electronics

Article

Understanding and Detecting Adversarial Examples in IoT Networks: A White-Box Analysis with Autoencoders

Wafi Danesh, Srinivas Rahul Sapireddy and Mostafizur Rahman *

School of Science and Engineering, Division of Energy, Matter and Systems, University of Missouri, Kansas City, MO 64110, USA; wdhv3@mail.umkc.edu (W.D.); ssdx5@mail.umkc.edu (S.R.S.)
* Correspondence: rahmanmo@mail.umkc.edu

Abstract

Novel networking paradigms such as the Internet of Things (IoT) have expanded their usage and deployment to various application domains. Consequently, unseen critical security vulnerabilities such as zero-day attacks have emerged in such deployments. The design of intrusion detection systems for IoT networks is often challenged by a lack of labeled data, which complicates the development of robust defenses against adversarial attacks. As deep learning-based network intrusion detection systems, network intrusion detection systems (NIDS) have been used to counteract emerging security vulnerabilities. However, the deep learning models used in such NIDS are vulnerable to adversarial examples. Adversarial examples are specifically engineered samples tailored to a specific deep learning model; they are developed by minimal perturbation of network packet features, and are intended to cause misclassification. Such examples can bypass NIDS or enable the rejection of regular network traffic. Research in the adversarial example detection domain has yielded several prominent methods; however, most of those methods involve computationally expensive retraining steps and require access to labeled data, which are often lacking in IoT network deployments. In this paper, we propose an unsupervised method for detecting adversarial examples that performs early detection based on the intrinsic characteristics of the deep learning model. Our proposed method requires neither computationally expensive retraining nor extra hardware overhead for implementation. For the work in this paper, we first perform adversarial example generation on a deep learning model using autoencoders. After successful adversarial example generation, we perform adversarial example detection using the intrinsic characteristics of the layers in the deep learning model. A robustness analysis of our approach reveals that an attacker can easily bypass the detection mechanism by using low-magnitude log-normal Gaussian noise. Furthermore, we also test the robustness of our detection method against further compromise by the attacker. We tested our approach on the Kitsune datasets, which are state-of-the-art datasets obtained from deployed IoT network scenarios. Our experimental results show an average adversarial example generation time of 0.337 s and an average detection rate of almost 100%. The robustness analysis of our detection method reveals a reduction of almost 100% in adversarial example detection after compromise by the attacker

Keywords: adversarial machine learning; deep learning; unsupervised learning

Academic Editors: George Angelos Papadopoulos and Andreas Mauthe

Received: 6 June 2025
Revised: 21 July 2025
Accepted: 27 July 2025
Published: 29 July 2025

Citation: Danesh, W.; Sapireddy, S.R.; Rahman, M. Understanding and Detecting Adversarial Examples in IoT Networks: A White-Box Analysis with Autoencoders. *Electronics* **2025**, *14*, 3015. https://doi.org/10.3390/electronics14153015

1. Introduction

The emergence and rapid incorporation of novel computing systems such as the Internet of Things (IoT), cloud computing, edge computing, and fog computing into various application domains has given rise to unforeseen security vulnerabilities [1]. Considering IoT systems, it is estimated that there will be 3.74 billion IoT mobile connections worldwide by 2025 and more than 64 billion IoT devices installed by 2026 [2–5]. IoT networks are generally resource-constrained, incorporate heterogeneous integration, and have a distributed deployment architecture. Due to resource constraints, IoT networks cannot incorporate conventional security measures such as encryption, authentication, access control, network, and access security [6]. Consequently, new attack vectors are exposed that are prone to exploitation by unforeseen intrusions, particularly zero-day attacks [7]. In addition, IoT networks generally lack sufficient labeled data due to a lack of domain expertise in the various deployment scenarios. Network intrusion detection systems (NIDS) for IoT networks have been developed as a last line of defense to counteract such potential intrusions. Network intrusion detection systems (NIDS) are broadly classified as signature-based, anomaly-based, or hybrid. For IoT networks, anomaly-based NIDS incorporating machine learning (ML) or deep learning (DL) methods are most effective against zero-day attacks. Anomaly-based NIDS develops a profile for normal behavior of the IoT network without intrusions, with which new incoming network traffic can then be compared. Any deviation from the normal behavior profile beyond a defined threshold indicates a potential network intrusion [3–5,7–15]. Deep learning methods, particularly unsupervised methods, are well suited for intrusion detection in IoT networks due to their inherent ability to handle large volumes of data, extract useful features, and provide results with high accuracy. Unsupervised deep learning methods are essential for IoT intrusion detection, as such deployments often lack labeled data. The computational complexity and deployment overhead for common deep learning applications such as computer vision and natural language processing is unnecessary for anomaly detection in network traffic [14]. Despite the effectiveness of deep learning models for intrusion detection in IoT systems, such models are vulnerable to adversarial examples [16]. Adversarial examples are specially crafted observations into which an imperceptible perturbation has been introduced which causes the deep learning model to misclassify the input samples. Such malicious examples have been studied in application domains such as computer vision [17], natural language processing [18], speech recognition [19], and reinforcement learning [20]. Adversarial examples have also been studied extensively in the network intrusion detection domain. Adversarial examples can cause malicious network traffic to bypass NIDS or reject normal network traffic, leading to denial of service (DoS). However, key differences persist in network intrusion detection compared with other application domains. Generating adversarial examples in network intrusion detection is extremely complex due to the difficulty of identifying appropriate network packet attributes to be modified and deciding on the modification of identified network attributes to generate adversarial examples [21]. Adversarial example detection in NIDS for IoT networks is an emerging area of research. Several key methods have been proposed for detection; however, most focus on adversarial training to improve the robustness of NIDS against adversarial attacks, which can be time-consuming and computationally intensive. The lack of labeled data in IoT network scenarios also adds to the complexity of the detection process. Furthermore, most adversarial training methods require complex deep learning architectures, which can entail extra hardware overhead that is not possible in the resource-constrained environment of IoT network deployment. This paper proposes an unsupervised adversarial example detection method for deep learning-based intrusion detection systems in IoT networks. Our proposed method performs early detection using the intrinsic characteristics of the layers of the implemented deep learning model. No computationally intensive adversarial training is required in our approach. In addition, as our proposed method is based on the intrinsic characteristics of the deep learning model layers, it bypasses the need for potential extra hardware overhead for implementation. By reclaiming security, we refer to defenders' efforts to restore trust in intrusion detection systems operating under adversarial threats, specifically through early and unsupervised detection techniques that require no retraining. To summarize, the main contributions of this paper are as follows:

- We propose an unsupervised early detection technique for adversarial example detection which leverages the intrinsic characteristics of the hidden layers of deep learning architectures.
- We evaluate the robustness of the proposed early detection technique against potential attackers' attempts to bypass or compromise the system.

We implemented our proposed adversarial example detection on one of the Kitsune datasets. Kitsune datasets are state-of-the-art datasets for network intrusion detection obtained from practical real-time IoT network deployments. The dataset used for experimentation is derived from IoT network scenarios and consists of 100 features. The Kitsune dataset is specifically designed to efficiently detect patterns in network traffic by monitoring its flow. In this paper, we detect these patterns using encoders in combination with ensemble methods. The Kitsune dataset comprises 20 statistical parameters recorded over a window of 5 time steps. This network traffic dataset is used for intrusion and anomaly detection. Each row in the dataset represents a packet, which can contain either malicious or regular traffic. When adversarial attacks are introduced, the number of malicious packets increases, leading to a rise in root mean square error (RMSE) values. We calculate the highest RMSE observed after the attacker injects malicious data during training.

2. Related Work

The key characteristic of adversarial examples is the ability to force an ML/DL classifier to generate wrong predictions. Adversarial examples can be categorized as anomalies, specifically intentional anomalies [22], since they must be engineered by perturbation of specific critical features. As shown in Figure 1, the adversarial examples, labeled by the red data points, can easily traverse the decision boundary of the implemented deep learning model with minimal disturbance. In other words, as can be observed from Figure 1, the data points close to the decision boundary for each class are suitable candidates for adversarial example generation. In the present context, we focus on detecting adversarial examples in network intrusion detection for IoT networks.

Network intrusion detection, in the context of anomaly detection can be considered a binary classification problem. Adversarial examples, categorized as intentional anomalies, modify the network traffic surreptitiously to cause misclassification. Normal network traffic can be misclassified as malicious and cause a denial of service, while malicious network traffic can be identified as normal, enabling man-in-the-middle attacks. Adversarial example detection in network intrusion detection for IoT networks is an emerging research area. This section briefly reviews some of the recent literature in this area.

In [23], the authors propose a framework termed FGMD (Feature Grouping and Multi-model fusion Detector), which defends against adversarial attacks through feature grouping and multi-model fusion. Generating realistic adversarial examples in the NIDS scenario is crucial. In [24], the authors proposed A2PM (Adaptive Perturbation Pattern Method), which generates adversarial samples in a gray-box setting. In [25], an ensemble technique was proposed that uses a two-module framework consisting of a multi-class generative adversarial network and multi-source adversarial retraining for detecting adversarial examples. In this method, the multi-class detector is retrained only once. The authors of [26] proposed a neural network architecture named DiPSeN (Differentially Private Self-Normalizing Neural Network), which combines the characteristics of differential privacy, self-normalization, and a novel optimization algorithm for adversarial defense in federated learning for IoT networks. In [27], the authors proposed using adversarial training for various machine learning classifiers in supervised learning to increase the robustness against denial-of-service (DoS) attacks. In [28], Fu et al. used three different training methods for three intrusion detection models based on convolutional neural networks (CNN), long short-term memory (LSTM), and gated recurrent units (GRU) to improve the robustness of supervised adversarial example detection.

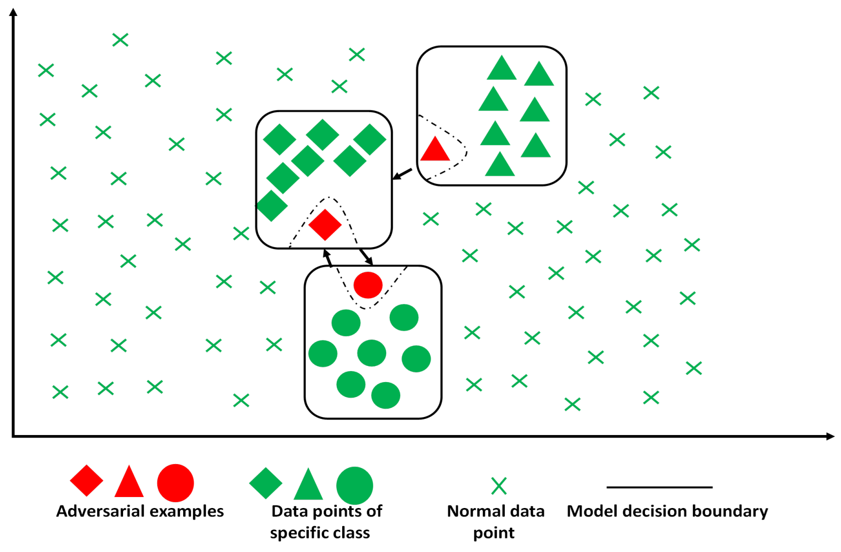

Figure 1. Definition of adversarial examples. Adversarial examples (in red) are generated by adding small perturbations to valid data points (green) from specific classes. The dotted lines indicate the local decision boundaries of the classifier, and the arrows illustrate how perturbations shift normal inputs across these boundaries, resulting in misclassification.

In [29], adversarial examples for spatiotemporal data were generated using both white-box and black-box methods based on travel time estimation (TTE). The authors introduced small perturbations to the input data, which remain imperceptible to humans but can disrupt deep learning models. In [30], adversarial examples were created using the fast gradient sign method (FGSM). The trajectory data were then transformed into image format using convolutional autoencoders (CAE), enhancing data security within deep learning models. Recent studies have also highlighted emerging adversarial vectors in IoT systems, including side-channel attacks such as FOAP and RF energy harvesting [31,32], as well as physical patch attacks targeting sensor systems [33]. These threats further underscore the need for robust and adaptive detection mechanisms in resource-constrained deployments.

As mentioned above, adversarial example detection in NIDS for IoT networks is an emerging research area that has yet to be extensively investigated. Among the recent works in the preceding brief overview, it can be ascertained that most methods focus on adversarial training to improve robustness against adversarial attacks. Adversarial training can be a time-consuming and computationally intensive process; furthermore, in the case of IoT networks, many applications lack the necessary domain expertise due to deployment in environments designed to minimize human intervention. Therefore, our proposed method uses an unsupervised deep learning algorithm for adversarial example detection in IoT networks. Our proposed method incurs no extra hardware overhead, since we perform adversarial example detection using the intrinsic characteristics of the deep learning architecture. Unsupervised learning is particularly useful in the context of IoT networks, as a large amount of labeled data corresponding to normal behavior is usually available for IoT applications, while labels for malicious behavior are not; in addition, unsupervised intrusion detection approaches can be deployed to IoT networks in isolated environments where domain expertise is lacking.

3. Attack Model

In general, network intrusions can be categorized as affecting the network's confidentiality (C), integrity (I), and availability (A). An attacker's objective in generating adversarial examples in the network intrusion detection context is to compromise the intrusion detection process, and consequently the NIDS. Adversarial attacks can be categorized into three distinct categories [27]:

- Black-box attack: The attacker does not know the deep learning model implemented in the target NIDS.
- White-box attack: The attacker has complete knowledge of the deep learning model in the target NIDS, including the architecture, input data, and features.
- Gray-box attack: The attacker has partial knowledge of the deep learning model in the target NIDS, including the architecture, input data, and features.

We assume a white-box attack model for the adversarial example detection method implemented in this paper; hence, the attacker is assumed to be cognizant of all the characteristics of the deep learning model implemented in the target NIDS, such as the model architecture, hyperparameters, and type of network traffic entering the NIDS. The attacker first captures the network traffic flowing through the gateway device, such as the router or switch, to generate adversarial examples. In general, NIDS are implemented at gateway devices, since they act as entry points into a network. The captured network traffic is analyzed to determine the critical features in the network packets. In the network intrusion detection context, the critical features are those which can cause the implemented deep learning model to misclassify incoming network traffic with minimal perturbation. In this study, adversarial examples are generated after the training phase and used exclusively during evaluation to test the detection capability of our proposed method. These examples are not included in the training dataset and do not influence model learning, maintaining the unsupervised nature of our detection strategy. In the next step, the attacker generates adversarial examples using an algorithm and injects the generated examples into the target NIDS. The attacker can choose to either perform a DoS attack, causing benign packets to be classified as malicious, or an evasion attack, causing malicious packets to be classified as benign. Figure 2 shows the attack model used for generating adversarial examples in the network intrusion detection context. As shown in Figure 2, traffic data are collected through a router and transmitted to a remote site. In this system, traffic is labeled as malicious after an attack is identified; Table 1 summarizes the approach adversarial example generation implemented in this work.

Table 1. Types of adversarial example generation in the current work.

Type of Network Attack	Type of Adversarial Attack	Adversarial Example Generation Method	Effect on Network Traffic	Aftermath on Networked System
Man-in-the-middle	White-box	Fast gradient sign method (FGSM)	Benign traffic classified as malicious	Denial-of-Service
Man-in-the-middle	White-box	Fast gradient sign method (FGSM)	Malicious traffic classified as benign	Evasion of NIDS

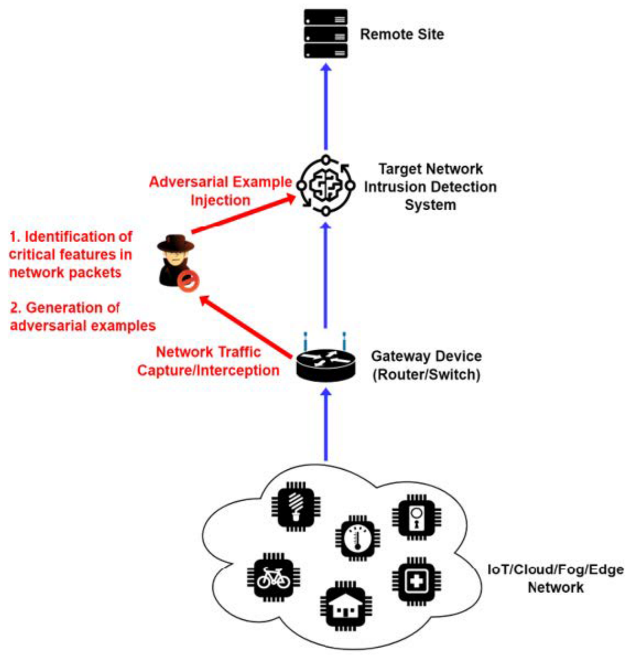
Remote Site
Adversarial Example Injection
Target Network Intrusion Detection System
1. Identification of critical features in network packets
2. Generation of adversarial examples
Network Traffic Capture/Interception
Gateway Device (Router/Switch)
IoT/Cloud/Fog/Edge Network

Figure 2. Adversarial example generation and injection into target NIDS.

We focus on white-box attacks in this study because they represent the most challenging and high-risk adversarial scenario, i.e., where the attacker has full knowledge of the model architecture, hyperparameters, and data distribution. Evaluating our method under this strong threat model establishes a robust baseline for its effectiveness. In the broader context of adversarial defense, supervised techniques such as MagNet and Defense-GAN utilize auxiliary classifiers or input reconstruction, but often require labeled datasets and retraining, which are impractical for lightweight IoT deployments. Unsupervised methods such as feature squeezing can detect perturbations through input transformations, but may lack early detection capability. In contrast, our method leverages the clustering of hidden-layer activations for early unsupervised detection, offering a low-overhead solution that is well suited for real-time operation in resource-constrained environments. We adopt the white-box threat model to evaluate our method under the most demanding conditions, namely, where the attacker fully understands the model. This helps to assess the true limits of detection performance. Stacked autoencoders are widely used in IoT NIDS because they can learn typical traffic patterns without labeled data. Their lightweight design and ability to detect anomalies make them practical for real-time deployment in resource-constrained IoT systems.

Our proposed method was evaluated on one of the Kitsune datasets. These datasets contain statistical features of network traffic traversing deployed IoT networks in successive time windows. A total of nine datasets constitute the Kitsune datasets, in which the network attacks can be categorized as reconnaissance, denial-of-service, man-in-the-middle attack, and botnet malware [34]. In preprocessing, we first normalized the selected dataset. We used one of the Kitsune datasets obtained for our evaluation of a man-in-the-middle (MitM) attack, as in Table 1. The Kitsune datasets can be adapted for use in machine learning applications. To enhance early detection of adversarial attacks, we developed a generalized model capable of identifying adversarial examples early in the training process. This approach improves the system's security and reliability. The MitM attack was performed in an IP camera video surveillance network, with the attacker intercepting all LAN traffic through an ARP poisoning attack. Figure 3 provides the network topology from which the Kitsune dataset were obtained. In Figure 3, the labels 1, 2, 3, and X represent the locations where the attacker launches the attacks, and the label Kitsune represents the points from which network traffic was captured. The MitM attack in the current dataset affects the confidentiality of the network.

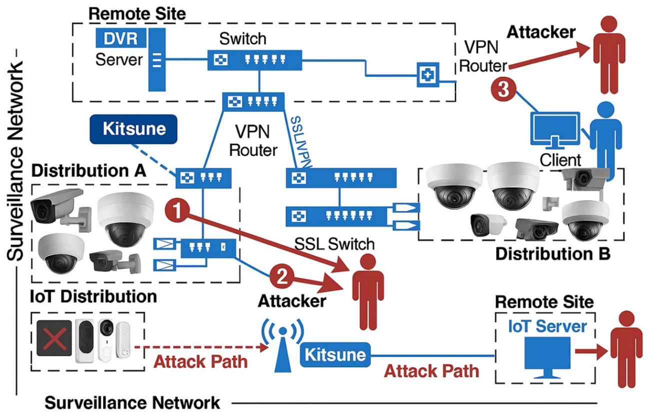

Figure 3. Network topologies used for obtaining the Kitsune datasets [34].

As shown in Table 1, a white-box attack model has been assumed in the current work for adversarial example generation. The method chosen for adversarial example generation was a modified version of the fast gradient sign method (FGSM) [16] called the iterative FGSM. Using this method, adversarial examples can be generated by minute perturbations of critical features in the direction of the gradient. Consequently, Table 1 shows that adversarial attacks can either cause benign traffic to be misclassified as malicious, or cause malicious traffic to be misclassified as benign. As a result, the aftermath of the first case manifests in a DoS attack and in the second case as an evasion attack. In the current work, we have considered the first case of adversarial example generation, which results in a DoS attack on the target network.

4. Proposed Methodology

For the current work, a stacked autoencoder (SAE) is assumed to be the deep learning algorithm implemented in the target NIDS. Autoencoders have been used relatively extensively in cybersecurity applications [35]. We perform anomaly-based intrusion detection using the SAE architecture. As can be observed in Figure 4, an autoencoder consists of two parts: an encoder and a decoder. In the encoder, successive dimensionality reductions are performed on the input vector to encode the input. The encoded input can be obtained from the bottleneck layer, which can be considered as separating the encoder and decoder. In the decoder, the dimensionality is successively increased to match that of the input vector. An approximate recreation of the input occurs at the output of the SAE. Although SAEs are categorized as unsupervised algorithms, they can be considered for performing self-supervised learning as well. The functionality of the SAE can be expressed mathematically as follows:

$$\lambda : X \to L, \tag{1}$$

$$\Upsilon : L \to X \tag{2}$$

$$\lambda, \Upsilon = \arg\min_{\lambda,\Upsilon} \|X - (\lambda \cdot \Upsilon)X\|^2. \tag{3}$$

In (1), at the encoder, the function denoted by λ maps the original data X to a latent space L. The output of this latent space is obtained at the bottleneck layer. At the decoder, the function denoted by Υ remaps the latent space L to the original space of the input data. As a result of this remapping process, the SAE is able to approximate the input at the output. The objective of the SAE is to minimize the loss between the original input and the reconstructed output, as shown in (3).

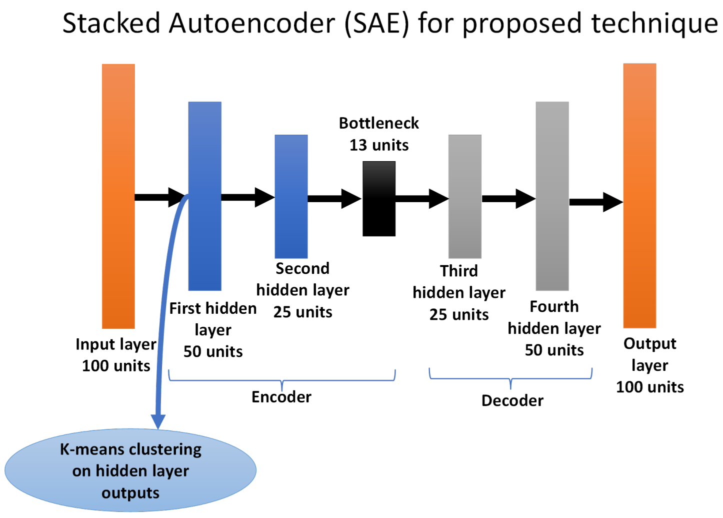
Stacked Autoencoder (SAE) for proposed technique
Bottleneck 13 units
Second hidden layer 25 units
Third hidden layer 25 units
Fourth hidden layer 50 units
First hidden layer 50 units
Input layer 100 units
Output layer 100 units
Encoder
Decoder
K-means clustering on hidden layer outputs

Figure 4. A stacked autoencoder (SAE) architecture with k-means clustering applied to the output of hidden layer.

With regard to the encoder, the transformation of the input vector into a vector in the latent space can be represented by an activation function σ, as follows:

$$z = \sigma(Wx + b) \tag{4}$$

where z is the vector in the latent space, W is the weight matrix, and b is the bias.

Similarly, at the decoder, the recreated output can be represented using a different activation function σ', as follows:

$$x' = \sigma'(W'z + b'). \tag{5}$$

The loss function used in backpropagation for training the SAE is expressed as follows:

$$\mathcal{L}(x, x') = \|x - x'\|^2 = \left\|x - \sigma'\left(W'(\sigma(Wx + b)) + b'\right)\right\|^2. \tag{6}$$

An intrusion in a networked system is a rare event relative to normal network operation. For anomaly-based intrusion detection, the SAE is first trained with normal examples of network traffic to learn the normal behavior profile. The loss function threshold value is determined and set during the training stage. After training, incoming network traffic is analyzed and the loss value is compared with the threshold. If the threshold is exceeded, the network traffic is identified as an anomaly and classified as an intrusion. Our current work's algorithmic approach to anomaly detection is described below.

Let the loss function used to determine the threshold be the root mean squared error (RMSE); if Φ is the anomaly threshold with an initial value of -1, $\beta \in [1, \infty)$ is a sensitivity parameter, and θ denotes the hyperparameters of the deep learning architecture implemented, then:

In the case of adversarial example generation in anomaly-based network intrusion detection, the attacker's objective is to bypass the alerting mechanism or increase the false alarm rate. In the proposed method, we consider the case of increased false alarms, which inevitably lead to denial of service. In addition, we consider white-box attacks for our adversarial example generation process. In the next subsections, we explain our method for adversarial example generation, our early detection methodology for adversarial examples, and our methodology for performing a robustness analysis of our approach.

4.1. Adversarial Example Generation

In network intrusion detection, adversarial example generation can be computationally expensive and extremely time-consuming compared to other application domains. Datasets for network intrusion detection consist of structured data that are very challenging to manipulate. Two key factors must be considered regarding adversarial example generation: identification of the appropriate network packet attributes to be modified, and stealthy modification of identified network attributes for generating necessary adversarial input vectors. In addition, malicious examples are located far from the decision boundary of normal examples, which compounds the difficulty of generating adversarial examples. For our proposed method, we first obtain the critical features for adversarial example generation using saliency maps [36]. As network intrusion detection is a binary classification problem, the saliency map of an input vector 'x' with 'i' features is provided by

$$\left[\mathrm{ReLU}\left(\frac{\delta p_c}{\delta x_i}\right)\right]_i, \tag{7}$$

where $\mathrm{ReLU} : x \to \max(0, x)$ is the rectified linear unit activation (ReLU) function applied to the input vector x and $c \in \{0, 1\}$ is the corresponding class of the input. The critical feature is the one for which the ReLU function for the opposite class yields a larger value, meaning that the predicted labels can be altered with minimal disturbance. After identifying the critical features, we perform adversarial example generation using the iterative version of the fast gradient sign method (FGSM) [16]. The FGSM generates adversarial examples by perturbing the critical features by a specific magnitude in the direction of the gradient. Based on the specific features obtained from the saliency map, the iterative FGSM algorithm, which has been modified for our current work, is outlined in Algorithm 1. The time taken to generate adversarial examples using the iterative FGSM method is within the interval of successive network packets arriving at the NIDS. Therefore, the iterative FGSM provides a less computationally intensive and more time-efficient method for adversarial example generation. In network intrusion detection, adversarial examples can either cause malicious samples to evade detection or cause benign samples to be rejected, resulting in denial of service. In this work, we consider the latter case and develop our detection approach accordingly. The ARP MitM scenario from the Kitsune dataset was chosen for its real-world relevance and rich set of statistical features, making it well suited for evaluating adversarial detection in IoT network environments.

4.2. Adversarial Example Detection

We propose an early detection approach to detect adversarial examples. Our proposed approach is unsupervised and uses the inherent characteristics of the layers of the stacked autoencoder (SAE). No additional hardware overhead is encountered, and computationally intensive retraining is unnecessary for our proposed detection method. In this approach, the output of the hidden layers of the SAE, which perform successive dimensionality reduction on the input vector, is used for adversarial example detection. Dimensionality reduction preserves the critical features that affect the deep learning algorithm's output. As a result, anomalies become more pronounced in the reduced dimensional space of the hidden layers. Our proposed method aims to exploit this feature to detect adversarial examples. We perform a clustering method on the output of the hidden layer, specifically, k-means clustering. During the training stage with normal examples, the hidden layer outputs are clustered and the respective cluster centers are noted. The distance from each cluster center for each example is determined and the maximum distance is noted. In the test stage, the hidden layer outputs for the incoming examples are obtained and the distances to the cluster centers are calculated. If the distances from the respective cluster centers are greater than the maximum distances obtained in the training stage, then the incoming network traffic is designated as malicious.

Algorithm 1 SAE-Based Detection and Perturbation Process

```
1: Input: $x, l, \epsilon, M, T$
2: Output: Feature $q$

3: Training Phase:
4: for each $x_i \in X$ do
5:     $s \leftarrow \text{RMSE}(\vec{x}_i, h_\theta(\vec{x}_i))$
6:     if $s \geq \Phi$ then
7:         $\Phi \leftarrow s$
8:     end if
9:     Update $\theta$ with $x_i$
10: end for

11: Execution Phase:
12: $s \leftarrow \text{RMSE}(\vec{x}', h_\theta(\vec{x}'))$
13: if $s \geq \Phi$ then
14:     Raise Alert
15: end if

16: Perturbation Loop:
17: for $i \leftarrow 0$ to $M$ do
18:     if $l = 1$ and RMSE $< T$ then
19:         return $q$
20:     else if $l = 0$ and RMSE $> T$ then
21:         return $q$
22:     end if
23:     $q \leftarrow q - \epsilon \cdot \text{sign}\left(\frac{\partial(-1)^{i+1} \cdot \text{RMSE}}{\partial q}\right)$
24: end for
```

In general, the k-means algorithm divides a set of n samples of the data X into K disjoint clusters C, each described by the mean μ_j of the samples in the cluster. The within-cluster sum-of-squares criterion is provided by

$$\sum_{i=0}^{n} \min_{\mu_j}\left(\|x_i - \mu_j\|^2\right). \tag{8}$$

During the training stage, the clusters represent the region of the hidden layer space in which normal examples reside. A distance measure is needed to calculate the distance between the example points and the cluster centers. In this regard, we use the cosine distance between two vectors as our distance measure. The cosine distance between two 1D vectors u and v is provided by

$$1 - \frac{u \cdot v}{\|u\|_2 \cdot \|v\|_2}. \tag{9}$$

We obtain the cosine distances of each example point from their respective cluster centers. The observation is labeled as adversarial in the test stage if the cosine distances for the adversarial examples are more significant than the maximum cosine distances obtained during training.

4.3. Robustness Analysis

We assume a white-box attack model for our proposed early detection approach; as such, the attacker is cognizant of all the details of the deep learning algorithm architecture implemented and the input data. To defeat the adversarial example detection method, an attacker can corrupt the output of the hidden layers by adding log-normal Gaussian noise. To evaluate the robustness of our detection method, we introduced log-normal Gaussian noise into the hidden layers to mimic the actions of a knowledgeable attacker attempting to evade detection. This noise is not random but rather carefully designed to interfere with the feature patterns that our method relies on. As the intensity of the noise increases, the ability of the detector to distinguish adversarial examples declines noticeably. This experiment highlights a potential evasion pathway and underscores the importance of designing defenses that remain reliable under more adaptive adversarial conditions.

In the context of IoT networks, minimizing false positives is particularly important, as even small misclassifications can trigger unnecessary alerts, disrupt normal device operations, or consume valuable system resources. The balance between detection sensitivity and system stability is critical in these environments, and any intrusion detection solution must ensure that benign traffic is not frequently misclassified as malicious. In our approach, the detection threshold is carefully selected to achieve high true positive rates while maintaining reliable discrimination between benign and adversarial samples. By leveraging intrinsic model behavior through hidden-layer representations, the proposed method allows for early identification of abnormal patterns without excessive reliance on labeled data. This design helps to preserve operational continuity in IoT systems while providing timely alerts against sophisticated adversarial intrusions. Because it relies on hidden-layer features already computed during forward propagation, our method introduces minimal computational overhead, making it suitable for low-power IoT deployments.

5. Experimental Results

We tested our approach on one of the Kitsune datasets, namely, ARP MitM. For the Kitsune datasets, statistical features of network traffic are obtained from a feature extractor module and then fed to the NIDS. In the case of the ARP MiTM dataset, a man-in-the-middle (MitM) attack is performed in which an attacker intercepts all LAN traffic through an ARP poisoning attack [34]. This MitM attack is categorized as violating the confidentiality of the network. Each observation in the dataset is a 100-dimensional vector. The deep learning algorithm and detection method were implemented in TensorFlow 2.8.0.

To implement our proposed adversarial example detection approach, we first generate adversarial examples that lead to the misclassification of normal traffic as malicious by the NIDS. We then integrate the SAE architecture into the NIDS and train it using the first 100,000 observations to establish a threshold. The threshold provides a measure of the tradeoff between various performance metrics, such as the true positive rate (TPR), false positive rate (FPR), false negative rate (FNR), and ROC-AUC score. In the test stage, we observed the performance metrics on the remaining observations. The performance metrics were obtained for different threshold values versus the ratio change, as shown in Figure 5. A suitable threshold value is selected from Figure 5, which provides a reasonably good tradeoff between the TPR and FNR. We chose a threshold value of 1.4, which provided a TPR of 73.88%, FNR of 26.12%, and ROC-AUC of 0.841.

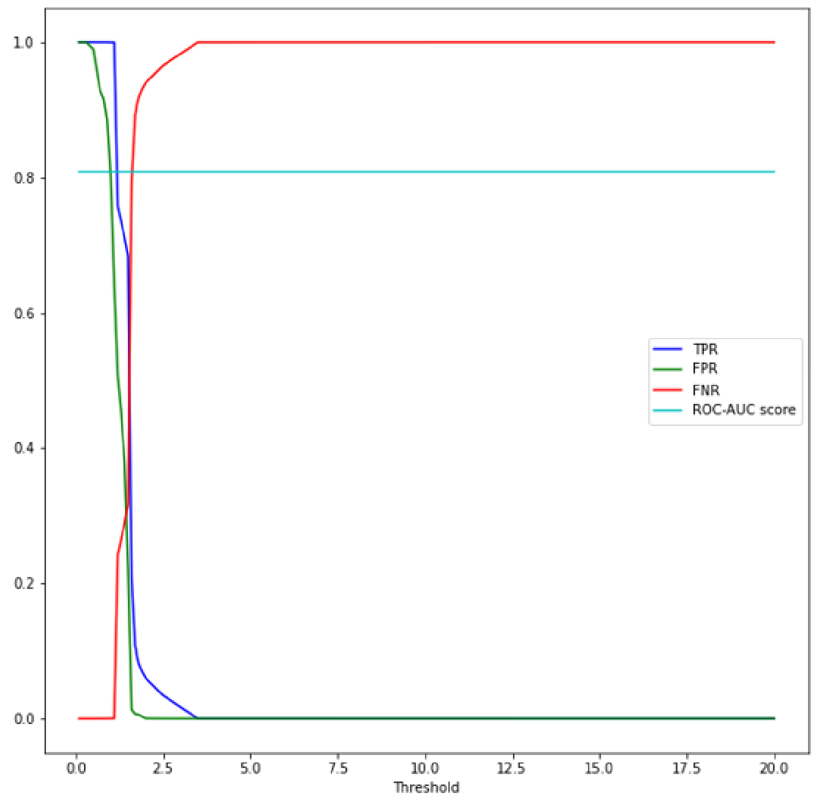

Figure 5. ARP MitM dataset performance metrics.

The first step in the adversarial example generation process is the identification of the critical features which affect the label assigned to the traffic. As we generated adversarial examples that cause benign samples to be labeled as malicious, we generated the saliency map for eleven randomly selected benign samples, identifying features that give a higher probability for the malicious class, as shown in Figure 6. Figure 6 shows that the 94th feature (index = 93) greatly affects the malicious label; in other words, adversarial examples can be generated with minimal perturbation of the 94th feature. As shown in Figure 3, certain features play a crucial role in shifting data points from one decision boundary to another. These features are identified as critical. In this study, we utilize the FGSM to introduce perturbations, modifying the features in a way that causes the data point to be classified as adversarial.

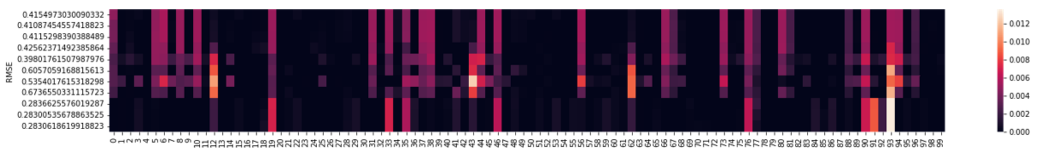

Figure 6. Saliency map for the eleven randomly chosen benign samples in the ARP MitM dataset

We generated the adversarial examples using the iterative FGSM algorithm provided in Algorithm 1. The maximum number of iterations M was chosen to be 1000, while the perturbation ϵ was selected as 1. We observed that the average time for generating a single adversarial example is 0.337 s and that the time taken was much smaller than the packet arrival rate at the NIDS; hence, the generation of adversarial examples does not impose a severe time constraint on the normal operation of the NIDS.

After the adversarial example generation stage, we attempted our early detection approach. During the training stage, k-means clustering was applied to the outputs from each hidden layer in the encoder in Figure 4. The outputs from these hidden layers were grouped into two clusters and the cluster centers were determined. The cosine distance was used to determine the distances between the cluster centers and each of the reduced dimensional vectors from the hidden layers. The maximum cosine distances from each cluster center were obtained and the remaining observations were fed to the trained SAE at the test stage. The cosine distance for each observation in every hidden layer of the encoder was computed, with an observation is considered adversarial if its cosine distance exceeded the maximum distance from any cluster center.

From Table 2, it can be observed that the adversarial examples are identified upon traversing the hidden layers. The dimensionality reduction performed in the hidden layers of the SAE preserves the critical features that the SAE needs to reconstruct the output at the decoder side. An adversarial example lies far away from the regions of normal traffic; as it is generated by perturbation of the critical features, an adversarial example is revealed after projection into a lower dimension. This is because dimensionality reduction preserves the critical features with the greatest impact on the output of the SAE. Because the detection rate is nearly 100% for all the hidden layers, we consider only the output from the first hidden layer in our early detection approach. The reason for this is that upon detection of adversarial examples, the error can be mitigated without requiring backpropagation through multiple layers if the detection is performed at the first hidden layer instead of traversing into deeper layers. Hence, the proposed approach can reduce the computational cost of mitigating the effect of adversarial examples in the layers of the SAE architecture.

Table 2. Adversarial detection rates for different hidden layers.

Dataset	1st Hidden Layer	2nd Hidden Layer	3rd Hidden Layer
ARP MitM	1	0.9856	0.9722

In our final analysis, we tested the robustness of our proposed approach. As we assume a white-box attack model, we assume the attacker has information about the details of the implemented SAE architecture and the input data from the NIDS. Because the attacker can access even the hidden layers of the model, a simple countermeasure against adversarial example detection is to add zero-mean and log-normal Gaussian noise to the output of the hidden layers. Table 3 shows the reduction in the adversarial example detection rate for log-normal noise with varying degrees of standard deviation added to the output of the first hidden layer. At runtime, the model efficiently detects adversarial examples in real time. Optimizing the model architecture by reducing its depth and utilizing smaller layer sizes allows it to perform encoding and decoding operations more effectively.

Table 3. Reduction in adversarial detection rate for different magnitudes of log-normal Gaussian noise.

Dataset	Standard Deviation = 0.2	Standard Deviation = 0.4	Standard Deviation = 0.6	Standard Deviation = 0.8	Standard Deviation = 1
ARP MitM	1	0.998	1	0.984	1

6. Conclusions

In conclusion, we have proposed an unsupervised adversarial example detection method for NIDS in IoT networks based on an early detection approach from the hidden layers in the implemented deep learning algorithm. Unsupervised deep learning algorithms are particularly useful for IoT networks due to the need for domain expertise and labeling in IoT network deployments. Our proposed method requires no additional hardware overhead for implementation, and the early detection approach does not require computationally intensive retraining of the deep learning algorithm implemented in the target NIDS. Our proposed method obtains an adversarial example detection rate of nearly 100% for almost all the hidden layers. We use autoencoders to generate adversarial examples for the deep learning model. After these adversarial examples are created, we identify them by leveraging the intrinsic properties of the model's layers.

Furthermore, we conducted a robustness analysis of our approach in which the attacker can easily bypass the detection mechanism using log-normal Gaussian noise of small magnitude. In future work, we intend to investigate ways of making the proposed detection approach robust against such bypass strategies as well as to deploy additional methods for generating adversarial examples and to implement our proposed detection approach in a real-time IoT network. While the detection accuracy of the proposed method is promising, we recognize that false positives (misclassifying benign traffic as malicious) can cause disruption in practical IoT environments. This concern highlights the importance of balancing sensitivity with operational usability, which we plan to address through adaptive thresholding and feedback-based calibration in future deployments.

While our approach focuses on detecting adversarial examples crafted via perturbations to input data, IoT networks are also vulnerable to a range of other attack vectors. Side-channel attacks such as those based on power consumption, RF leakage, or timing behavior pose serious threats to IoT systems with constrained hardware; similarly, physical adversarial attacks, including sensor manipulation and environmental interference, can introduce data patterns that challenge traditional detection schemes. Additionally, low-power wide-area network technologies such as LoRa introduce their own security challenges, including replay attacks and over-the-air manipulation. While these threats are beyond the scope of our current implementation, we recognize the importance of evaluating detection methods under such conditions and plan to explore these directions in future work.

Author Contributions: Methodology, W.D. and M.R.; Software, W.D.; Resources, W.D.; Writing—original draft, W.D.; Writing—review & editing, S.R.S.; Visualization, S.R.S.; Supervision, M.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Chen, Y.; Qiu, J.; Du, X.; Yin, L.; Tian, Z.
Leveraging Transferability and Improved Beam Search in Textual Adversarial Attacks. *Neurocomputing* **2022**, *500*, 135–142. [CrossRef]
2. Meola, A. A Look at Examples of IoT Devices and Their Business Applications in 2022. Available online: https://www.businessinsider.com/internet-of-things-devices-examples (accessed on 6 June 2025).
3. Qiu, J.; Chen, Y.; Tian, Z.; Guizani, N.; Du, X. The Security of Internet of Vehicles Network: Adversarial Examples for Trajectory Mode Detection. *IEEE Netw.* **2021**, *35*, 279–283. [CrossRef]
4. Chen, Y.; Qiu, J.; Du, X.; Yin, L.; Tian, Z. Security of Mobile Multimedia Data: Adversarial Examples for Spatio-Temporal Data. *Comput. Netw.* **2020**, *181*, 107432. [CrossRef]
5. Sun, Z.; Ni, T.; Yang, H.; Liu, K.; Zhang, Y.; Gu, T.; Xu, W. FLoRa: Energy-Efficient, Reliable, and Beamforming-Assisted Over-The-Air Firmware Update in LoRa Networks. In Proceedings of the 22nd International Conference on Information Processing in Sensor Networks (IPSN '23), San Antonio, TX, USA, 9–12 May 2023; pp. 14–26. [CrossRef]
6. Albulayhi, K.S. IoT Intrusion Detection Taxonomy, Reference Architecture, and Analyses. *Sensors* **2021**, *21*, 6432. [CrossRef]
7. Asharf, J.M. A Review of Intrusion Detection Systems Using Machine and Deep Learning in Internet of Things: Challenges, Solutions and Future Directions. *Electronics* **2020**, *9*, 1177. [CrossRef]

8. Chaabouni, N.M. Network Intrusion Detection for IoT Security Based on Learning Techniques. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2671–2701. [CrossRef]
9. da Costa, K.A. Internet of Things: A Survey on Machine Learning-Based Intrusion Detection Approaches. *Comput. Netw.* **2019**, *151*, 147–157. [CrossRef]
10. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. A Critical Review of Intrusion Detection Systems in the Internet of Things. *Cybersecurity* **2021**, *4*, 1–27. [CrossRef]
11. Wang, X.; Dai, L.; Yang, G. A network intrusion detection system based on deep learning in the IoT. *J. Supercomput.* **2024**, *80*, 24520–24558. [CrossRef]
12. Shadroo, S.; Rahmani, A.M.; Rezaee, A. Survey on the application of deep learning in the Internet of Things. *Telecommun. Syst.* **2022**, *79*, 601–627. [CrossRef]
13. Thakkar, A.; Lohiya, R. A Review on Machine Learning and Deep Learning Perspectives of IDS for IoT. *Arch. Comput. Methods Eng.* **2021**, *28*, 3211–3243. [CrossRef]
14. Tsimenidis, S.; Lagkas, T.; Rantos, K. Deep Learning in IoT Intrusion Detection. *J. Netw. Syst. Manag.* **2022**, *30*, 8. [CrossRef]
15. Alsoufi, M.A. Anomaly-Based Intrusion Detection Systems in IoT Using Deep Learning: A Systematic Literature Review. *Appl. Sci.* **2021**, *11*, 8383. [CrossRef]
16. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. *arXiv* **2014**, arXiv:1412.6572.
17. Chan, T.H. PCANet: A Simple Deep Learning Baseline for Image Classification? *IEEE Trans. Image Process.* **2015**, *24*, 5017–5032. [CrossRef]
18. Zhang, W.E.; Sheng, Q.Z.; Alhazmi, A.; Li, C. Adversarial Attacks on Deep-Learning Models in NLP: A Survey. *ACM Trans. Intell. Syst. Technol.* **2020**, *11*, 24.
19. Carlini, N.; Wagner, D. Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. In Proceedings of the 2018 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 24 May 2018; pp. 1–7.
20. Sun, J.Z.; Wang, Y.; Liu, Y. Stealthy and Efficient Adversarial Attacks Against Deep Reinforcement Learning. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 5883–5891. [CrossRef]
21. Qiu, H.D.; Du, M.; Zhang, Y.; Li, X. Adversarial Attacks Against Network Intrusion Detection in IoT Systems. *IEEE Internet Things J.* **2020**, *8*, 10327–10335. [CrossRef]
22. Bulusu, S.K. Anomalous Example Detection in Deep Learning: A Survey. *IEEE Access* **2020**, *8*, 132330–132347. [CrossRef]
23. Jiang, H.; Lin, J.; Kang, H. FGMD: A Robust Detector Against Adversarial Attacks in the IoT Network. *Future Generation Computer Systems* **2022**, *132*, 194–210. [CrossRef]
24. Vitorino, J.O.; Oliveira, N.; Praça, I. Adaptative perturbation patterns: Realistic adversarial learning for robust intrusion detection. *Future Internet* **2022**, *14*, 108. [CrossRef]
25. Wang, J.; AlQerm, J.P.I.; Liu, Y. Def-IDS: An Ensemble Defense Mechanism Against Adversarial Attacks for Deep Learning-based Network Intrusion Detection. In Proceedings of the 2021 International Conference on Computer Communications and Networks, Athens, Greece, 19–22 July 2021; pp. 1–9.
26. Ibitoye, O.S.; Shokri, R. Differentially Private Self-Normalizing Neural Networks for Adversarial Robustness in Federated Learning. *Comput. Secur.* **2022**, *116*, 102631. [CrossRef]
27. Anthi, E.W. Hardening Machine Learning Denial of Service (Denial-of-Service (DoS)) Defences Against Adversarial Attacks in IoT Smart Home Networks. *Comput. Secur.* **2021**, *108*, 102352. [CrossRef]
28. Fu, X.Z. The Robust Deep Learning–Based Schemes for Intrusion Detection in Internet of Things Environments. *Ann. Telecommun.* **2021**, *76*, 273–285. [CrossRef]
29. Qiu, J.; Du, L.; Chen, Y.; Tian, Z.; Du, X.; Guizani, M. AI Security in 5G Networks: Adversarial Examples for Travel Time Estimation. *IEEE Veh. Technol. Mag.* **2020**, *15*, 95–100. [CrossRef]
30. Sun, Z.; Yin, L.; Li, C.; Zhang, W.; Li, A.; Tian, Z. QoS and Privacy Trade-Off in Adversarial Deep Learning: An Evolutionary Game Approach. *Comput. Secur.* **2020**, *96*, 101876. [CrossRef]
31. Li, J.; Zhou, H.; Wu, S.; Luo, X.; Wang, T.; Zhan, X.; Ma, X. FOAP: Fine-Grained Open-World Android App Fingerprinting. In Proceedings of the 31st USENIX Security Symposium (USENIX Security 22), Boston, MA, USA, 10–12 August 2022; pp. 1579–1596. Available online: https://www.usenix.org/conference/usenixsecurity22/presentation/li-jianfeng (accessed on 26 July 2025).
32. Ni, T.; Lan, G.; Wang, J.; Zhao, Q.; Xu, W. Eavesdropping Mobile App Activity via Radio-Frequency Energy Harvesting. In Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23), Anaheim, CA, USA, 9–11 August 2023; Article No. 197, pp. 3511–3528.
33. Yuan, S.; Li, H.; Han, X.; Xu, G.; Jiang, W.; Ni, T.; Zhao, Q.; Fang, Y. ITPatch: An Invisible and Triggered Physical Adversarial Patch against Traffic Sign Recognition. *arXiv* **2024**, arXiv:2409.12394. Available online: https://arxiv.org/abs/2409.12394 (accessed on 26 July 2025).
34. Mirsky, Y.; Doitshman, T.; Elovici, Y.; Shabtai, A. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. *arXiv* **2018**, arXiv:1802.09089. [CrossRef]

35. Yousefi-Azar, M.; Varadharajan, V.; Hamey, L.; Tupakula, U. Autoencoder-Based Feature Learning for Cyber Security Applications. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 3854–3861. [CrossRef]
36. Huang, X.; Shen, C.; Boix, X.; Zhao, Q. Salicon: Reducing the Semantic Gap in Saliency Prediction by Adapting Deep Neural Networks. In Proceedings of the 2015 IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 262–270.