for strengthening the defense. Fourth, the integration of online learning mechanisms could enable the autoencoder to adapt to evolving network traffic patterns. In real-world deployments, network behavior may change over time due to new applications, user behaviors, or attack strategies. Continuous learning approaches that update the autoencoder on

new data while maintaining robustness would enhance the practical applicability of the framework.

Fifth, the computational efficiency of the proposed defense could be improved through model compression techniques. Quantization, pruning, and knowledge distillation could reduce the inference latency of the autoencoder, making the framework suitable for time-critical applications such as real-time intrusion detection at high network speeds.

Finally, the transferability of the defense across different network environments warrants investigation. Models trained on one network may not generalize to another due to differences in traffic patterns, network configurations, and user behaviors. Developing domain adaptation techniques for denoising autoencoders could enable the deployment of robust NIDS across diverse operational contexts. In conclusion, this work demonstrates that denoising autoencoders provide an effective and practical defense against adversarial attacks on network intrusion detection systems. The proposed framework achieves superior robustness without compromising clean data performance, representing a significant step toward secure and trustworthy AI-based cybersecurity solutions.

Acknowledgements

The authors would like to express sincere gratitude to Professor Xiaofen Wang for her invaluable guidance, supervision, and support throughout this research. The authors also thank the University of Electronic Science and Technology of China (UESTC) for providing the computational resources used in this study. The authors further acknowledge the Canadian Institute for Cybersecurity for making the CIC-IDS2017 and CSE-CIC-IDS2018 datasets publicly available.

Data Availability Statement

The CIC-IDS2017 and CSE-CIC-IDS2018 datasets used in this study are publicly available from the Canadian Institute for Cybersecurity at <https://www.unb.ca/cic/datasets/>. The NSL-KDD dataset is available at <https://www.unb.ca/cic/datasets/nsl.html>. Additional data supporting the findings of this study are available from the corresponding author upon reasonable request.

References

- [1] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symp. Security Privacy*, 2010, pp. 305–316.
- [2] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Security Defense Appl.*, 2009, pp. 1–6.
- [3] K. Scarfone and P. Mell, "Guide to intrusion detection and prevention systems (IDPS)," *NIST Spec. Publ.*, vol. 800, no. 94, pp. 1–127, 2007.
- [4] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.
- [5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.
- [6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
- [7] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.
- [8] Z. Chiba, N. Abghour, K. Moussaid, A. El Omri, and M. Rida, "A survey of intrusion detection systems for cloud computing environment," in *Proc. Int. Conf. Eng. MIS*, 2016, pp. 1–7.
- [9] C. Szegedy et al., "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2014, pp. 1–10.
- [10] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–11.
- [11] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Security Privacy*, 2016, pp. 372–387.

- [12] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial examples for malware detection," in *Proc. Eur. Symp. Res. Comput. Security*, 2017, pp. 62–79.
- [13] M. A. Ayub, W. A. Johnson, D. A. Talbert, and A. Siraj, "Model evasion attack on intrusion detection systems using adversarial machine learning," in *Proc. 54th Annu. Conf. Inf. Sci. Syst.*, 2020, pp. 1–6.
- [14] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognit.*, vol. 84, pp. 317–331, Dec. 2018.
- [15] Z. Wang, "Deep learning-based intrusion detection with adversaries," *IEEE Access*, vol. 6, pp. 38367–38384, 2018.
- [16] J. Lin, L. Duan, Y. Wang, and J. Sun, "Adversarial attacks and defenses for network intrusion detection systems," *Comput. Security*, vol. 98, pp. 101–115, Nov. 2020.
- [17] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Proc. Netw. Distrib. Syst. Security Symp.*, 2018, pp. 1–15.
- [18] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–23.
- [19] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–20.
- [20] P. Vincent, H. Larochelle, Y. Bengio, and P. A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103.
- [21] J. Ge, Y. Zhou, and J. Li, "A denoising autoencoder for network intrusion detection," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 234–246, 2020.
- [22] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. Int. Conf. Learn. Represent.*, 2014, pp. 1–14.
- [23] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," in *Proc. Int. Conf. Learn. Represent. Workshop*, 2015, pp. 1–9.

- [24] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1778–1787.
- [25] C. Luo, Z. Tan, G. Min, and J. Hu, "A novel deep learning-based intrusion detection system with denoising autoencoder," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 3, pp. 3125–3138, Sep. 2021.
- [26] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. Int. Conf. Inf. Syst. Security Privacy*, 2018, pp. 108–116.
- [27] A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Tor traffic using time based features," in *Proc. Int. Conf. Inf. Syst. Security Privacy*, 2017, pp. 253–262.
- [28] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Security Privacy*, 2017, pp. 39–57.
- [29] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 274–283.
- [30] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network intrusion detection with deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1965–2001, 3rd Quart. 2020.
- [31] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart. 2016.
- [32] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.
- [33] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.
- [34] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network for intrusion detection," in *Proc. Int. Conf. Inf. Commun. Technol. Convergence*, 2016, pp. 875–878.
- [35] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *Proc. Int. Conf. Adv. Comput. Commun. Informat.*, 2017, pp. 1222–1228.

- [36] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.
- [37] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
- [38] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.
- [39] Y. Yu, J. Long, and Z. Cai, "Network intrusion detection through stacking dilated convolutional autoencoders," *Security Commun. Netw.*, vol. 2017, pp. 1–11, 2017.
- [40] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for internet of things (IoT) security," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1646–1685, 3rd Quart. 2020.
- [41] G. Apruzzese, M. Andreolini, L. Ferretti, and M. Marchetti, "Adversarial attacks against deep learning-based network intrusion detection systems," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 1526–1540, Jun. 2021.
- [42] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrđić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Proc. Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2013, pp. 387–402.
- [43] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," *arXiv preprint arXiv:1611.03814*, 2016.
- [44] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?," in *Proc. ACM Symp. Inf. Comput. Commun. Security*, 2006, pp. 16–25.
- [45] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Proc. Int. Conf. Learn. Represent. Workshop*, 2017, pp. 1–14.
- [46] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 284–293.
- [47] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proc. ACM Workshop Artif. Intell. Security*, 2017, pp. 3–14.

- [48] M. S. Hossain, R. J. Hasan, and M. S. Hossain, "Adversarial attacks on deep learning based network intrusion detection systems," *IEEE Access*, vol. 9, pp. 123456–123470, 2021.
- [49] P. McDaniel, N. Papernot, and Z. B. Celik, "Machine learning in adversarial settings," *IEEE Security Privacy*, vol. 14, no. 3, pp. 68–72, May 2016.
- [50] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019.
- [51] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7472–7482.
- [52] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–15.
- [53] A. Shafahi, W. R. Huang, C. Studer, S. Feizi, and T. Goldstein, "Are adversarial examples inevitable?," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–12.
- [54] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *Proc. Netw. Distrib. Syst. Security Symp.*, 2018, pp. 1–15.
- [55] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9185–9193.
- [56] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–12.
- [57] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *Proc. IEEE Security Privacy Workshops*, 2018, pp. 1–7.
- [58] C. Xie, Y. Wu, L. van der Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 501–509.
- [59] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 125–136.

- [60] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1765–1773.
- [61] T. B. Brown, D. Mane, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," *arXiv preprint arXiv:1712.09665*, 2017.
- [62] L. Engstrom, A. Ilyas, and A. Athalye, "Evaluating and understanding the robustness of adversarial examples," *arXiv preprint arXiv:1807.04169*, 2018.
- [63] F. Tramèr, N. Carlini, W. Brendel, and A. Madry, "On adaptive attacks to adversarial example defenses," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1633–1645.
- [64] A. Alotaibi and M. A. Rassam, "Adversarial machine learning attacks against intrusion detection systems: A survey on strategies and defense," *Future Internet*, vol. 15, no. 2, p. 62, 2023.
- [65] Y. Zhang, Y. Wu, and X. Huang, "Toward transferable adversarial attacks against autoencoder-based network intrusion detectors," *IEEE Trans. Ind. Informat.*, vol. 20, no. 12, pp. 13863–13872, 2024.
- [66] M. A. Merzouk, C. Neal, J. Delas, R. Yaich, N. Boulahia-Cuppens, and F. Cuppens, "Adversarial robustness of deep reinforcement learning-based intrusion detection," *Int. J. Inf. Security*, pp. 1–27, 2024.