electronics

MDPI

Article

Efficient Real-Time Anomaly Detection in IoT Networks Using One-Class Autoencoder and Deep Neural Network

Aya G. Ayad [1,*], Mostafa M. El-Gayar [1,2,*], Noha A. Hikal [1] and Nehal A. Sakr [1]

[1] Information Technology Department, Faculty of Computers and Information, Mansoura University, Mansoura 35516, Egypt; dr_nahikal@mans.edu.eg (N.A.H.); nehal_sakr@mans.edu.eg (N.A.S.)
[2] Department of Computer Science, Arab East Colleges, Riyadh 11583, Saudi Arabia
* Correspondence: ayaayad1011@mans.edu.eg (A.G.A.); mostafa_elgayar@mans.edu.eg (M.M.E.-G.)

Abstract: In the face of growing Internet of Things (IoT) security challenges, traditional Intrusion Detection Systems (IDSs) fall short due to IoT devices' unique characteristics and constraints. This paper presents an effective, lightweight detection model that strengthens IoT security by addressing the high dimensionality of IoT data. This model merges an asymmetric stacked autoencoder with a Deep Neural Network (DNN), applying one-class learning. It achieves a high detection rate with minimal false positives in a short time. Compared with state-of-the-art approaches based on the BoT-IoT dataset, it shows a higher detection rate of up to 96.27% in 0.27 s. Also, the model achieves an accuracy of 99.99%, precision of 99.21%, and f1 score of 97.69%. These results demonstrate the effectiveness and significance of the proposed model, confirming its potential for reliable deployment in real IoT security problems.

Keywords: asymmetric stacked autoencoder; deep neural network; dimensionality reduction; internet of things; intrusion detection system; one-class classifier

Academic Editor: Aryya Gangopadhyay

Received: 8 November 2024
Revised: 19 December 2024
Accepted: 27 December 2024
Published: 30 December 2024

Citation: Ayad, A.G.; El-Gayar, M.M.; Hikal, N.A.; Sakr, N.A. Efficient Real-Time Anomaly Detection in IoT Networks Using One-Class Autoencoder and Deep Neural Network. *Electronics* **2025**, *14*, 104. https://doi.org/10.3390/electronics14010104

1. Introduction

The Internet of Things (IoT) is a network of physical items, devices, automobiles, and appliances with sensors, software, and connections. It transforms our interaction with technology by facilitating data collection and exchange [1]. The ease of use of IoT is one of its essential characteristics that gives customers the ability to remotely manage several areas of their lives, including energy management [2], health monitoring, and home lighting. Furthermore, IoT facilitates the automation of routine tasks, enhancing efficiency and productivity [3].

Although this network has numerous significant features [4,5], it also has limitations and security issues that should be addressed [6]. IoT devices are vulnerable to cyberattacks because of their high level of connectivity, which may risk critical data and result in physical harm. It is crucial to address these security concerns [7].

An Intrusion Detection System (IDS) is crucial for mitigating the security threats associated with IoT devices [8]. This system actively tracks network traffic [9], looking for unusual patterns or actions that could point to a potential cyberattack [10]. Additionally, it protects user privacy and safety and ensures the IoT network's confidentiality, integrity, and availability [11]. It detects the attack by analyzing the data gathered from several connected devices, after which it alerts the administrator with reports.

There are two primary types of IDS commonly employed in IoT environments: Signature-IDS (SIDS) and Anomaly-IDS (AIDS) [12]. The main difference between them is their approach to detecting malicious activities [13]:

- SIDS, known as pattern-matching or misuse IDS, compares system events or network traffic against an attack signatures database. These signatures may be specific behaviors, patterns, or characteristics associated with known attacks [14]. If a match is found, the SIDS triggers an alert. SIDS effectively detects known attacks but may fail against new attacks [15].
- AIDS analyzes network traffic or system behaviors to identify patterns that significantly vary from the expected normal behavior [16]. AIDS compares real-time traffic to a baseline of normal behavior established using machine learning or deep learning algorithms [15]. AIDS may either raise an alert or take action to mitigate the possible threat if abnormalities are discovered. Its goal is to detect and address potential security risks or attacks not covered by traditional SIDS [17].

AIDS is particularly suitable for IoT environments due to the dynamic and varied nature of IoT devices [18]. Its ability to adapt to evolving IoT device patterns enhances its effectiveness in identifying emerging threats. Given these advantages, this paper focuses on AIDS as a key approach for securing IoT networks. However, despite its strengths, AIDS faces significant challenges in IoT environments [19]. The sheer size of IoT networks creates tremendous difficulties for AIDS. As the number of connected devices grows, AIDS may fail to manage the enormous volume of data, leading to false positives or missed alarms [20]. Additionally, the resource constraints of IoT devices—such as limited processing power, memory, and bandwidth—further hinder their ability to operate effectively [21]. Furthermore, the heterogeneous nature of IoT devices makes developing a unified detection model challenging, restricting its capacity to adapt to the diverse types and behaviors of IoT devices.

Deep Neural Network (DNN) models show promise in addressing some of these limitations, notably in capturing complex relationships in high-dimensional data. However, typical DNN techniques have difficulties in IoT scenarios. They frequently require extensive labeled data for training, which may not always be accessible. They may also face challenges with unbalanced datasets in which majority-class traffic dominates minority-class traffic. Moreover, the computing demands of deep learning models may not be compatible with IoT device resources.

To address these challenges, this work introduces a novel AIDS model that employs an asymmetric stacked autoencoder for dimensionality reduction to enhance a DNN for anomaly detection. One-class learning allows the model to focus on learning normal behavior patterns and identify deviations without relying heavily on labeled attack samples. The proposed approach improves the detection rate by effectively handling high-dimensional data, addressing class imbalance, and overcoming computational constraints, all while maintaining efficiency in resource-constrained IoT environments.

The primary contributions of the proposed work are as follows:

- Proposes a hybrid model that combines asymmetric stacked autoencoder for dimensionality reduction with a one-class Deep Neural Network for precise anomaly detection.
- Introduces a data-driven Thresholding mechanism based on precision–recall curves to enhance adaptability and detection efficiency.
- Reduces detection times that enhance real-time performance while maintaining a consistently high detection rate for reliable anomaly detection.

The structure of this paper is as follows: Section 2 reviews recent related works. Section 3 provides a detailed explanation of the proposed model. Section 4 presents the experimental setup and results. Section 5 discusses the findings, while Section 6 concludes the paper and suggests directions for future research.

2. Related Work

This section reviews the literature on anomaly detection and IDS algorithms, evaluating their strengths and weaknesses and highlighting gaps that our proposed approach aims to address. Table 1 summarizes the key findings.

Table 1. Summary of Related Work. Results include Accuracy (%), Precision (%), Recall (%), f1 score (%), Detection Rate (%), and Detection Time (seconds).

Study	Method	Dataset(s)	Results
Yang et al. [22]	ASE + CNN	CTU-UNB, Contagio-CTU-UNB	CTU-UNB: f1 score = 99.00 (6 classes), 98.00 (8 classes); Contagio-CTU-UNB: f1 score = 98.00 (6 classes), 97.00 (6 classes)
Aygun and Yavuz [23]	AE, DnAE	NSL-KDD	AE: Accuracy = 88.28; DnAE: Accuracy = 88.65
Zhang et al. [24]	DnAE + WL	UNSW-NB15	Accuracy = 98.80; Precision = 95.00; Recall = 95.00
Mao et al. [25]	GAN + AE	KDDCUP99, Arrhythmia, MNIST, CIFAR-10	KDDCUP99: f1 score = 87.00
Ieracitano et al. [26]	SA, SAE, Q-SVM	NSL-KDD	Q-SVM: Accuracy = 83.15 (binary), 83.65 (multi); SAE: f1 score = 87.10 (Normal), 97.08 (DoS), 77.13 (Probe)
Hikal and Elgayar [9]	SVM, BPNN, RF, J-48	Standard	Accuracy = 99.7, detection time = 30:80
Binbusayyis et al. [27]	1D CAE + OCSVM	UNSW-NB15	Detection Rate = 97.80; Accuracy = 97.60; f1 score = 98.10
Adeniyi et al. [28]	DFFNN + SAE	NF-ToN-IoT	Accuracy = 89.00
Yao et al. [29]	OC-BiGRU-AE + Ens	WSN-DS, UNSW-NB15, KDDCUP99	Detection Rates = 97.91 (WSN-DS), 98.92 (UNSW-NB15), 98.23 (KDDCUP99)
Hou et al. [30]	NAE + DNN	NSL-KDD, N-BaIoT, BoT-IoT	Accuracy = 90.03 (NSL-KDD), 99.51 (N-BaIoT), 99.80 (BoT-IoT)
El-Gayar et al. [20]	Stacking	CICIDS2017, car-hacking	Accuracy = 98
Proposed	**One-Class Asymmetric Stacked Autoencoder + One-Class DNN**	**BoT-IoT**	**Detection Rate = 96.27; Accuracy = 99.99; Precision = 99.21; f1 score = 97.69; Detection Time = 0.27**

Bold text indicates the proposed work.

2.1. Overview of Existing Methods

Yang et al. [22] suggested utilizing a stacked autoencoder (SAE) and Convolutional Neural Network (CNN) for enhanced intrusion detection. The model performed well with f1 scores of 99.00% and 98.00% for 6- and 8-class classification tasks on the CTU-UNB dataset, and similarly high scores on the Contagio-CTU-UNB dataset. They used resource-intensive data conversion to fit the CNN architecture, which made real-time applications difficult.

Considering resource constraints, Aygun and Yavuz [23] compared traditional and denoising autoencoders for anomaly detection. Despite 88.28% and 88.65% accuracies, their results are inferior to modern benchmarks. According to Zhang et al. [24], integrating DnAE with a weighted loss function improved feature selection accuracy to 98.80% on the UNSW-NB15 dataset, with precision and recall scores around 95.00%.

In a novel approach, Mao et al. [25] combined a discriminative encoder and generator in a Generative Adversarial Network (GAN) for training and used the encoder as an Autoencoder during testing On the KDDCUP99 dataset, this architecture scored 87.00% f1 score. However, using the outdated KDDCUP99 dataset limited its relevance for modern IoT-based IDS. Using statistical analysis (SA) and SAE architectures, Ieracitano et al. [26]

achieved competitive results on the NSL-KDD dataset. The Q-SVM classifier had 83.15% and 83.65% accuracies for binary and multiclass classification, while the SAE excelled with f1 scores for specific attack types. The performance was dataset-dependent and did not address modern IoT challenges.

Hikal and Elgayar [9] suggested a lightweight IoT botnet detection IDS using anomalies. The IDS detects botnet attacks using SVM, BPNN, RF, and Decision Tree models. The proposed framework achieves 99.7% detection accuracy and 30–80s detection time on a standard dataset, according to experiments. Ensemble preprocessing improves learner performance, detecting botnet attacks in IoT networks.

In a joint optimization framework, Binbusayyis et al. [27] developed an advanced unsupervised intrusion detection method utilizing a 1D Convolutional AE (CAE) and One-Class SVM (OCSVM). Although they achieved high detection rates of 97.8% and f1 scores of 98.1% on the UNSW-NB15 dataset, their joint optimization approach increased computational complexity.

Adeniyi et al. [28] used DFFNN and SAE to improve deep learning model fine-tuning for intrusion detection on the NF-ToN-IoT dataset. Although they achieved 89.00% accuracy, optimizing the autoencoder's learning process for feature extraction was challenging. Yao et al. [29] suggested combining ensemble learning with OC-Bi-GRU-AE for hybrid solutions. The model achieved high detection rates of 97.91%, 98.92%, and 98.23% on the WSN-DS, UNSW-NB15, and KDD-CUP99 datasets. Ensemble learning was resource-intensive, limiting its scalability.

Hou et al. [30] developed a hybrid model combining a CNN-based one-class nonsymmetric autoencoder and a DNN to improve performance. With accuracies of 90.03%, 99.51%, and 99.80% on various datasets, their approach showed strong detection capabilities. Fining decision boundaries and handling resource-intensive data conversion processes, especially in convolutional layers, remained difficult.

El-Gayar et al. [20] introduced DFSENet, a new IDS, to protect the Internet of Vehicles (IoVs) from cyberattacks. DFSENet detects cyber threats accurately by stacking multiple machine learning models sequentially. Experiments using CICIDS2017 and car-hacking datasets yielded over 98% accuracy.

2.2. Limitations of Existing Methods

Prior research on anomaly detection in IoT networks has several limitations that hinder their effectiveness in addressing real-world challenges. For instance, methods such as those proposed by Yang et al. [22] and Hou et al. [30] employed autoencoders combined with CNN, which require resource-intensive data conversion. This process is computationally expensive and less suitable for IoT datasets that predominantly consist of numerical data. Another critical challenge is the issue of class imbalance, which is prevalent in IoT datasets. While some studies, such as Binbusayyis et al. [27], attempted to address this, the solutions remain insufficient to handle diverse IoT traffic patterns. Furthermore, techniques like the one proposed by Hou et al. [30] lack data-driven thresholding mechanisms, limiting their adaptability to varying attack scenarios. Lastly, many studies, including Mao et al. [25], relied on outdated datasets like NSL-KDD, which fail to represent the complexity and diversity of modern IoT environments, limiting their findings' generalizability.

2.3. Addressing Existing Challenges

Our approach integrates several key innovations to overcome the challenges identified in existing methods. We combine an autoencoder with a DNN optimized for the numerical nature of IoT datasets, reducing computational overhead while maintaining high detection accuracy. This architecture eliminates the need for computationally expensive data

conversion, commonly seen in CNN-based approaches. We employ a one-class learning technique, enabling the model to focus on learning normal behavior patterns and better handling class imbalance. This improves detection performance, particularly in scenarios with uneven class distributions. Additionally, we implement a data-driven thresholding technique based on precision–recall curves, offering greater adaptability and accuracy in anomaly detection. This method allows the model to adjust detection thresholds effectively. Finally, our model is trained on a modern, comprehensive dataset that includes various attack types, enhancing its generalizability and real-world applicability. These innovations contribute to a robust, high-performance solution for anomaly detection in IoT networks.

3. The Proposed Model

IoT networks are being challenged by the need to manage massive volumes of high-dimensional data, which may consume resources and hinder rapid anomaly detection. Dimensionality reduction methods are critical for reducing this cost but must be carefully constructed to ensure efficiency without decreasing detection accuracy. We suggest combining a one-class asymmetric stacking autoencoder for dimensionality reduction with a one-class DNN for anomaly detection. This approach alleviates computational challenges and enhances the precision of anomaly detection. The proposed model is visually summarized in Figure 1.

The proposed model processes network traffic in three main stages. Initially, the data are preprocessed to ensure they are suitable for the model. Next, dimensionality reduction is performed using an unsupervised asymmetric stacked autoencoder, which is trained exclusively on normal traffic data to extract significant features. These features are then passed to the third component, a DNN, which is also trained only on normal traffic. The DNN detects anomalous network activity by comparing the input features against a predefined threshold. If the detection probability exceeds the threshold, the traffic is classified as anomalous; otherwise, it is considered normal. This targeted training strategy enables the DNN to identify deviations from typical behavior effectively. The subsequent subsections provide a comprehensive discussion of the proposed model.

3.1. Data Preprocessing

IoT networks produce large-scale, heterogeneous datasets comprising diverse feature types, including numerical data such as packet sizes and categorical data like protocol types. These features often have varying value ranges, which can affect the performance of machine learning models. Therefore, effective preprocessing is essential to standardize the data and make them suitable for model input. In this study, we use the Bot-IoT dataset, which is well organized, free of null values, and does not contain redundant samples. The preprocessing stage focuses on two primary objectives: encoding categorical data into a format that machine learning models can process and normalizing numerical features to ensure consistent value ranges.

3.1.1. Data Encoding

The label encoding was applied to convert categorical features, such as the state attribute with values like ‘RST’, ‘CON’, and ‘REQ’ into numerical values such as 1, 2, and 3. This technique guarantees compliance with the model [31], which is especially advantageous for resource-constrained IoT devices. Label encoding simplifies representation, saves memory, and aligns categorical data with machine learning model needs by providing a unique integer value to each category within a feature. It translates a categorical feature $(F_1, F_2, \cdots, F_n)$ with C distinct values to an integer $(F_1\_enc, F_2\_enc, \cdots, F_n\_enc)$ in the range of 0 to $C-1$, where C is the feature length, as described in Algorithm 1.

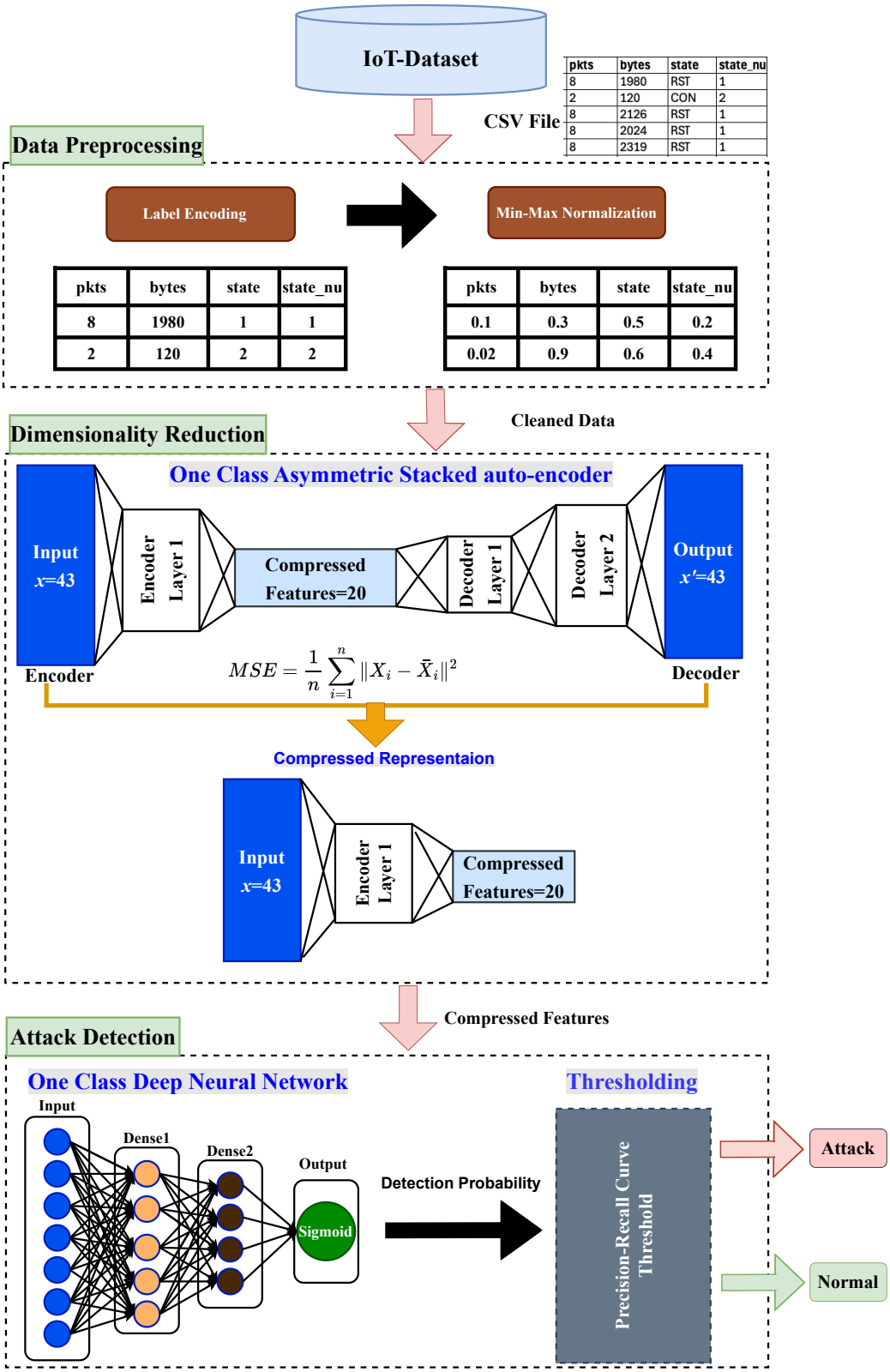

Figure 1. Framework of the proposed model.

Algorithm 1 Label Encoding

Input: Categorical feature F
Output: Numerical feature F_enc
$unique\_labels \leftarrow []$
$label\_map \leftarrow dict\{\}$
for each $value$ in F **do**
  if $value$ not in $unique\_labels$ **then**
    add $value$ to $unique\_labels$
    $label\_map[value] \leftarrow$ length($unique\_labels$) - 1
  end if
  $encoded\_value \leftarrow label\_map[value]$
  replace $value$ with $encoded\_value$ in F
end for
Return F_en

3.1.2. Normalization

Feature normalization is critical, specifically when dealing with datasets that have features with significantly different ranges. This step mitigates the impact of features with large values, which can negatively affect the performance of detectors, such as the 'bytes' feature, which contains values ranging from 120 to 1980. Research suggests that scaling improves the training process by ensuring consistent contributions from all features [32]. We utilize the Min-Max scaling method, which rescales all feature values to the range of [0, 1], as illustrated in Equation (1) [33] and Algorithm 2.

$$x = \frac{F\_enc - F\_enc_{min}}{F\_enc_{max} - F\_enc_{min}} \tag{1}$$

where x is the normalized value, F_enc is an original feature value, and F_enc_{max} and F_enc_{min} are the maximum and minimum values of this feature.

Algorithm 2 Min-Max Normalization

Require: $data$: The dataset to be normalized
Ensure: $normalizedData$: The normalized dataset
$F\_enc_{min} \leftarrow \min(data)$
$F\_enc_{max} \leftarrow \max(data)$
for each $value$ in $data$ **do**
  Compute x from Equation (1)
  $normalizedData$.addItem(x)
end for
return $normalizedData$

3.2. Dimensionality Reduction

This section focuses on the essential dimensionality reduction process, which improves model efficiency and accuracy, especially in high-dimensional IoT datasets. We employ an asymmetric stacked autoencoder (ASAE), an advanced autoencoder established specifically for dimensionality reduction applications, which captures correlations and preserves essential information through its reconstruction capabilities. Autoencoder, a kind of Artificial Neural Network (ANN), is generally employed for unsupervised learning tasks [34]. We utilize the autoencoder rather than traditional approaches, such as Principal Component Analysis (PCA) [35], since it can capture nonlinear correlations within the data. This selection makes the autoencoder a more flexible and adaptable technique than the linear transformations offered by PCA [36]. While alternative nonlinear methods like t-SNE, UMAP, and ISOMAP are also effective at capturing nonlinear relationships, they are primarily designed for visualization and do not offer the same reconstruction capabilities.

This makes ASAE more suitable for our task of preserving critical information for anomaly detection and classification. Additionally, ASAE's scalability, integration with machine learning pipelines, and adaptability to IoT data make it an ideal choice for our study. In the following sections, we briefly introduce the Traditional autoencoder, followed by a detailed explanation of ASAE.

3.2.1. Traditional Autoencoder

Traditional autoencoders usually consist of one layer in both the encoder and decoder, with symmetric architecture in which the number of neurons in the encoder matches the number of neurons in the decoder [37], as shown in Figure 2.

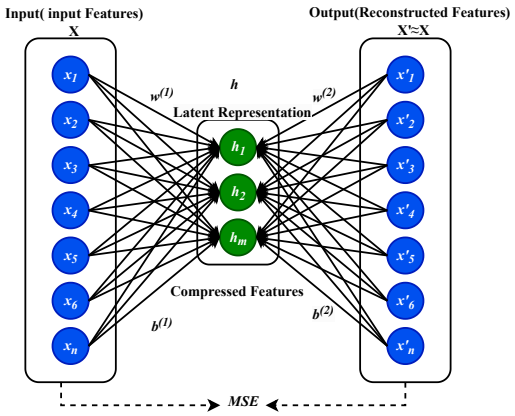

Figure 2. Structure of autoencoder.

3.2.2. Asymmetric Stacked Autoencoder

The ASAE differs from traditional autoencoders, which are typically shallow and symmetric. The ASAE employs a deep-layered and asymmetric structure, as depicted in Figure 3. This means the ASAE incorporates multiple hidden layers in the encoder and decoder, allowing for more complex feature extraction. Additionally, the ASAE's architectural flexibility provides for varying the number of layers and neurons between the encoder and decoder, resulting in more meaningful data representations and improved reconstruction accuracy.

As illustrated in Figure 3, the encoder and decoder networks are considered the main working blocks of ASAE, as explained below. Given the input dataset after preprocessing $(x_1, x_2, x_3, \ldots, x_n)$, where x indicates the feature representation of the input data and $\bar{x}$ represents the reconstructed output data after decoding from the hidden layer.

- The Encoder Network
  It transforms the input data into a representation in lower-dimensional latent space. The encoder has fewer layers but more neurons per layer, focusing on dimensionality reduction and abstraction [38]. Equation (2) shows the mathematical formula for the encoding process.

$$h(x_i) = F(W^{(1)}x_i + b^{(1)}) \tag{2}$$

  where $W^{(1)}$ represents the weight matrix connecting the input layer to the hidden layer, while $b^{(1)}$ denotes the bias vector associated with the input layer. The activation function F, a ReLU, is applied across all encoder layers to incorporate nonlinearity and mitigate challenges such as vanishing gradients [39]. The ReLU activation function is mathematically expressed in Equation (3).

$$\text{ReLU}(z_i) = max(0, z_i) \tag{3}$$

where z_i represents the value of the *i-th* element in the input vector z.

- The Decoder Network

  It reconstructs the original data from the compressed representation. It typically has more layers with fewer neurons, focusing on accurately reconstructing the original input data [38]. The decoding process is shown in Equation (4).

$$\bar{x}_i = g(W^{(2)}h(x_i) + b^{(2)}) \tag{4}$$

where $W^{(2)}$ is the weight matrix between the hidden layer and the output layer, $b^{(2)}$ is the bias vector of the hidden layer, and g is the decoding activation function, sigmoid, that takes a vector of real numbers as input and transforms them into a vector of values ranging from 0 to 1 [39]. Equation (5) mathematically represents the sigmoid function.

$$\text{sigmoid}(z_i) = \frac{1}{1 + e^{-z_i}} \tag{5}$$

The parameter matrix of the autoencoder is optimized to minimize the reconstruction error, or Mean Square Error (*MSE*), as shown in Equation (6).

$$MSE = \frac{1}{n}\sum_{i=1}^{n} \|X_i - \bar{X}_i\|^2 \tag{6}$$

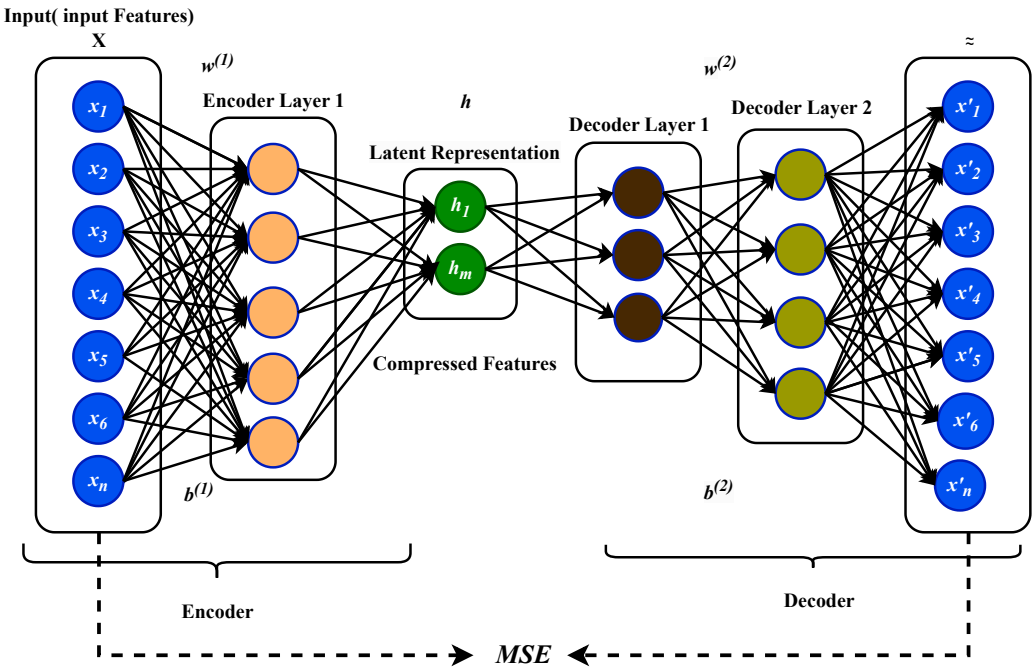

Figure 3. Structure of asymmetric stacked autoencoder.

3.3. Attack Detection

This section describes our framework's crucial attack detection stage, achieved through a fully connected DNN. However, unlike traditional learning methods, we employ a one-class (OC) learning paradigm. This approach focuses on learning the features of normal network traffic [40], allowing the DNN to effectively identify deviations from normal patterns that are classified as attacks [41].

During detection, the DNN utilizes the decision boundary to identify incoming data as normal or abnormal. If the data fall within the boundary, they are regarded as normal. Conversely, data points outside the boundary are considered attacks and trigger

appropriate responses from the IDS. This approach effectively addresses the challenge of imbalanced datasets.

3.3.1. Threshold Selection for OC-DNN

As previously discussed, the one-class (OC) model relies on a threshold or decision boundary during detection to differentiate between normal and abnormal behavior. This section examines two methods for determining the optimal threshold:

1. Receiver Operating Characteristic (ROC) Curve: This curve, typically used for balanced datasets [42], visually depicts the trade-off between correctly identifying normal (true positives) and mistakenly classifying attack traffic as normal (false positives) at various threshold settings [43]. The ROC curve is useful for balanced datasets as it illustrates the model's capability to differentiate between classes, taking into account both true and false positives. Since both classes are represented relatively, the ROC curve can accurately reflect the trade-offs between detecting positive instances and avoiding false alarms.
2. Precision–Recall Curve: This curve becomes particularly valuable for imbalanced datasets [42]. It graphically illustrates the precision (low false positive rate) against the recall (detection rate) [44]. By examining this curve, we can select a threshold that balances the required recall and precision values. Because it focuses on the model's performance about the positive class, the precision–recall curve provides more information for unbalanced datasets. Precision–recall curves do not consider the true negative rate, which might be excessively large in unbalanced datasets. As a result, even in rare cases, they are more applicable for assessing how well the model detects the positive class.

3.3.2. Data-Driven Threshold Determination (DDTD) for Anomaly Detection

Instead of being established randomly or predetermined, the threshold in the Data-Driven Threshold Determination (DDTD) method is decided based on the dataset's characteristics, such as the precision–recall trade-off. As seen in Figure 4 and explained in Algorithm 3, this method ensures that the threshold adjusts to the unbalanced nature of IoT network data using validation data. The technique minimizes false positive rates and creates a robust decision boundary by optimizing precision at intersection points while balancing recall and precision.

Algorithm 3 Data-Driven Threshold Determination using Precision–Recall Curve

Require: Trained model, $X_{\text{Validation}}$, y_{valid} (true labels of the validation set)
Ensure: Optimal threshold
Using the trained model, predict valid_y_predictions based on $X_{\text{Validation}}$
Compute the Precision–Recall Curve
for each data point in the validation set **do**
    precision, recall, thresholds ← Compute Precision–Recall($X_{\text{Validation}}$, y_{valid}, valid_y_predictions)
end for
intersection_indices ← Find Intersection Points(precision, recall)
intersection_values ← PrecisionAt(*intersection_indices*)
max_p ← max(intersection_values)
intersection_thresholds ← ThresholdsAt(*intersection_indices*)
optimal_threshold ← intersection_thresholds[IndexAt(max_p)]
return optimal_threshold

The presence of intersections between the precision and recall curves is guaranteed due to the imbalanced validation dataset, reflecting the coexistence of normal and abnormal traffic in real IoT environments. These intersections enable the model to determine a meaningful threshold during the validation phase, ensuring accurate classification of test data.

In contrast, scenarios lacking such intersections would occur only in one-class datasets (e.g., containing solely normal or anomalous data). However, these cases are not relevant to the proposed method or typical IoT environments, where both the validation and test datasets are designed to contain two classes, guaranteeing the existence of intersection points.

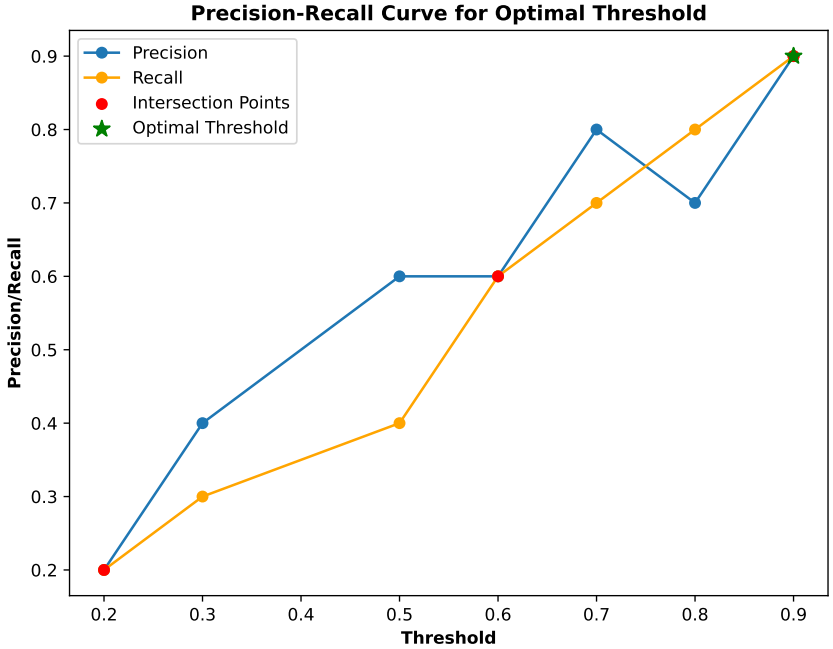

Figure 4. Determination of the optimal threshold based on the intersection points between precision and recall curve.

4. Experimental Evaluation and Results

This section provides an experimental evaluation of the proposed model's performance. It outlines evaluation metrics, experimental setup, and the used dataset for testing. Then, the results are compared against state-of-the-art intrusion detection methods to highlight the model's effectiveness.

4.1. Dataset Description

We used the BoT-IoT data to evaluate the proposed model. The BoT-IoT dataset, originating from the work by [45], was explicitly designed to reflect realistic network environments. It encompasses data collected from common smart home devices, including smart refrigerators, garage doors, thermostats, lights, and weather monitoring systems. Table 2 presents the characteristics of the BoT-IoT dataset, and Figure 5 illustrates a snapshot from the dataset file. Additionally, Table 3 presents the dataset's features from their name, description, domain, and data type.

Table 2. Characteristics of the BoT-IoT dataset.

Description	Value
Source	Cyber Range Lab at UNSW Canberra
Original_Size	72,000,000 samples
Utilized_ Size	3.6 million records (5% of the full dataset)
Class Distribution	477 normal flow samples, 3,688,045 attack flow samples
Features	46 per sample

daddr	dport	pkts	bytes	state	state_number	ltime	seq	dur	mean
192.168	80	8	1,980	RST	1	1,528,088,529	9	7.0564	0.0689
192.168	-1	2	120	CON	2	1,528,088,522	10	0.0001	0.0001
192.168	80	8	2,126	RST	1	1,528,088,529	11	7.0479	0.0645
192.168	80	8	2,024	RST	1	1,528,088,529	12	7.0476	0.0642
192.168	80	8	2,319	RST	1	1,528,088,529	13	7.0468	0.0639
192.168	80	8	1,983	RST	1	1,528,088,529	14	7.0466	0.0636
192.168	80	8	1,978	RST	1	1,528,088,529	15	7.0464	0.0631
192.168	80	8	2,021	RST	1	1,528,088,529	16	7.0461	0.063
192.168	80	8	1,996	RST	1	1,528,088,529	17	7.0458	0.0629
192.168	80	8	2,038	RST	1	1,528,088,529	18	7.0421	0.0611
192.168	80	8	2,300	RST	1	1,528,088,529	19	7.0413	0.0608

Figure 5. Example subset of the BoT-IoT dataset.

Table 3. Bot-IoT Dataset Features.

No.	Name	Description	Domain	Data Type
1	pkSeqID	Unique row identifier for each entry	Statistical	int64
2	stime	Timestamp marking the start of the record	Time	float64
3	flgs	Indicators for the transaction's flow state	Categorical	object
4	flgs_number	Numeric representation of the flow state flags	Statistical	int64
5	proto	Protocol name used during the transaction	Categorical	object
6	proto_number	Numeric encoding of the protocol used	Statistical	int64
7	saddr	IP address of the source initiating the transaction	Categorical	object
8	sport	Port number on the source side	Categorical	object
9	daddr	IP address of the destination endpoint	Categorical	object
10	dport	Port number on the destination side	Categorical	object
11	pkts	Total packet count in a transaction	Statistical	int64
12	bytes	Total byte count in a transaction	Statistical	int64
13	state	Status or condition of the transaction	Categorical	object
14	state_number	Numeric encoding of the transaction state	Statistical	int64
15	ltime	Timestamp indicating the end of the record	Time	float64
16	seq	Sequence number from the Argus tool	Statistical	int64
17	dur	Total duration of the transaction	Time	float64
18	mean	Average duration across aggregated records	Statistical	float64
19	stddev	Standard deviation of durations in aggregated records	Statistical	float64
20	sum	Sum of durations across aggregated records	Statistical	float64
21	min	Minimum duration within aggregated records	Statistical	float64
22	max	Maximum duration within aggregated records	Statistical	float64
23	spkts	Number of packets sent from the source to the destination	Statistical	int64
24	dpkts	Number of packets sent from the destination to the source	Statistical	int64
25	sbytes	Byte count for source-to-destination data transfer	Statistical	int64
26	dbytes	Byte count for destination-to-source data transfer	Statistical	int64
27	rate	Packets transferred per second in the transaction	Time	float64
28	srate	Source-to-destination packet transfer rate	Time	float64
29	drate	Destination-to-source packet transfer rate	Time	float64
30	TnBPSrcIP	Total bytes transferred per source IP	Statistical	int64
31	TnBPDstIP	Total bytes transferred per destination IP	Statistical	int64
32	TnP_PSrcIP	Total packets transferred per source IP	Statistical	int64
33	TnP_PDstIP	Total packets transferred per destination IP	Statistical	int64
34	TnP_PerProto	Total packets for each protocol type	Statistical	int64
35	TnP_Per_Dport	Total packets transferred per destination port	Statistical	int64
36	AR_P_Proto_P_SrcIP	Average packet rate per protocol for each source IP	Statistical	float64
37	AR_P_Proto_P_DstIP	Average packet rate per protocol for each destination IP	Statistical	float64
38	N_IN_Conn_P_DstIP	Number of inbound connections per destination IP	Statistical	int64
39	N_IN_Conn_P_SrcIP	Number of inbound connections per source IP	Statistical	int64
40	AR_P_Proto_P_Sport	Average packet rate per protocol for source port	Statistical	float64
41	AR_P_Proto_P_Dport	Average packet rate per protocol for destination port	Statistical	float64
42	Pkts_P_State_P_Protocol_P_DestIP	Packet count grouped by state and protocol for each destination IP	Statistical	int64
43	Pkts_P_State_P_Protocol_P_SrcIP	Packet count grouped by state and protocol for each source IP	Statistical	int64
44	attack	Label indicating traffic type: 0 for normal, 1 for attack	Categorical	int64
45	category	General category of the traffic	Categorical	object
46	subcategory	Specific subcategory of the traffic	Categorical	object

Evaluation Environment Overview

The dataset was generated within a sophisticated testbed environment meticulously designed to simulate IoT traffic and various attack scenarios, as illustrated in Figure 6. The testbed consists of the following core modules [45]:

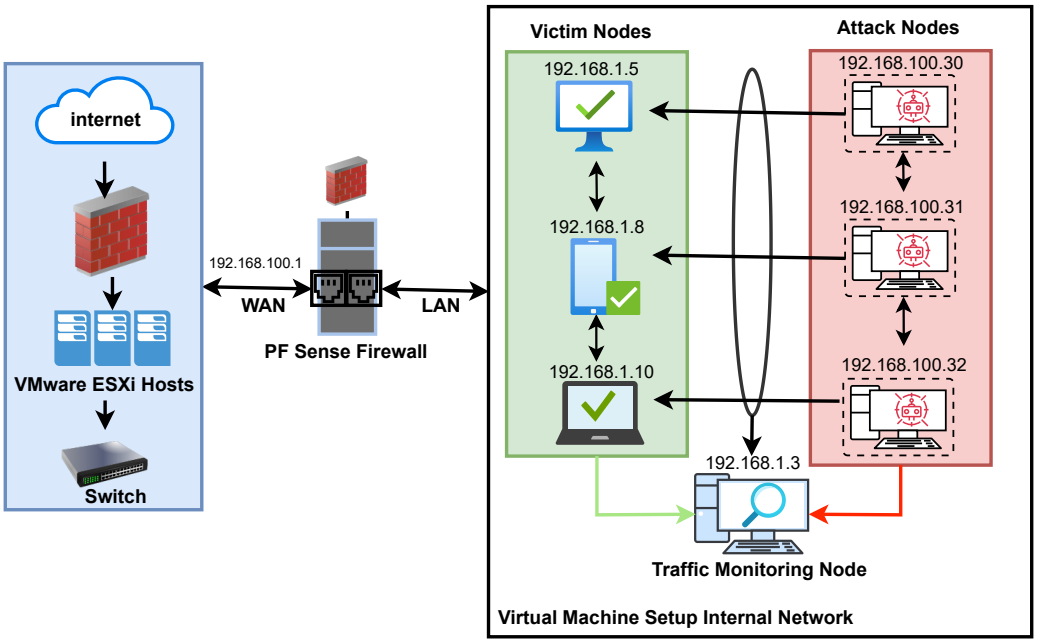

Figure 6. Testbed setup for the Bot-IoT dataset.

1. Network Platforms: The dataset comprises both normal and abnormal samples. Normal traffic includes benign communications from IoT devices (e.g., smart refrigerators, weather monitoring systems) interacting with their respective cloud services. Abnormal traffic is generated by attacker VMs running various attack scenarios (DDoS, DoS, Probing, Information Theft) aimed at IoT devices and network infrastructure. These VMs are managed within a vSphere platform on an ESXi cluster, including network devices like firewalls and taps for monitoring traffic. Captured traffic reflects real-world conditions, with benign IoT communications and malicious activities captured to evaluate the IDS framework's detection capabilities.
2. Simulated IoT Services: IoT services were simulated using the Node-RED middleware, which facilitates lightweight communication between IoT devices and cloud servers via the MQTT protocol. JavaScript scripts developed within Node-RED emulate temperature, pressure, and humidity sensors, periodically publishing data to the AWS IoT hub. This setup realistically represents smart home device interactions, including regular device-to-cloud communication.
3. Feature Extraction and Forensic Analytics: The captured network traffic, stored in pcap files, is processed using the Argus tool to extract flow-based features. Features such as packet counts, byte counts, and traffic rates are computed and stored in CSV format for further analysis.

The experiments were performed on an HP laptop running Windows 10 Pro Enterprise 64-bit, equipped with an Intel Core i7-5500 CPU (2.40 GHz, dual-core, four logical processors) and 16 GB of RAM, supported by 14.6 GB of virtual memory. Python 3.10 was utilized for implementation, with PyCharm (2022.2) as the development environment. Data preprocessing was carried out using Pandas (1.5.2) and NumPy (1.26.0), while machine learning and deep learning tasks were handled using Scikit-learn (1.1.3) and Keras with TensorFlow (2.10.0), respectively.

4.2. Performance Evaluation Metrics

Different performance evaluation metrics were employed to validate the proposed model's effectiveness, as described in the following sections.

4.2.1. Classification Performance Metrics

This section describes the metrics used to evaluate the performance of the proposed model. The commonly utilized metrics include *accuracy*, *precision*, *recall*, and the *f1 score*, which are defined in Equations (7)–(10). True positive (*TP*) and true negative (*TN*) refer to correctly classified positive and negative samples, respectively. Conversely, false positive (*FP*) and false negative (*FN*) represent positive and negative samples that are misclassified.

- Accuracy: measures the proportion of correct predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

- Precision: represents the model's ability to predict positive samples correctly.

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

- Recall/Detection rate: measures the model's ability to identify true positive samples.

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

- f1 score: combines precision and recall, offering a balanced view of the model's performance.

$$f1\ score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{10}$$

4.2.2. Resource Efficiency Metric

To comprehensively assess the efficacy of the proposed AIDS model tailored for IoT networks, it is imperative to evaluate its impact on network resources, particularly those targeted for optimization. Unfortunately, the datasets lack detailed information on device structures, such as battery capacity, energy usage, and bandwidth. Consequently, the only quantifiable resource at our disposal is the detection time. A decrease in detection time signifies a reduction in resource consumption and makes this able to be implemented in real-world scenarios. This metric acts as a proxy for resource consumption, reflecting the efficiency gains achieved by our proposed AIDS model within the constraints of IoT networks with inherent limitations. The detection time (*Time*) is defined as the duration between the moment the detection starts and the moment it ends, as depicted in Equation (11).

$$Detection\ Time = End - Start \tag{11}$$

4.3. Experimental Evaluation

This section presents the proposed model's analysis and the experiments' results. The K-fold cross-validation technique, with the k = 10, was utilized to evaluate the model. In this method, the dataset is divided into 10 equal parts or 'folds' at random. For each iteration, the model is trained on nine folds, while the remaining fold is used for validation. This process is repeated 10 times, ensuring each fold is used as the validation set once. The performance metrics are then averaged across all iterations to provide a robust estimate of the model's performance. This approach helps minimize overfitting by testing the model on various subsets of data, thereby reducing the bias associated with a single train-test split.

4.3.1. Analysis of ASAE Structure

This section explores the architectural aspects of the ASAE model. We discuss its key features and design choices, contrasting it with traditional autoencoders and highlighting its suitability for numeric IoT data.

Unlike some autoencoders that employ CNN layers [30], our model utilizes dense layers to align with the numeric nature of the input data and avoid unnecessary resource consumption associated with converting data for CNN. This decision prioritizes resource efficiency for IoT environments. The network structure significantly impacts classification performance. We determined the optimal configuration through experiments with various structural combinations as illustrated in Table 4. The encoder utilizes three dense layers with ReLU activation functions, reducing the initial 43 features to 20 in the bottleneck layer (also with ReLU). The decoder utilizes five dense layers and an output layer containing the reconstructed 43 features employing the sigmoid activation function in the last layer.

As this autoencoder is used in this paper for dimensionality reduction, we must force the network to concentrate on the most essential features. We apply the L1 or Lasso regularizer module to the activity of a first layer in the encoder. L1 regularization is highly effective for feature selection in high-dimensional datasets. By introducing a penalty term to the loss function, sparse weights in the network are promoted. This sparsity minimizes overfitting by discouraging excessive reliance on specific features. A learning rate of 0.00001 is applied to the L1 function to define the regularization intensity. The model is compiled with the Mean Squared Error (MSE) loss function and the Adam optimizer.

The selection of a batch size for training was purposefully made from a range of options, including 1, 8, 16, 32, 64, and 128, with a specific choice of 1. This selection demonstrates its benefits, particularly when confronted with a constrained training set. In our case, the training dataset for normal samples, as mentioned previously, consists of 477 samples, and after splitting, the resulting number became relatively small.

We also implement the early stopping technique on 100 epochs to avoid overfitting, optimize the training process, and save computational resources. Early stopping is a regularization technique that halts the training process when the model's performance on the validation set begins to decline.

Table 4. Configuration Details of the Autoencoder.

Component	Subcomponent	Details
Encoder	Input	43 features
	Output	20 features (bottleneck layer)
	Activation Function	ReLU
	Layers	Three dense layers
Decoder	Input	20 features (from bottleneck layer)
	Output	43 features (reconstructed output)
	Activation Function	Sigmoid
	Layers	Five dense layers
Loss Function	–	Minimum Square Error (MSE)
Optimizer	–	Adam optimizer
Batch Sizes	–	1, 8, 16, 32, 64, 128
Training Epochs	–	100

4.3.2. Performance of ASAE Against Traditional Autoencoders

Following the analysis of the ASAE structure, we implemented and evaluated the proposed model alongside various traditional autoencoders on the BoT-IoT dataset. The comparison focuses on models with 20 dimensions, which were used with a DNN classifier, as these parameters yielded the best results for the ASAE model. Additionally, the DNN configuration, including the number of layers, number of neurons per layer, and other hyperparameters, was fixed across all autoencoders to control for potential confounding factors. This setup demonstrated that detection accuracy and performance improvements are primarily due to the ASAE architecture rather than differences in the neural network structure or training parameters. Table 5 summarizes the quantitative assessment of how the ASAE model performs compared with traditional autoencoders.

Table 5. Performance of ASAE and its variants.

AE_Method	Accuracy	Precision	Recall	f1 Score	Detection Time
Traditional AE	97.8231	65.1290	65.2331	65.1814	0.4381
Sparse AE	98.9812	97.4398	97.7650	97.6008	0.3921
Variational AE	95.9900	94.8782	92.9867	93.9314	0.8576
ASAE	**99.9989**	**99.2059**	**96.2686**	**97.6920**	**0.2746**

Bold text indicates the optimal result.

The results highlight the remarkable performance of the ASAE model in anomaly detection compared with traditional, sparse, and variational autoencoders. In terms of accuracy, ASAE achieves an outstanding 99.9989%, far surpassing traditional autoencoders at 97.8231% and variational autoencoders at 95.9900% and showing a clear improvement over sparse autoencoders at 98.9812%, which indicates that ASAE captures the latent representations of data more effectively, leading to significantly more reliable classifications. This improvement can be attributed to ASAE's increased capacity to capture and utilize latent data representations, resulting in more accurate and exact classifications even in challenging conditions. The ASAE model's precision of 99.2059% demonstrates its superiority, a significant improvement over the sparse autoencoder at 97.4398%, and substantially better than the traditional autoencoder at 65.1290%. The rise in precision shows ASAE's ability to reduce false positives, ensuring that detected anomalies truly reflect actual threats or data abnormalities. ASAE outperforms traditional autoencoders at 65.2331% and variational autoencoders at 92.9867%, with a high recall value of 96.2686%, comparable to sparse autoencoders at 97.7650%. This result implies that ASAE detects the vast majority of anomalies in the dataset, demonstrating strong performance even under challenging conditions. ASAE's f1 score of 97.6920% reflects its ability to detect anomalies while accurately avoiding false positives.

ASAE additionally stands out in detection time, with the shortest duration of 0.2746 s recorded. This efficiency is especially significant for real-time applications that require quick anomaly detection. The variational autoencoder has the longest detection time of 0.8576 s, while traditional and sparse autoencoders follow the ASAE at 0.4381 and 0.3921 s, respectively. ASAE's low computing overhead makes it suitable for resource-constrained or latency-sensitive situations while providing accurate results.

4.3.3. Performance of the Classifier

This section discusses the architecture and performance of the DNN, the anomaly detector, in the proposed AIDS model summarized in Table 6. It utilizes six dense layers and prevents overfitting by incorporating dropout layers at specific points with various rates (0.5, 0.2, 0.2, 0.1, 0.1, 0.1). Additionally, early stopping with a maximum of 200 epochs is implemented to optimize training. We employ the L2 regularization, also known as

the Ridge regularizer module, which penalizes large weights more severely than smaller ones and promotes the model's distribution across all input features, enhancing generalization. The learning rate passed to this module is set at 0.001. All dense layers employ the ReLU activation function except for the output layer, which utilizes the sigmoid function. The binary_cross-entropy loss and Adam optimizers are used when compiling, and a batch size of 1 is used.

Table 6. Learning Configuration for Deep Neural Network.

Component	Details
Layers	six dense and six dropout
Training Epochs	200
Regularizer	L2
Activation Function	ReLU and Sigmoid
Learning Rate	0.001
Loss Function	binary_cross-entropy
Optimizer	Adam
Batch Size	1

Having established the practical DNN framework, we deploy diverse classifiers on the Bot-IoT dataset without applying the proposed ASAE, as summarized in Table 7. These classifiers include Sigmoid and DNN as deep learning models, complemented by One-Class Support Vector Machine (OCSVM) and Isolation Forest (IF) as machine learning models. This comparison aims to understand the impact of autoencoder features on classification.

Table 7. Performance of state-of-the-art classifiers without ASAE.

Method	No.Epochs	Threshold	Accuracy	Precision	Recall	f1 Score	Detection Time
Sigmoid	6	0.00051961	99.0336	49.9192	49.9990	49.7666	**0.2798**
DNN	11	7.144177×10^{-14}	**99.8767**	**97.9564**	**97.9565**	**81.9797**	0.3119
OCSVM	-	-	00.0105	43.2543	14.5883	00.0106	0.5983
IF	-	-	21.6777	15.9982	49.9710	17.8165	0.4303

Bold text indicates the optimal result.

The Sigmoid classifier demonstrates moderate accuracy of 99.03% but exhibits relatively low precision, recall, and f1 score, indicating challenges distinguishing between positive and negative classes. This leads to a notable number of false positives and false negatives. The choice of Sigmoid, potentially motivated by simplicity, may not be optimal for capturing complex data patterns.

In contrast, the DNN classifier surpasses Sigmoid with a high accuracy of 99.88% and superior precision, recall, and f1 score, showcasing the capacity of a deeper neural network to discern intricate relationships in the data. The extremely low threshold 7.144177×10^{-14} indicates a high confidence threshold for classification, and the elevated f1 score underscores its effectiveness in handling imbalanced datasets.

OCSVM is designed for one-class problems and demonstrates a low accuracy of 0.01%. Its high recall suggests an ability to identify samples of the minority class. The Isolation Forest exhibits improved accuracy by 21.68% compared with OCSVM.

Considering their operational characteristics, the absence of explicit threshold values for OCSVM and Isolation Forest is reasonable. These models might not provide explicit threshold values because they do not rely on probability scores like other classifiers. OCSVM separates normal samples from the origin in feature space, and Isolation Forest isolates anomalies based on the number of partitions required. Therefore, traditional threshold values may not be applicable.

4.3.4. Performance of the Proposed Model

This section investigates how the autoencoder affects classifiers by reducing their dimensionality. The encoded representation from the autoencoder acts as input for the following classifier. We analyze the autoencoder's impact on enhancing feature representation and facilitating improved classification results.

Table 8 evaluates the hybrid model's performance based on the autoencoder's encoded representations (AE_Configuration), followed by classification by various classifiers (Classifier_Configuration). Notably, each configuration explores different dimensions and training epochs for the autoencoder, while the classifiers maintain a consistent number of epochs and different thresholds.

Table 8. Performance of the proposed model.

AE_Configuration			Classifier_Configuration			Accuracy	Precision	Recall	f1 Score	Detection Time
Dimensions	**Epochs**	**Threshold**	**Epochs**	**Threshold**	**Type**					
8	37	0.0484	6	0.0002	Sigmoid	99.9620	95.2196	60.9610	67.6376	**0.2670**
	20	0.0431	6	3.7156×10^{-9}	DNN	99.8759	96.3669	53.9684	57.2814	0.2726
	28	0.0510	-	-	OCSVM	71.4784	46.8532	49.9982	41.6923	0.2967
	41	0.0484	-	-	IF	73.1973	37.7932	49.9929	42.2634	0.6364
10	44	0.0439	6	0.0006	Sigmoid	99.9897	**99.9949**	76.3598	84.5180	0.2724
	29	0.0449	6	0.0003	DNN	99.9874	96.8194	73.6944	81.4635	0.2731
	20	0.0439	-	-	OCSVM	57.9091	40.0677	49.9953	36.6781	0.3404
	28	0.0388	-	-	IF	99.9861	50.3956	51.7185	50.6417	0.5260
15	34	0.0381	6	0.0015	Sigmoid	99.9883	96.4231	74.6831	82.2285	0.2764
	27	0.0442	6	9.9143×10^{-22}	DNN	99.9906	96.4243	77.7247	84.7158	0.2743
	24	0.0472	-	-	OCSVM	99.9883	96.4231	74.6831	82.2285	0.2820
	40	0.0399	-	-	IF	99.9874	50.3962	53.3277	50.7061	0.2688
20	33	0.0444	6	0.0018	Sigmoid	99.9952	98.8073	85.5490	91.1359	0.2775
	28	0.0533	6	0.0039	DNN	**99.9989**	99.2059	**96.2686**	**97.6920**	0.2746
	41	0.0409	-	-	OCSVM	99.4750	60.4563	50.2321	50.3314	0.3940
	43	0.0412	-	-	IF	99.9006	51.9399	50.2501	50.4280	0.5012

Bold text indicates the optimal result.

Regarding the hybrid model, Sigmoid and DNN classifiers exhibit varying performance based on different autoencoder configurations. Notably, DNN consistently outperforms Sigmoid in accuracy, precision, recall, and f1 score across all configurations. This reinforces the idea that a deeper neural network with dimensionality reduction layers can better capture complex data patterns, making it a more suitable choice for the proposed model. On the contrary, the OCSVM and Isolation Forest classifiers designed for one-class problems displayed mixed performance. OCSVM has difficulty with imbalanced datasets, which is evident from its lower accuracy and precision. Meanwhile, Isolation Forest shows improved accuracy and can find anomalies in the dataset.

The proposed model's results assert the autoencoder's positive impact on feature representation. In most cases, the proposed model outperforms individual classifiers. The encoded representations from the autoencoder contribute to improved accuracy, precision, recall, and f1 score, highlighting the effectiveness of the dimensionality reduction stage. Temporal factors are crucial for assessing the performance of the proposed model for AIDS-IoT. Beyond standard metrics like accuracy, precision, recall, and f1 score, the time required for detection is an important consideration, especially in real-time or resource-constrained scenarios like those seen in IoT. The reported time values in Tables 5, 7 and 8 indicate the computational efficiency of each configuration. Lower time values are desirable, especially in applications requiring quick responses. The DNN classifier consistently demonstrates competitive time efficiency across various autoencoder configurations, reinforcing its suitability for real-time or resource-limited scenarios.

To comprehensively assess the effectiveness of the autoencoder on individual classes (normal and abnormal), we meticulously compiled the evaluation metrics for each class across various classifiers and distinct dimension settings.

The Figure denoted as Figure 7 illustrates these evaluations. For instance, in Figure 7a, "DNN-8" denotes the DNN configuration with eight dimensions, and "DNN-10" denotes the DNN with ten dimensions. Precision-0, recall-0, and f1-0 correspond to the precision, recall, and f1 score metrics for the "normal" class, while precision-1, recall-1, and f1-1 correspond to the precision, recall, and f1 score metrics for the "abnormal" class. Also, this applies to Figure 7b–d for the other classifiers. In addition, Figures 8 and 9 illustrate the ROC curve and confusion matrix for the proposed mode based on 20 dimensions.

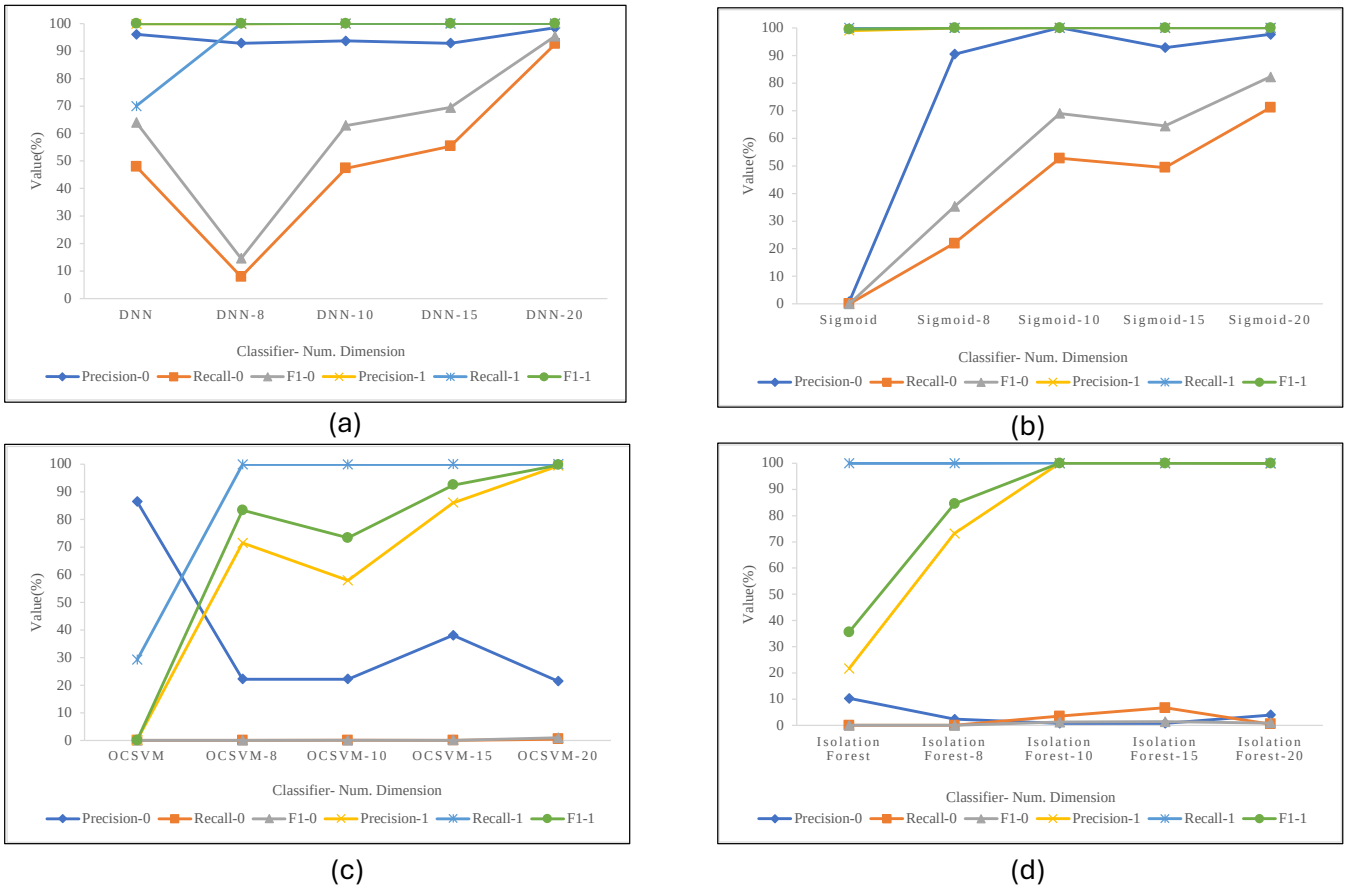

Figure 7. In-depth analysis of class-wise metrics for (**a**) DNN, (**b**) Sigmoid, (**c**) OCSVM, (**d**) IF.

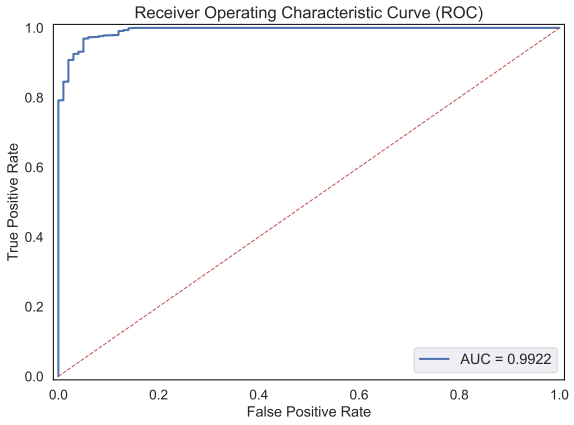

Figure 8. Roc curve for the proposed model using 20-dimensional feature space.

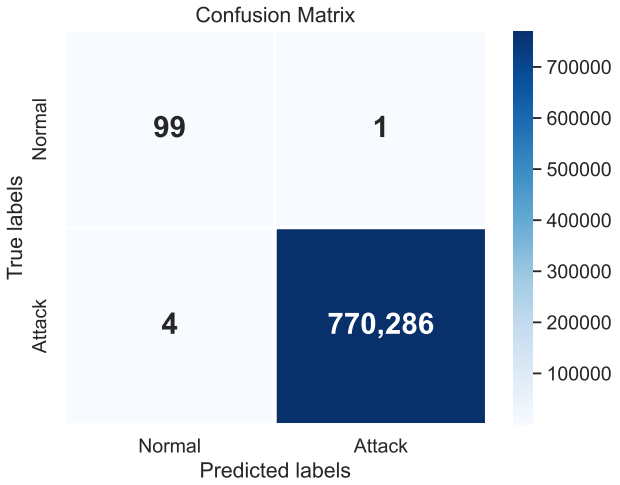

Figure 9. Confusion matrix for the proposed model using 20-dimensional feature space.

4.3.5. A Comparison with Previous Research

This section presents a comparison of the proposed OC-ASAE and DNN models with several advanced approaches, emphasizing key performance metrics such as accuracy, precision, recall, f1 score, and detection time. As demonstrated in Table 9, our model delivers outstanding results, significantly outperforming many contemporary methods.

Our proposed model, integrating a One-Class Asymmetric Autoencoder (OC-ASAE) with a DNN, demonstrates significant advancements over state-of-the-art IDS. It achieves an accuracy of 99.9989%, surpassing the DNN model by Shareena et al. [46] and the CNN model by Saba et al. [47] by about 5.8% and 7.1%, respectively. This outstanding accuracy demonstrates the proposed model's superior ability to handle complicated IoT data compared with traditional techniques.

The proposed model achieves 99.2059% precision, surpassing the DNN model by Shareena et al. [46] and the OC-CNN model by Tran et al. [48] by 4.2% and 7.9%, respectively. Furthermore, it surpasses at minimizing false positives, with precision levels above those of autoencoder-based techniques such as the OC-AE+OCSVM model by Dong and Kotenko [49] and the DSAE model by Kalidindi and Arrama [50] by 25.84% and 10.81%, respectively. These results demonstrate our model's ability to detect precise anomalies, which is crucial for reducing unnecessary alarms in IoT systems.

The proposed model's recall of 96.2686% surpasses competing approaches, including the Long Short-Term Memory (LSTM) by Sharma and Prasad [51], which achieves a recall of 96.32%, and significantly outperforms the GRU and Recurrent Neural Network (RNN) models by Alabsi et al. [52], which show limited accuracies of 69.50% and 69.30%, respectively. This highlights our model's ability to detect diverse threats while retaining effective detection rates.

The f1 score of the proposed model of 97.6920% demonstrates its significant detection capacity without generating high false positives. Our model provides a more reliable solution than LSTM-based [51,53], which often struggles to achieve this balance. Notably, the detection time of 0.2746 s underscores the suitability of our approach for real-time IoT applications, outperforming CCNN's 57.02 s [54] and establishing a benchmark for efficiency.

In contrast, existing methods face limitations that hinder their applicability and effectiveness in IoT environments. While models like those proposed by Sharma and Prasad [51] and Khanday et al. [53] focus on lightweight designs, their accuracies of 95.39% and 95%, respectively, fall short of delivering the high detection rates necessary for IoT systems. Similarly, Bojarajulu and Tanwar's [54] CCNN model, despite addressing class imbalance with

SMOTE-ENC, achieves only 94.24% accuracy. Advanced approaches, such as the MRFM by Xie et al. [55], offer a promising accuracy of 99.81% but lack comprehensive evaluations of resource efficiency critical for IoT deployment. Additionally, autoencoder-based techniques, like Liu et al.'s AE+MLP [56], exhibit lower precision and recall rates while effectively learning compressed representations.

These limitations highlight the need for a model like ours, which addresses these shortcomings and delivers unparalleled accuracy, precision, and efficiency. By surpassing these limitations, our OC-ASAE+DNN model establishes itself as a robust solution for intrusion detection in IoT networks, capable of meeting the demands of real-time, high-performance anomaly detection.

Table 9. Comparative analysis of the proposed models with the state-of-the-art approaches.

Year	Publication	Technique	Accuracy	Precision	Recall	f1 Score	Detection Time
2021	Liu et al. [56]	AE+MLP	-	-	-	95	-
2021	Shareena et al. [46]	DNN	94.00	95.00	93.00	94.00	–
2022	Saba et al. [47]	CNN	92.85	–	–	–	–
2022	Tran et al. [48]	OC-CNN	-	91.3260	79.869	85.2140	-
2023	Dong&Kotenko [49]	OC-AE+OCSVM	97.94	73.3700	92.24	81.73	-
2023	Kalidindi&Arrama [50]	DSAE	-	88.4	87.4	87.9	-
2023	Awajan [57]	AE	91.56	-	-	89.32	-
2023	Dina et al. [58]	CNN	86.77	-	-	-	-
2023	Dina et al. [58]	FNN	91.55	-	-	-	-
2023	Hou et al. [30]	NAE + DNN	99.80	-	-	-	-
2023	Khanday et al. [53]	LSTM	95	-	-	-	-
2023	Alabsi et al. [52]	LSTM	97.8	-	-	-	-
2023	Alabsi et al. [52]	RNN	69.3	-	-	-	-
2023	Alabsi et al. [52]	GRU	69.5	-	-	-	-
2024	Bojarajulu&Tanwar [54]	CCNN	94.2446	96.33.3	-	96.3303	57.02474
2024	Xie et al. [55]	MRFM	99.81	-	-	-	-
2024	Sharma& Prasad [51]	LSTM	95.39	94.12	96.32	95.23	-
2024	**Proposed**	**OC-ASAE+DNN**	**99.9989**	**99.2059**	**96.2686**	**97.6920**	**0.2746**

Bold text indicates the proposed work.

5. Discussion

Combining the DNN classifier with the Asymmetric Autoencoder with one-class learning generates a hybrid model that outperforms IoT-based IDS. This combination captures essential features with highly effective accuracy, precision, recall, and f1 score, making it a reliable approach for high-dimensional data and class imbalance concerns in IoT networks. The model's computational efficiency is seen in its optimal detection time of 0.2746 s, making it ideal for real-time IoT applications that need quick detection. This effective balance of high-performance metrics and detection time demonstrates its suitability for implementation in resource-constrained environments.

DNNs have significant advantages for IoT-based IDS applications, mainly when dealing with high-dimensional numerical data, unlike models as sequential as RNNs, LSTMs, and GRUs, or as convolutional as CNNs, which have specific applications in tasks involving temporal or spatial data. RNNs, LSTMs, and GRUs are designed to handle time series data but are less effective for IoT data, which typically lack strong sequential dependencies. While CNNs excel in image or spatial data processing, they are more complex and computationally intensive when applied to high-dimensional numerical data. DNNs, on the other hand, offer a simpler and more efficient solution, avoiding issues like the vanishing gradient problem found in RNNs and providing a streamlined approach for anomaly detection in IoT environments.

Furthermore, the OC-ASAE improves DNN performance by focusing on the most significant features, reducing complexity, and addressing class imbalance concerns, which are crucial in IoT. This combination ensures the model generalizes well to unseen data, offering scalability and reliability for intrusion detection in IoT networks. Experimental

results confirm the superiority of the proposed OC-ASAE-DNN-based IDS model over other state-of-the-art architectures. With a detection time of just 0.2746 s, the model outperforms more complex models like LSTM and GRU, which demand more excellent computational resources without offering proportional improvements in performance. Precision–recall curves were utilized to optimize anomaly detection thresholds, resulting in a balanced trade-off between precision and recall and minimizing false positives and negatives. Compared with models such as GAN-based approaches like CTGAN, which are more suited for synthetic data generation than anomaly detection, the DNN-based approach excels in both performance and efficiency. The proposed model significantly surpasses traditional and advanced models, such as those developed by Sharma and Prasad [51], Bojarajulu and Tanwar [54], and Xie et al. [55], particularly in detection accuracy, precision, and recall.

While the detection time of 0.2746 s is optimal for most IoT applications, such as smart homes and industrial networks, further optimization will be necessary for domains with stricter real-time requirements, such as automotive systems governed by IEEE 802.1 standards [59].

6. Conclusions

Securing IoT devices in today's interconnected world is of significant concern. As the IoT network expands, robust security procedures must be proposed to safeguard the devices and data on this network. Intrusion Detection Systems effectively identify any deviation from normal behavior. Integrating IDS into IoT networks strengthens security, protects sensitive data, and ensures the integrity and privacy of IoT devices and users.

This paper advances the state of the art by presenting a lightweight AIDS for IoT devices. This model contains several stages: datasets, preprocessing, dimensionality reduction, and anomaly detection. The first stage aims to select a real-time dataset for our model. Then, we proceed with suitable preprocessing techniques that include encoding and normalization. We employ a one-class asymmetric stacked autoencoder for dimensionality reduction, which helps efficiently reduce the dataset's dimensionality. Finally, the DNN is trained using a one-class approach, and a threshold is adjusted using the precision–recall curve to detect deviations from normal behavior. The proposed model can accurately identify abnormal traffic and solve the high imbalance problem in the dataset. It outperformed other attempts with an accuracy equal to 99.9989%, an f1 score equal to 97.6920%, and a detection time of 0.2746 s.

In the future, we aim to expand the classification capabilities to identify specific types of detected assaults at a secondary classification level, enhancing our detection model's granularity and interpretability. These will improve the overall performance and introduce a more nuanced understanding of the detected anomalies.

Author Contributions: Conceptualization, A.G.A., N.A.S., M.M.E.-G. and N.A.H.; methodology, A.G.A., N.A.S., M.M.E.-G. and N.A.H.; software, A.G.A.; validation, N.A.S., M.M.E.-G. and N.A.H.; formal analysis, A.G.A.; investigation, N.A.S., M.M.E.-G. and N.A.H.; resources, A.G.A.; data curation, A.G.A.; writing—original draft preparation, A.G.A., N.A.S. and N.A.H.; writing—review and editing, A.G.A., N.A.S., M.M.E.-G. and N.A.H.; visualization, A.G.A. and N.A.S.; supervision, N.A.S., M.M.E.-G. and N.A.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The dataset analyzed during the current study, BoT-IoT dataset https://unsw-my.sharepoint.com/:f:/g/personal/z5131399_ad_unsw_edu_au/EjlBDf2KODxPgXmqbO3MxxsBBVARCKZxGUG47OiFHb_AnQ, accessed on 1 September 2024.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Koohang, A.; Sargent, C.S.; Nord, J.H.; Paliszkiewicz, J. Internet of Things (IoT): From awareness to continued use. *Int. J. Inf. Manag.* **2022**, *62*, 102442. [CrossRef]
2. Mohammed, R.J.; Abed, E.A.; Elgayar, M.M. Comparative study between metaheuristic algorithms for internet of things wireless nodes localization. *Int. J. Electr. Comput. Eng. (IJECE)* **2022**, *12*, 660–668. [CrossRef]
3. Nimodiya, A.R.; Ajankar, S.S. A Review on Internet of Things. *Int. J. Adv. Res. Sci. Commun. Technol.* **2022**, *113*, 135–144. [CrossRef]
4. Hussain, F. *Internet of Things: Building Blocks and Business Models*; Number 978-3; Springer: Berlin/Heidelberg, Germany, 2017.
5. Hussain, F.; Hussain, R.; Hassan, S.A.; Hossain, E. Machine learning in IoT security: Current solutions and future challenges. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1686–1721. [CrossRef]
6. Ali, O.; Ishak, M.K.; Bhatti, M.K.L. Emerging IoT domains, current standings and open research challenges: A review. *PeerJ Comput. Sci.* **2021**, *7*, e659. [CrossRef]
7. Jeyanthi, D.; Indrani, B. Intrusion Detection System Intensive on Securing IoT Networking Environment Based on Machine Learning Strategy. In *Intelligent Data Communication Technologies and Internet of Things*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 139–157. [CrossRef]
8. Mishra, N.; Pandya, S. Internet of things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review. *IEEE Access* **2021**, *9*, 59353–59377. [CrossRef]
9. Hikal, N.A.; Elgayar, M. Enhancing IoT botnets attack detection using machine learning-IDS and ensemble data preprocessing technique. In *Internet of Things—Applications and Future*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 89–102. [CrossRef]
10. Heidari, A.; Jabraeil Jamali, M.A. Internet of Things intrusion detection systems: A comprehensive review and future directions. *Clust. Comput.* **2023**, *26*, 3753–3780. [CrossRef]
11. Elrawy, M.F.; Awad, A.I.; Hamed, H.F. Intrusion detection systems for IoT-based smart environments: A survey. *J. Cloud Comput.* **2018**, *7*, 21. [CrossRef]
12. Jyothsna, V.; Prasad, R.; Prasad, K.M. A review of anomaly based intrusion detection systems. *Int. J. Comput. Appl.* **2011**, *28*, 26–35. [CrossRef]
13. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 20. [CrossRef]
14. Ghafir, I.; Husak, M.; Prenosil, V. A survey on intrusion detection and prevention systems. In Proceedings of the Student Conference Zvule, IEEE/UREL, Brno University of Technology, Zvůle, Czech Republic, 25–27 August 2014; Volume 1014.
15. Thakkar, A.; Lohiya, R. A review on machine learning and deep learning perspectives of IDS for IoT: Recent updates, security issues, and challenges. *Arch. Comput. Methods Eng.* **2021**, *28*, 3211–3243. [CrossRef]
16. Panigrahi, R.; Borah, S.; Bhoi, A.K.; Mallick, P.K. Intrusion detection systems (IDS)—An overview with a generalized framework. In Proceedings of the Cognitive Informatics and Soft Computing, Balasore, India, 12–13 December 2020; pp. 107–117. [CrossRef]
17. Lin, K.; Xu, X.; Xiao, F. MFFusion: A Multi-level Features Fusion Model for Malicious Traffic Detection based on Deep Learning. *Comput. Netw.* **2022**, *202*, 108658. [CrossRef]
18. Alsoufi, M.A.; Razak, S.; Siraj, M.M.; Nafea, I.; Ghaleb, F.A.; Saeed, F.; Nasser, M. Anomaly-based intrusion detection systems in iot using deep learning: A systematic literature review. *Appl. Sci.* **2021**, *11*, 8383. [CrossRef]
19. Talaei Khoei, T.; Kaabouch, N. A Comparative Analysis of Supervised and Unsupervised Models for Detecting Attacks on the Intrusion Detection Systems. *Information* **2023**, *14*, 103. [CrossRef]
20. El-Gayar, M.M.; Alrslani, F.A.; El-Sappagh, S. Smart Collaborative Intrusion Detection System for Securing Vehicular Networks Using Ensemble Machine Learning Model. *Information* **2024**, *15*, 583. [CrossRef]
21. Bakhsh, S.A.; Khan, M.A.; Ahmed, F.; Alshehri, M.S.; Ali, H.; Ahmad, J. Enhancing IoT network security through deep learning-powered Intrusion Detection System. *Internet Things* **2023**, *24*, 100936. [CrossRef]
22. Yu, Y.; Long, J.; Cai, Z. Network intrusion detection through stacking dilated convolutional autoencoders. *Secur. Commun. Netw.* **2017**, *2017*, 4184196. [CrossRef]
23. Aygun, R.C.; Yavuz, A.G. Network anomaly detection with stochastically improved autoencoder based models. In Proceedings of the 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), New York, NY, USA, 26–28 June 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 193–198.
24. Zhang, H.; Wu, C.Q.; Gao, S.; Wang, Z.; Xu, Y.; Liu, Y. An effective deep learning based scheme for network intrusion detection. In Proceedings of the 2018 24th International Conference on Pattern Recognition (ICPR), Beijing, China, 20–24 August 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 682–687.
25. Mao, S.; Guo, J.; Li, Z. Discriminative autoencoding framework for simple and efficient anomaly detection. *IEEE Access* **2019**, *7*, 140618–140630. [CrossRef]
26. Ieracitano, C.; Adeel, A.; Morabito, F.C.; Hussain, A. A novel statistical analysis and autoencoder driven intelligent intrusion detection approach. *Neurocomputing* **2020**, *387*, 51–62. [CrossRef]

27. Binbusayyis, A.; Vaiyapuri, T. Unsupervised deep learning approach for network intrusion detection combining convolutional autoencoder and one-class SVM. *Appl. Intell.* **2021**, *51*, 7094–7108. [CrossRef]
28. Adeniyi, E.A.; Folorunso, S.O.; Jimoh, R.G. A Deep Learning-Based Intrusion Detection Technique for a Secured IoMT System. In Proceedings of the Informatics and Intelligent Applications: First International Conference, ICIIA 2021, Ota, Nigeria, 25–27 November 2021; Revised Selected Papers; Springer Nature: Berlin/Heidelberg, Germany, 2022; p. 50. [CrossRef]
29. Yao, W.; Hu, L.; Hou, Y.; Li, X. A Lightweight Intelligent Network Intrusion Detection System Using One-Class Autoencoder and Ensemble Learning for IoT. *Sensors* **2023**, *23*, 4141. [CrossRef] [PubMed]
30. Hou, Y.; Fu, Y.; Guo, J.; Xu, J.; Liu, R.; Xiang, X. Hybrid intrusion detection model based on a designed autoencoder. *J. Ambient Intell. Humaniz. Comput.* **2023**, *14*, 10799–10809. [CrossRef]
31. Bisong, E. *Building Machine Learning and Deep Learning Models on Google Cloud Platform*; Springer: Berlin/Heidelberg, Germany, 2019. [CrossRef]
32. Laurent, C.; Pereyra, G.; Brakel, P.; Zhang, Y.; Bengio, Y. Batch normalized recurrent neural networks. In Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, China, 20–25 March 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 2657–2661.
33. Han, J.; Pei, J.; Tong, H. *Data Mining: Concepts and Techniques*; Morgan Kaufmann: Burlington, MA, USA, 2022.
34. Adhikari, D.; Jiang, W.; Zhan, J.; Rawat, D.B.; Bhattarai, A. Recent advances in anomaly detection in Internet of Things: Status, challenges, and perspectives. *Comput. Sci. Rev.* **2024**, *54*, 100665. [CrossRef]
35. Wang, Y.; Yao, H.; Zhao, S. Auto-encoder based dimensionality reduction. *Neurocomputing* **2016**, *184*, 232–242. [CrossRef]
36. Sakurada, M.; Yairi, T. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis, Gold Coast, QLD, Australia, 2 December 2014; pp. 4–11.
37. Mousa, A.K.; Abdullah, M.N. An improved deep learning model for DDoS detection based on hybrid stacked autoencoder and checkpoint network. *Future Internet* **2023**, *15*, 278. [CrossRef]
38. Sun, Y.; Mao, H.; Guo, Q.; Yi, Z. Learning a good representation with unsymmetrical auto-encoder. *Neural Comput. Appl.* **2016**, *27*, 1361–1367. [CrossRef]
39. Sharma, S.; Sharma, S.; Athaiya, A. Activation functions in neural networks. *Towards Data Sci.* **2017**, *6*, 310–316. [CrossRef]
40. Tajoddin, A.; Abadi, M. RAMD: Registry-based anomaly malware detection using one-class ensemble classifiers. *Appl. Intell.* **2019**, *49*, 2641–2658. [CrossRef]
41. Khan, S.S.; Madden, M.G. One-class classification: Taxonomy of study and review of techniques. *Knowl. Eng. Rev.* **2014**, *29*, 345–374. [CrossRef]
42. Tharwat, A. Classification assessment methods. *Appl. Comput. Inform.* **2020**, *17*, 168–192. [CrossRef]
43. Fawcett, T. An introduction to ROC analysis. *Pattern Recognit. Lett.* **2006**, *27*, 861–874. [CrossRef]
44. Ozenne, B.; Subtil, F.; Maucort-Boulch, D. The precision–recall curve overcame the optimism of the receiver operating characteristic curve in rare diseases. *J. Clin. Epidemiol.* **2015**, *68*, 855–859. [CrossRef] [PubMed]
45. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. [Dataset] Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Gener. Comput. Syst.* **2019**, *100*, 779–796. [CrossRef]
46. P, J.; Shareena, J.; Ramdas, A.; AP, H. Intrusion detection system for iot botnet attacks using deep learning. *SN Comput. Sci.* **2021**, *2*, 205. [CrossRef]
47. Saba, T.; Rehman, A.; Sadad, T.; Kolivand, H.; Bahaj, S.A. Anomaly-based intrusion detection system for IoT networks through deep learning model. *Comput. Electr. Eng.* **2022**, *99*, 107810. [CrossRef]
48. Tran, D.H.; Nguyen, V.L.; Nguyen, H.; Jang, Y.M. Self-Supervised Learning for Time-Series Anomaly Detection in Industrial Internet of Things. *Electronics* **2022**, *11*, 2146. [CrossRef]
49. Dong, H.; Kotenko, I. Train Without Label: A Self-supervised One-Class Classification Approach for IoT Anomaly Detection. In *Proceedings of the International Conference on Intelligent Information Technologies for Industry*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 81–89.
50. Kalidindi, A.; Arrama, M.B. Botnet attack detection in IoT using hybrid optimisation enabled deep stacked autoencoder network. *Int. J. Bio-Inspired Comput.* **2023**, *22*, 77–88. [CrossRef]
51. Sharma, T.; Prasad, S.K. Enhancing cybersecurity in IoT networks: SLSTM-WCO algorithm for anomaly detection. *Peer-to-Peer Netw. Appl.* **2024**, *17*, 2237–2258. [CrossRef]
52. Alabsi, B.A.; Anbar, M.; Rihan, S.D.A. Conditional tabular generative adversarial based intrusion detection system for detecting ddos and dos attacks on the internet of things networks. *Sensors* **2023**, *23*, 5644. [CrossRef]
53. Khanday, S.A.; Fatima, H.; Rakesh, N. Implementation of intrusion detection model for DDoS attacks in Lightweight IoT Networks. *Expert Syst. Appl.* **2023**, *215*, 119330. [CrossRef]
54. Bojarajulu, B.; Tanwar, S. Customized convolutional neural network model for IoT botnet attack detection. *Signal Image Video Process.* **2024**, *18*, 5477–5489. [CrossRef]

55. Xie, L.; Yuan, B.; Yang, H.; Hu, Z.; Jiang, L.; Zhang, L.; Cheng, X. MRFM: A timely detection method for DDoS attacks in IoT with multidimensional reconstruction and function mapping. *Comput. Stand. Interfaces* **2024**, *89*, 103829. [CrossRef]
56. Liu, T.; Sabrina, F.; Jang-Jaccard, J.; Xu, W.; Wei, Y. Artificial intelligence-enabled DDoS detection for blockchain-based smart transport systems. *Sensors* **2021**, *22*, 32. [CrossRef]
57. Awajan, A. A novel deep learning-based intrusion detection system for IOT networks. *Computers* **2023**, *12*, 34. [CrossRef]
58. Dina, A.S.; Siddique, A.; Manivannan, D. A deep learning approach for intrusion detection in Internet of Things using focal loss function. *Internet Things* **2023**, *22*, 100699. [CrossRef]
59. IEEE 802.1 Standards. Available online: https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://www.ieee802.org/1/files/public/docs2019/admin-messenger-TSN-Auto-flyer-2019.pdf&ved=2ahUKEwibv7TwxcyKAxUJVaQEHeA3PRMQFnoECCcQAQ&usg=AOvVaw17qoxFpwSp0brZnpt5Qi2_ (accessed on 7 November 2024).