Received: October 21, 2024. Revised: December 16, 2024.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

A Transformer-Based Intrusion Detection Approach with Fast Gradient Sign Method Adversarial Training

Bashar Alhadad[1]* **Mohsen Nickray[1]**

[1]*Department of Computer and Information Technology, Faculty of Engineering, University of Qom, Qom, Iran*
* Corresponding author's Email: basharcomputer22@gmail.com

Abstract: Intrusion detection systems are crucial for maintaining the security of network infrastructures, yet they remain vulnerable to sophisticated and adversarial attacks. To address this issue, we propose a novel intrusion detection method that combines the transformer architecture with Fast Gradient Sign Method (FGSM) adversarial training. The transformer-based network is specifically designed to process network traffic data, utilizing a transformer encoder with multi-head self-attention mechanisms and position-wise feed-forward layers. The core innovation lies in replacing standard training procedures with FGSM adversarial training, where the model is trained on both clean and adversarial examples. This enhances the system's ability to detect and resist adversarial attacks. The network's performance is evaluated under varying values of the adversarial rate parameter (lambda), with the best results achieved at lambda = 0.5. The model demonstrates a high classification accuracy of 99.7321% on the training data (using the NSL-KDD dataset), 99.603% on the testing data, and 99.4641% on adversarial data. These findings demonstrate the method's robustness and reliability, providing an effective solution for secure and efficient intrusion detection.

Keywords: Intrusion detection system (IDS), Transformer architecture, Adversarial training, Fast gradient sign method (FGSM), Adversarial attacks, Multi-head self-attention.

1. Introduction

In the evolving digital network landscape, Intrusion Detection Systems (IDS) are crucial for preventing unauthorized access and malicious activities. As networks grow more complex, traditional IDS face challenges in accurately detecting threats due to the increasing diversity and volume of network traffic [1]. These systems must discern between legitimate and malicious activities in real-time, but the sophistication of modern cyber threats often exposes vulnerabilities. One major issue is adversarial attacks, where attackers manipulate inputs to trick machine learning models into making incorrect predictions [2]. Such attacks highlight the need for IDS solutions that leverage advanced technologies like transformers while incorporating robust training techniques to defend against both conventional and adversarial attacks.

Multiple studies have proposed methods to enhance IDS performance. In SCADA systems, Reference [3] introduces a stacking ensemble model using a combination of random forest, light boosting gradient machine, and extreme gradient boosting. Although this approach has shown promising results in many scenarios, it may have limitations in processing long-term dependencies in the data. These methods might not be able to effectively model the complex and long-term relationships within network data, which could, in some cases, impact the accuracy and performance of the model in real-world network conditions. In reference [4], an autoencoder-based intrusion detection system (IDS) is used, focusing on the Distributed Network Protocol 3 (DNP3). While this approach may be effective in detecting attacks within the DNP3 protocol, its limitation to this specific protocol could prevent the model from effectively identifying attacks in other protocols or network data types. Furthermore, autoencoder-based models may encounter challenges in processing long-term dependencies and complex network data, and they tend to be more sensitive to noisy data, which

can affect detection accuracy in certain network conditions. In [5], a deep learning-based intrusion detection system (IDS) for SCADA networks compares feedforward neural networks (FNN) and long short-term memory (LSTM) models. The results indicate that LSTM outperforms FNN in detecting both correlated and uncorrelated attacks. However, while LSTM demonstrates its strength in handling temporal sequences, it may face challenges in capturing broader contextual relationships in network traffic data, which are essential for detecting complex attack patterns. Reference [6] introduces a deep reinforcement learning (DRL) framework specifically tailored for SCADA systems. The DRL approach effectively learns optimal defense strategies by interacting with the environment. Nonetheless, such methods may struggle to model long-term dependencies in network data comprehensively, especially when dealing with diverse and large-scale traffic patterns in modern networks. In [7], a hybrid approach is proposed to address class imbalance in IDS datasets by combining the Synthetic Minority Oversampling Technique (SMOTE) with support vector machine (SVM) classifiers. This approach improves the detection of underrepresented classes. However, it may fall short in leveraging contextual information present in network traffic, which could limit its capability to detect subtle and evolving attack scenarios.

In Wireless Sensor Networks (WSNs), several approaches are explored. Reference [8] introduces a machine learning-based intrusion detection system (IDS) that utilizes Gaussian Process Regression (GPR). This method demonstrates strong performance in predicting and detecting anomalous behaviors in network data. Reference [9] employs Support Vector Machine (SVM) and Stochastic Gradient Descent (SGD) with context awareness for intrusion detection. These approaches effectively extract key features from network data. However, both methods discussed in [8] and [9] are often sensitive to noise in the data and can be vulnerable to adversarial attacks, such as input manipulations designed to deceive the model, which may reduce their detection accuracy in real-world conditions. SLGBM, proposed in [10], addresses detection rates and computational overhead in WSNs using the LightGBM algorithm. While this approach effectively balances detection performance and efficiency, it may encounter challenges in capturing long-term dependencies in network traffic. Additionally, like many machine learning-based methods, it could be sensitive to noisy data and adversarial manipulations, which might affect its robustness in dynamic and unpredictable network environments. Reference [11] suggests a real-time IDS for WLANs using a Conditional Deep Belief Network (CDBN) and an instance selection algorithm to handle data redundancy. While this approach effectively reduces data redundancy and balances the dataset for training, it may still face challenges when dealing with noisy data and adversarial attacks aimed at manipulating the detection model. Reference [12] presents a stacked ensemble approach for intrusion detection, outperforming traditional models. It evaluates the system using the NSL-KDD dataset and compares its performance to individual models like ANN, CART, Random Forest, and SVM. Although the combination of different models in this approach has improved accuracy, these models may still face limitations in identifying dynamic changes in network data and unknown attacks.

Research in Software-Defined Networking (SDN) includes a three-tier IDPS for mitigating DDoS attacks [13], a decentralized IDS for large SDN networks [14], and a priority-based anomaly detection model [15]. While the three-tier IDPS in [13] enhances security through layered processing, it may face limitations in scalability and adapting to dynamic attack patterns. Similarly, the decentralized IDS in [14] reduces controller overhead but might encounter challenges in detecting complex attack behaviors and handling adversarial traffic. The priority-based anomaly detection model in [15] improves efficiency for critical applications but may struggle to identify low-frequency anomalies and advanced persistent threats. Additionally, [16] integrates machine learning with SDN for attack detection, and [17] proposes a hybrid feature selection framework using the Whale Optimization Algorithm (WOA) for improved network intrusion detection. While the machine learning-based approach in [16] achieves high accuracy for known attack detection, it may face limitations in processing long-term dependencies in network traffic and adapting to evolving attack patterns. Similarly, the feature selection framework in [17] improves efficiency but might struggle with capturing contextual relationships in complex datasets, which are crucial for identifying sophisticated attacks. For Internet of Things (IoT), several studies propose IDS enhancements. Reference [18] introduces a Network Intrusion Detection System (NIDS) using supervised machine learning on the UNSW-NB15 dataset. While this approach achieves high accuracy in classifying malicious and normal traffic, it may encounter challenges in processing long-term dependencies in network traffic and adapting to evolving attack scenarios. Additionally, [19] employs a fuzzy rough

set-based approach combined with convolutional neural networks (CNN) and GAN for IoT edge computing. Although this method shows promise in feature selection and data augmentation, it might face difficulties in addressing real-time detection requirements and handling adversarial attacks that can manipulate data and deceive the IDS. The Cognitive Memory-guided AutoEncoder (CMAE) for intrusion detection is presented in [20], and [21] evaluates the robustness of intrusion detection systems (IDS) against adversarial examples using the FGSM algorithm. While the CMAE model in [20] offers an innovative approach to memory-based feature extraction, it may face challenges in adapting to evolving attack patterns. Similarly, [21] effectively evaluates robustness against adversarial attacks but does not address sensitivity to noisy data, which could impact the accuracy of the model. Reference [22] addresses dataset imbalance using focal loss for improved detection in IoT environments. While focal loss effectively mitigates class imbalance by emphasizing hard-to-classify instances, this approach may face challenges in adapting to evolving attack patterns and handling noisy or incomplete datasets, which are common in dynamic IoT environments. In the Industrial Internet of Things (IIoT), reference [23] presents the CCSOA-OWKELM technique for feature selection and hyperparameter optimization. While the use of chaotic cuckoo search and sunflower optimization algorithms shows promising results in improving detection accuracy, the model may face challenges in managing adversarial traffic patterns, which are increasingly prevalent in IIoT networks. Additionally, this approach might have limitations in processing complex relationships within data and extracting deep contextual information, which are essential for identifying advanced attacks. An ensemble model combining feature selection with classifiers like XGBoost and Random Forest is proposed in [24]. The study uses the Chi-Square Statistical method for feature selection and applies ensemble classifiers to the ToN-IoT dataset, achieving high accuracy in detecting and classifying IIoT attacks. While this approach demonstrates impressive performance, it may face challenges in processing complex temporal relationships inherent in IIoT traffic data. Additionally, the reliance on specific statistical methods for feature selection might limit its adaptability to dynamically evolving attack scenarios. Reference [25] discusses using graph neural networks (GNN) for IIoT intrusion detection, leveraging their ability to model complex relationships in graph-structured data. While the method effectively captures dependencies among network nodes, it may face challenges in scalability for large-scale IIoT networks and managing adversarial modifications to graph structures, which could reduce its robustness. Similarly, [26] introduces a GNN-based framework to address data imbalance and high feature dimensions. While this approach improves feature representation and classification accuracy, it may face limitations in resilience against evolving attack scenarios. Finally, Reference [27] discusses using Singular Value Decomposition (SVD) and SMOTE for IIoT intrusion detection to address outdated datasets and overfitting challenges. While SVD effectively reduces feature dimensions and SMOTE mitigates dataset imbalance, the approach may face challenges in handling **c**omplex temporal dependencies and adversarial attacks that exploit vulnerabilities in static feature representations, potentially impacting the robustness of the detection system in dynamic IIoT environments. While deep learning models such as CNN and recurrent neural networks (RNN) have enhanced IDS, they struggle with long-range dependencies and are vulnerable to adversarial attacks. Transformers, initially developed for natural language processing (NLP), show potential in addressing these challenges by processing complex data and extracting contextual information. This paper introduces a novel approach to intrusion detection that combines the transformer architecture with FGSM adversarial training. The primary contribution of this work is the design and implementation of a transformer-based model specifically tailored for network traffic analysis, enhanced with adversarial training to improve robustness against attacks. The transformer encoder, augmented by a multi-head self-attention mechanism and position-wise feed-forward layers, provides a sophisticated framework for feature extraction and contextual understanding of network data. Additionally, by integrating FGSM adversarial training, the proposed method trains the model on both clean and adversarial examples, thereby enhancing its resilience and accuracy in detecting and mitigating sophisticated attacks. This approach not only improves the performance of the IDS under normal conditions but also strengthens its defense against adversarial manipulations.

The structure of this paper is organized as follows: Section 2 provides the basic concepts needed for understanding the proposed method. Section 3 details the proposed methodology, including the design of the transformer-based model and the implementation of FGSM adversarial training. Sections 4 and 5 provides necessary information about the dataset and evaluation metrics employed for evaluating the proposed method, respectively.

Section 6 presents the simulation results, including performance evaluations for various lambda values, and results for both normal and adversarial attack detection. A comparison between the proposed approach and other methods in the literature is presented in Section 7. Finally, Section 8 concludes the paper with a discussion on the implications of the results and future directions for research in this domain.

2. Basic concepts

In this section, a detailed overview of the fundamental principles and key concepts required to understand the proposed method is presented.

2.1 Transformer architecture

In recent years, Transformer models, introduced by Vaswani et al. (2017) [28], have become essential in sequence modeling tasks. Built on an encoder-decoder architecture, the encoder processes input sequences into continuous representations, while the decoder generates output sequences [29]. However, in intrusion detection, classification is the focus, so the decoder is discarded, and only the encoder is used. This version captures long-range dependencies using self-attention mechanisms and extracts global features from network traffic data, which is ideal for classification tasks [29]. The encoder consists of six identical layers, each with a multi-head self-attention mechanism and a position-wise fully connected feed-forward network, along with residual connections and layer normalization for stable training. The encoder output passes through a fully connected layer and a Softmax function to classify whether the input represents an intrusion.

1) Patch Embedding Layer

For tasks like network intrusion detection, input data is divided into patches representing different features. The patch embedding layer converts these into fixed-length vectors, allowing the Transformer to process the data uniformly across various input sizes. The mathematical representation of the Patch Embedding process is as follows:

$$LayerNorm(x) = \frac{x-\mu}{\sigma} \cdot \gamma + \beta \quad (1)$$

2) Position Embedding Layer

Unlike RNNs, Transformers lack a built-in mechanism to maintain sequence order. To resolve this, the position embedding layer is introduced, adding unique positional encodings to each input token or patch.

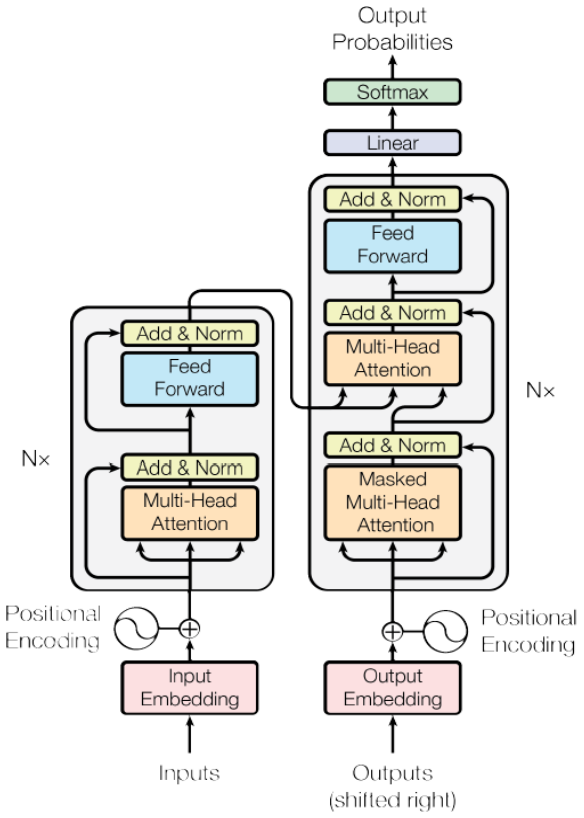

Figure. 1 The transformer model architecture [28]

In network intrusion detection, this ensures that the sequence order, such as packet sequences, is preserved during processing. By combining position embeddings with patch embeddings, the model captures both the content and the relative position of each element in the data. The combined input to the Transformer is given by:

$$X_{pos} = X_{emb} + P \quad (2)$$

3) Multi-Head Self-Attention Layer

Self-attention enables the model to focus on various parts of the sequence when representing a token, while multi-head attention allows the model to attend to multiple positions simultaneously through several attention heads. This mechanism is highly effective for intrusion detection, as it captures complex patterns and correlations in network traffic data without the limitations of traditional convolutional or recurrent architectures. Each attention head computes a weighted sum of input elements, allowing the model to focus on different aspects of the input at the same time.

Self-Attention:

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (3)$$

Multi-Head Attention:

$$MultiHead(Q,K,V) = Contact(head_1,\dots,head_h)W^O \quad (4)$$

where each $head_i = Attention\left(Q.W_i^Q, K.W_i^K, V*W_i^V\right)$

4) Point-Wise Fully Connected Layer

After self-attention, the output goes through two linear transformations that expand and then reduce its dimensionality, helping capture complex feature relationships. A non-linear activation adds flexibility, allowing the model to learn more intricate patterns. This process refines the features for better accuracy in intrusion detection. The transformation in the Point-Wise Fully Connected layer is described as:

$$FFN(x) = f(xW_1 + b_1)W_2 + b_2 \quad (5)$$

5) Layer Normalization

This technique normalizes inputs after each sub-layer to stabilize and speed up training, preventing issues like vanishing or exploding gradients. It ensures balanced outputs, which enhances the model's learning, particularly in complex intrusion detection tasks. The Layer Normalization process can be described by:

$$LayerNorm(x) = \frac{x-\mu}{\sigma}\cdot\gamma + \beta \quad (6)$$

6) Softmax Layer

The Softmax layer converts the model's raw output into a probability distribution, with values between 0 and 1 that sum to 1. It allows the model to classify input data as normal traffic or an intrusion by selecting the class with the highest probability. This makes the results interpretable and helps assess the likelihood of different intrusion types. The Softmax function is represented as:

$$Softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \quad (7)$$

2.2 Fast gradient sign method adversarial training

Fast Gradient Sign Method (FGSM) is a foundational approach in adversarial training that seeks to improve the robustness of machine learning models against adversarial attacks. These attacks involve slightly perturbing an input sample in a way that causes a model to misclassify it, even though the perturbed input remains nearly indistinguishable from the original [30]. FGSM, introduced by Goodfellow [31], is a single-step method for generating adversarial examples by making use of the gradient of the model's loss function with respect to the input.

Consider a classification task over pairs of samples (x, y), where x represents an input and y the corresponding label. Given a model $f_\theta(x)$parameterized by θ and a loss function L, the goal of adversarial training is to train the model to be robust to adversarial examples, x′, which are perturbed versions of x. The adversarial example satisfies $D\ (x,x') < \epsilon\ D\ (x,x')$ for some small $\epsilon > 0$, where D is a distance metric, commonly the lp norm. In the case of FGSM adversarial training, the $l1$ norm is often used to measure the distance between x and x′, and the aim is to ensure that the model does not misclassify x′.

The core idea of FGSM adversarial training is to train the model on adversarial generated samples, instead of purely natural examples. These adversarial examples are crafted by adding perturbations to the input data based on the gradient of the loss function. The FGSM attack is computed as follows [31]:

$$\acute{x} = x + \epsilon\ .sign\ (\nabla_x L(f_\theta(x), y)) \quad (8)$$

The sign of the gradient is taken to ensure that the perturbation increases the loss function, thereby forcing the model to make an incorrect prediction.

3. Methodology

In this section, the methodology used to develop the proposed intrusion detection system is explained in detail, combining the transformer architecture with adversarial training. The first significant contribution of this method lies in the design of the network architecture, with the transformer encoder forming the core. The original transformer architecture, typically used for sequence transduction, consists of both an encoder and a decoder. The encoder processes input data by capturing context and extracting relevant features, while the decoder uses these features to generate sequential outputs. However, for the classification task in this paper, only feature extraction is needed. As a result, the decoder is discarded, and the encoder is expanded to enhance performance.

The proposed architecture begins with an essential embedding layer, which is a fundamental component for transformers. However, in some cases, the input data may lack sufficient features for an effective embedding in a deep network like a transformer. One option is to manually extract

additional features by combining input data to create a larger feature space. However, handcrafted features may not always be ideal, making automatic feature extraction through the network preferable. To address this, the input data is projected into a 1x4096 vector using a feed-forward layer and then reshaped to a 64x64 matrix. This feed-forward layer automatically extracts features based on the weights and biases learned during training, while reshaping enhances flexibility for further processing.

Next, the projected values are embedded using a patch embedding layer with a patch size of 4x4 and an embedding dimension of 100. These embedded values are then fed into the transformer encoder. Since transformers lack an inherent mechanism to account for the order of inputs, a positional encoding layer is added after the embedding step. Positional encoding provides information about the relative or absolute position of elements within the sequence. By summing the embedding vectors with the corresponding positional encoding vectors, the transformer receives a combined representation of both content and positional information.

The processed data is then passed to the transformer encoder, which consists of two main layers, each containing two sub-layers. The first sub-layer is a multi-head self-attention mechanism with 4 heads and 12 key channels. This mechanism enhances the processing of data by capturing relationships between different feature patches, allowing the model to focus on multiple aspects of the extracted features simultaneously for more effective representation. Each attention head operates in parallel, attending to different aspects of the input, such as short-range dependencies and long-range contextual information.

The second sub-layer is a position-wise fully connected feed-forward layer with a hidden size of 150. This sub-layer is composed of two dense layers, separated by a non-linear activation function. Specifically, the Gaussian Error Linear Unit (GeLU) activation function is used to improve network performance. GeLU's probabilistic nature makes it particularly effective for capturing complex patterns in deep networks, which is crucial for transformer architectures.

Additionally, residual connections are incorporated into both the multi-head self-attention mechanism and the position-wise feed-forward layers. These connections allow information to flow directly from the input to the output of each layer, bypassing intermediate layers. This design helps prevent the vanishing gradient problem, which is critical for training deeper networks effectively. By directly adding the input to the output, residual connections ensure that important information is preserved throughout the network, allowing the model to build upon knowledge from previous layers without losing critical insights during processing.

After each layer, a dropout layer with a probability of 0.1 is applied, followed by layer normalization to enhance the training process. These additions help prevent overfitting by randomly deactivating certain neurons during training, which improves generalization. Layer normalization ensures stable and efficient learning by normalizing the outputs within each layer, promoting faster convergence and more robust model performance.

The network concludes with indexing, fully connected layers, and a SoftMax activation function for the classification of the input data. This final configuration results in a transformer encoder with approximately 7.9 million parameters. Fig. 2 illustrates the complete architecture of the designed network.

The second major contribution of this paper is the replacement of the standard training procedure with Fast Gradient Sign Method (FGSM) Adversarial Training. FGSM adversarial training is a technique designed to make deep learning models more robust against adversarial attacks—attacks that involve subtle manipulations of input data to deceive the model into making incorrect predictions. The key idea behind this method is to incorporate adversarial examples generated using FGSM during training. By exposing the model to both clean and adversarial data, it learns to resist such attacks, thus increasing its resilience.

To implement adversarial training, adversarial examples are generated during each iteration using FGSM. Given an input x, label y, and a model with parameters θ, the adversarial example x_{adv} is computed as:

$$x_{adv} = x + \begin{cases} \min\left(\cdot\, sign\left(\nabla_x J(\theta, x, y)\right), \alpha\right), & \nabla_x J(\theta, x, y) > 0 \\ \max\left(\cdot\, sign\left(\nabla_x J(\theta, x, y)\right), -\alpha\right), & \nabla_x J(\theta, x, y) \le 0 \end{cases} \tag{9}$$

$J(\theta, x, y)$ is the cross-entropy loss in this paper. The $\cdot$ and α are considered equal to the 0.02 and 0.015, respectively. These adjustments are done based on the dispersion of the data in different features.

The model should be trained on both the original clean examples and the adversarial examples generated by FGSM. This dual training helps the model learn to handle both types of data. Thus, the model's parameters θ are updated based on the loss

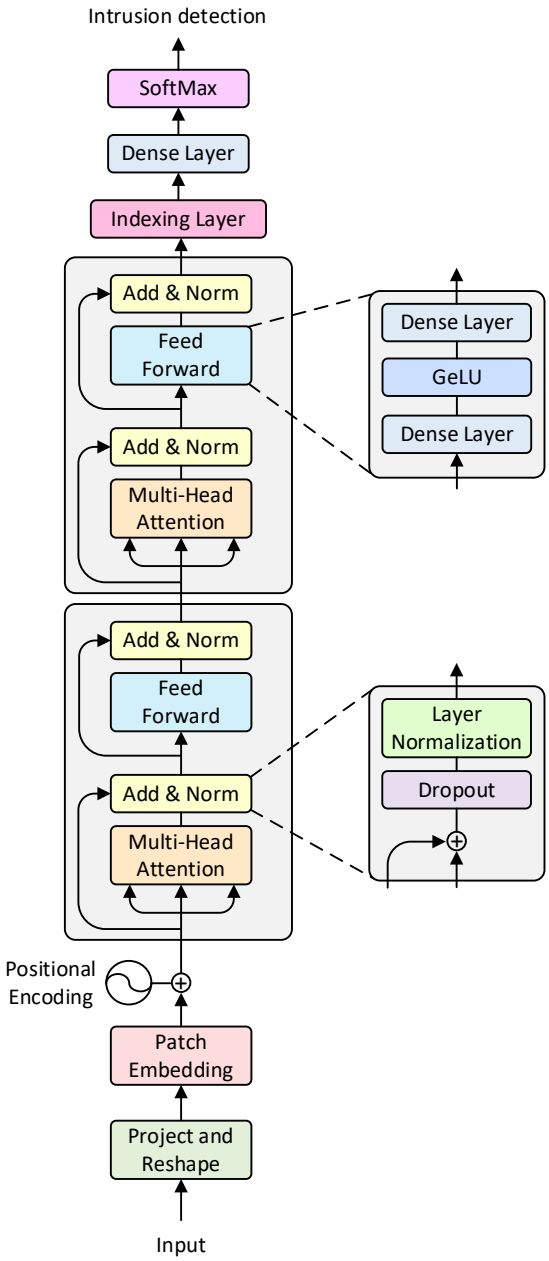

Figure. 2 The proposed network structure

computed from both clean and adversarial examples. The goal is to minimize the loss for both types of inputs, making the model more robust.

The overall objective of FGSM adversarial training is to minimize the loss function on both clean and adversarial examples. The loss function can be expressed as:

$$L_{total} = (1-\lambda)L_{clean}(x,y) + \lambda L_{adv}(x_{adv},y) \quad (10)$$

λ is the adversarial rate parameter making a trade-off between the importance of clean examples and adversarial examples for training the network.

Finally, the new parameters of the network θ are obtained using the ADAM optimizer as follows:

$$\theta = \theta - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\varepsilon\epsilon} \quad (11)$$

Both $\hat{m}_t$ and $\hat{v}_t$ are obtained using the gradient of the total loss (L_{total}) with respect to the model parameters. To calculate these parameters, the gradient decay factor is considered equal to 0.9 and squared gradient decay factor is considered 0,999. Moreover, η is the learning rate and ε is a small constant added for numerical stability.

4. Dataset

In this paper, we utilize the NSL-KDD dataset, which is an improved version of the original KDD Cup 1999 dataset [32], widely used for evaluating network intrusion detection systems. The NSL-KDD dataset addresses several issues in the original KDD dataset, such as the elimination of redundant records and providing a more balanced and reliable dataset for classification tasks. It consists of a wide variety of simulated network intrusions and normal activities, making it suitable for training and testing machine learning models for anomaly detection [33].

The dataset includes both training and testing sets, each containing a mixture of normal and malicious network traffic. The training set consists of 125,973 instances, while the testing set contains 22,544 instances. Each instance in the dataset is characterized by 41 features, categorized into three types: basic features, content features, and traffic features, which capture various aspects of network traffic behavior. Additionally, each instance is labeled as either "normal" or belonging to one of several attack types, which are grouped into four main categories: Denial of Service (DoS), Probe, User-to-Root (U2R), and Remote-to-Local (R2L) attacks. We leverage this dataset to evaluate the performance of our proposed intrusion detection model. By utilizing the NSL-KDD dataset, we aim to ensure that our model is trained and tested on a representative and balanced dataset, allowing for a comprehensive assessment of its ability to detect both known and unknown network intrusions [34].

In this paper, the NSL-KDD dataset features are normalized between 0 and 1 first. Then, the dataset is balanced by undersampling.

5. Evaluation metrics

In this paper, we use four evaluation metrics (accuracy, precision, recall, and F1 score) to evaluate the performance of our model. These metrics are crucial for providing a comprehensive assessment of how well the model detects network intrusions and

classifies normal traffic. The details of each of these metrics are given as follows:

• **Accuracy**: Accuracy measures the overall performance by calculating the proportion of correctly classified instances out of the total number of instances. It gives an overview of how well the model is performing across both positive and negative classes. Accuracy is calculated using Eq. (12):

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{12}$$

Where TP is True Positives which refers to correctly identified attacks, TN is True Negatives which refers to correctly identified normal traffic, FP is False Positives which indicates normal traffic misclassified as attacks, and FN is False Negatives which indicates attacks misclassified as normal traffic.

• **Precision**: Precision evaluates the model's ability to accurately identify positive instances (attacks). It is defined as the ratio of true positives to the total number of predicted positives, including false positives. This metric reflects the model's accuracy in its positive predictions and is given by Eq. (13):

$$Precision = \frac{TP}{TP+FP} \tag{13}$$

• **Recall**: Recall, also known as sensitivity or true positive rate, measures the model's ability to capture all relevant positive instances (attacks). It is calculated as the ratio of true positives to the total actual positives, ensuring that the model correctly identifies as many positive cases as possible. The calculation of recall is as given in Eq. (14):

$$Recall = \frac{TP}{TP+FN} \tag{14}$$

F1 Score: The F1 score is the harmonic mean of precision and recall, providing a single metric that balances the trade-off between false positives and false negatives. It is particularly useful when dealing with imbalanced datasets where one class might dominate. The F1 score is calculated using Eq. (15):

$$F_1 Score = \frac{2\times(Precision\times Recall)}{Precision+Recall} \tag{15}$$

6. Simulation results

This section presents simulation results for the proposed intrusion detection system, focusing on the effectiveness of the transformer-based model and the influence of FGSM adversarial training on detection accuracy. Evaluations include adjusting the lambda parameter and analyzing performance metrics for normal attack detection and resilience against adversarial attacks. The simulations were conducted using MATLAB 2024a on a system with an Intel Core i7 13650HX CPU, 16 GB RAM, and an NVIDIA RTX 4060 GPU.

6.1 Lambda parameter adjustment

In this section, we analyze the effect of varying the λ parameter on the performance of the proposed intrusion detection model using FGSM adversarial training. The parameter λ controls the trade-off between clean and adversarial data in the training process, and its impact on accuracy is evaluated across three different data sets: training data, testing data, and adversarial data. The values of λ considered are 0 (normal training), 0.25, 0.5, and 0.75. Fig. 3 presents the results for each data set, showing how the accuracy changes with different λ values.

As depicted in Fig. 3, the performance of the model on training data remains consistently high across all λ values. For λ = 0 (normal training), the accuracy is 99.7519%. As the adversarial component is introduced with λ = 0.25, the accuracy increases slightly to 99.8115%. However, further increases in λ lead to slight reductions in accuracy, with 99.7321% for λ = 0.5 and 99.474% for λ = 0.75. This trend indicates that while introducing adversarial training improves the robustness of the model, a high emphasis on adversarial examples can slightly affect performance on clean training data.

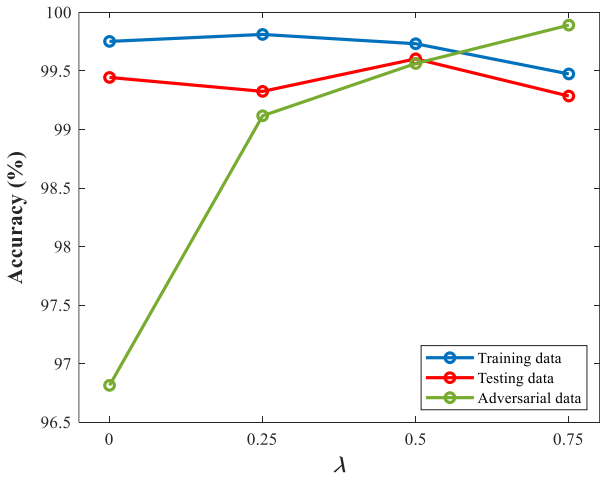

Figure. 3 Performance evaluation for different λ values

For the testing data, the normal training (with $\lambda = 0$) achieves an accuracy of 99.4442%. Interestingly, when $\lambda = 0.5$, the model reaches its highest accuracy on the testing data, with 99.603%, outperforming both normal training and lower values of λ. At $\lambda = 0.25$, the accuracy is 99.3251%, and at $\lambda = 0.75$, it decreases to 99.2854%. These results suggest that moderate values of λ (particularly 0.5) strike an effective balance between clean and adversarial data for testing scenarios, leading to better generalization.

The most significant results appear when analyzing the model's performance on adversarial data. As expected, the network trained with $\lambda = 0$ (normal training) performs poorly under adversarial attacks, with an accuracy of 96.8145%. However, as adversarial training is introduced with $\lambda = 0.25$, the model's accuracy increases significantly to 99.1168%. This trend continues, with $\lambda = 0.5$ achieving 99.5633%, and $\lambda = 0.75$ yielding the highest accuracy of 99.8908%. These results clearly demonstrate the effectiveness of FGSM adversarial training in improving the model's robustness against adversarial attacks.

In summary, the results suggest that while a moderate adversarial rate ($\lambda = 0.5$) offers the best trade-off between clean and adversarial performance, higher values of λ ($\lambda = 0.75$) provide optimal protection against adversarial attacks at the cost of slightly reduced performance on clean data. Since the obtained results of the trained network using $\lambda = 0.5$ shows a great balance, it is used to evaluate proposed approach's performance in the rest of this paper.

6.2 Normal attack detection results

In this section, we present the performance of the proposed intrusion detection model in identifying normal network attacks using a confusion matrix and Receiver Operating Characteristic (ROC) curve.

A confusion matrix is a useful tool for visualizing the performance of a classification model by comparing the predicted labels against the true labels. It provides detailed information about the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), which helps in understanding the model's accuracy, precision, recall, and overall reliability. The diagonal elements of the confusion matrix represent correct predictions, while the off-diagonal elements represent misclassifications. Specifically, true positives refer to attack instances that were correctly identified, while true negatives represent non-attack instances that were correctly classified. On the other hand, false positives occur when non-attack instances are incorrectly predicted as attacks, and false negatives occur when attack instances are misclassified as non-attacks.

Fig. 4 shows the confusion matrix for the model's performance on normal attack detection. The results indicate that the model correctly identified 1,347 attack samples (true positives) and 1,162 non-attack samples (true negatives). In contrast, there were only 6 false positives and 4 false negatives, showcasing the model's high precision and recall. These results demonstrate that the proposed model performs exceptionally well in distinguishing between normal network activity and attack instances.

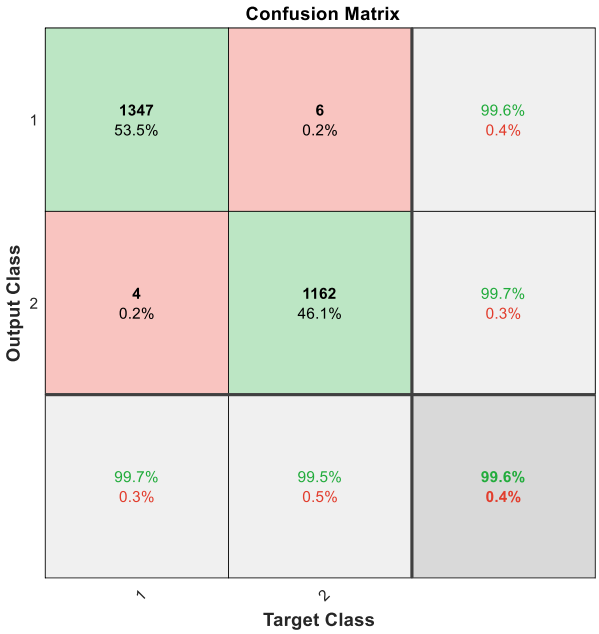

Figure. 4 The confusion matrix of evaluating the testing dataset using the proposed approach

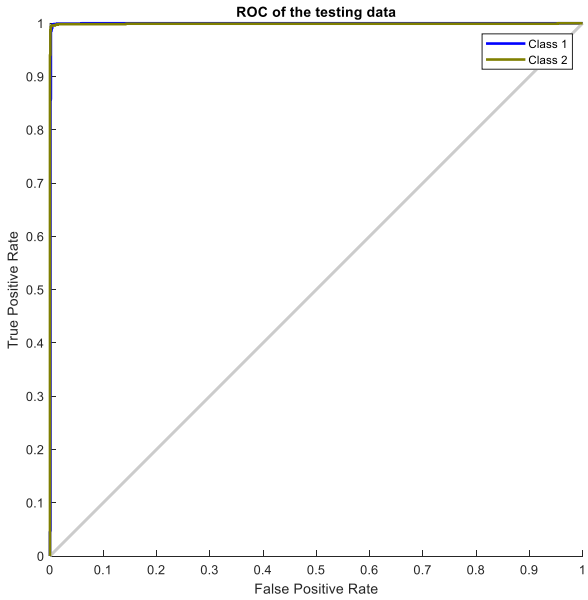

Figure. 5 The ROC curve of evaluating the testing dataset using the proposed approach

The minimal number of false positives and false negatives further suggests that the model can reliably detect attacks while maintaining a low error rate in classifying non-attack samples.

The Receiver Operating Characteristic (ROC) curve is a key metric for evaluating a model's classification performance. It plots the true positive rate (sensitivity) against the false positive rate (1-specificity) at various decision thresholds. The Area Under the Curve (AUC) quantifies the model's ability to distinguish between classes, with values close to 1 indicating high effectiveness and values near 0.5 suggesting poor performance. The ROC curve visually demonstrates model performance across thresholds, making it especially valuable for imbalanced datasets where one class is more prevalent than the other.

Fig. 5 shows the ROC curve for the proposed model's performance in detecting normal attacks. The AUC obtained for this curve is 0.99985, which is almost the maximum possible value, signifying near-perfect classification ability. This extremely high AUC demonstrates that the model can accurately detect attack instances while minimizing false positives. The curve's proximity to the top-left corner of the plot indicates that the model achieves a high true positive rate while maintaining a low false positive rate, further confirming its robustness and effectiveness in normal attack detection scenarios.

6.3 Adversarial attack detection results

In this section, the results for adversarial attack detection are analyzed and compared with those for clean data to evaluate the model's robustness. Fig. 6 displays the confusion matrix for the adversarial attack detection. The model shows strong performance with 1358 true positive samples and 1150 true negative samples, alongside only 3 false positives and 8 false negatives. The model achieves an accuracy of 99.56% on adversarial data, which is slightly lower compared to the 99.60% accuracy obtained on clean testing data. This slight difference indicates the effectiveness of the adversarial training procedure, as the model manages to maintain high performance even under adversarial attacks. When compared to the results from the previous section, where the normal attack detection confusion matrix was analyzed, it can be observed that the model shows a small increase in false negatives in adversarial attack detection, from 4 to 8, while false positives decreased from 6 to 3. Despite the adversarial nature of the input data, the model remains resilient, as demonstrated by its consistently high accuracy and classification ability.

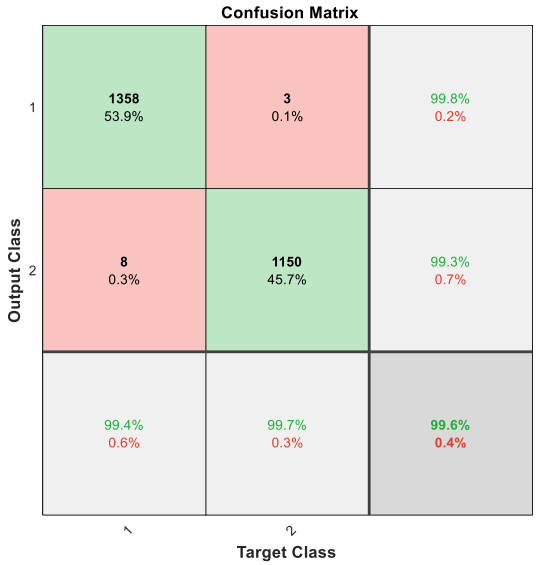

Figure. 6 The confusion matrix of evaluating the adversarial dataset using the proposed approach

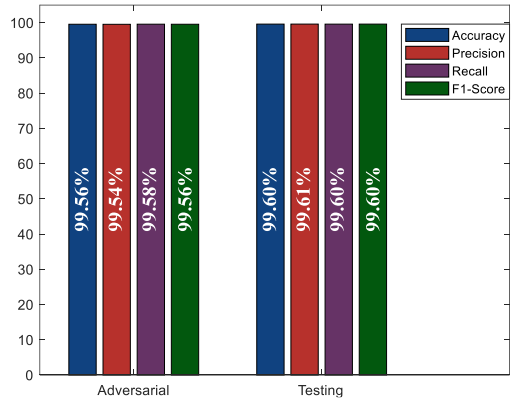

Figure. 7 The evaluation metrics bar chart of evaluating both clean and adversarial datasets using the proposed approach

Table 1. The assessment results of the proposed method using the evaluation metrics

	Accuracy	Precision	Recall	F1-Score
Adversarial	99.56%	99.54%	99.58%	99.56%
Normal Testing	99.60%	99.61%	99.60%	99.60%

Fig. 7 presents a comparison of the model's performance metrics (accuracy, precision, recall, and F1 score) on both adversarial and clean testing data. For adversarial data, the model achieved an accuracy of 99.56%, a precision of 99.54%, a recall of 99.58%, and an F1 score of 99.56%. When compared to the performance on clean data, where the model attained an accuracy of 99.60%, a precision of 99.61%, a recall of 99.60%, and an F1 score of 99.60%, the results indicate a slight decrease in all metrics for adversarial data. Despite this minor reduction, the

model maintains exceptional performance across all metrics, with differences in accuracy, precision, recall, and F1 score being less than 0.1%. This marginal drop suggests that the adversarial training strategy employed, particularly through FGSM adversarial training, has effectively enhanced the model's resilience to adversarial attacks. These findings confirm that the model can perform comparably well on both adversarial and clean data, making it a robust solution for real-world intrusion detection scenarios where adversarial examples may be present. Table 1 illustrates the summary of assessing the proposed method using the evaluation metrics.

7. Comparison

In this section, we compare the proposed method with state-of-the-art approaches in terms of performance metrics including accuracy, precision, recall, F1-score, and adversarial robustness.

Table 2. The comparison of the proposed method with other intrusion detection approaches

Ref.	Method	Dataset	Accuracy (%)	Precision (%)	Recall (%)	F1Score (%)	Able to handle adversarial attacks
[35]	Logistic regression	NSL-KDD	96.6	97	97	97	No
	K-nearest neighbor		95.5	96	95	96	No
	Random forest		95.7	96	96	96	No
	Decision tree		96.7	97	97	97	No
	Multi-layer perceptron (MLP)		97.8	97	98	98	No
	Long-short term memory (LSTM)		97.5	97	97	97	No
[36]	Artificial neural network (MLP)	NSL-KDD	97.5	99	96.7	95.7	No
[37]	XGBoost feature selection + recurrent neural network	NSL-KDD	83.70	N.M	N.M	N.M	No
	XGBoost feature selection + LSTM		88.13	N.M	N.M	N.M	No
	XGBoost feature selection + Gated recurrent unit		84.66	N.M	N.M	N.M	No
[38]	Adaptive boosting	NSL-KDD	92.20	N.M	N.M	91.70	No
	Random forest		99.70	N.M	N.M	99.70	No
	J84 classifier		99.60	N.M	N.M	99.60	No
	Naïve Bayse		86.60	N.M	N.M	84.20	No
Proposed method	Transformer network trained with FGSM adversarial training	NSL-KDD	99.60	99.61	99.60	99.60	Yes

*N.M: not mentioned

The comparison results reported in Table 2 provide insights into the strengths and limitations of each method.

The proposed method demonstrates a competitive performance across all metrics. In terms of accuracy, the Random Forest classifier in [38] achieves the highest value at 99.7%, slightly outperforming the proposed method, which achieves 99.6%. However, this marginal difference in accuracy is outweighed by the proposed method's ability to handle adversarial attacks, a critical advantage. Unlike [38], which does not explicitly address adversarial attacks, the proposed method maintains high accuracy and resilience when subjected to adversarial conditions, making it more reliable for real-world deployment. Other methods, such as those in [35] and [36], range between 95.5% and 97.8% in accuracy, showing that the proposed method stands on par with or outperforms many established models. Reference [37], which employs an XGBoost-based feature selection algorithm with recurrent neural networks, achieves lower accuracy compared to the proposed method, with a best performance of 88.13%.

When comparing precision, recall, and F1-score, the proposed method again excels. It achieves an impressive precision of 99.61%, recall of 99.60%, and F1-score of 99.60%, indicating a highly balanced and effective intrusion detection system. In contrast, the methods in [35] and [36] which achieved strong accuracy (ranging from 95.5% to 97.8%), show the same (and lower in some cases) recall or precision. For example, [36] achieves precision of 99%, but with a recall of 96.7%, indicating potential for missed detections. The method in [38] achieves high F1-score and accuracy, but the precision and recall are not reported to evaluate the balance performance like that the proposed method offers. Furthermore, the lack of detailed precision and recall metrics in [37] makes it difficult to fully assess its performance in this regard, though its accuracy of 88.13% falls behind the proposed method.

Overall, while the Random Forest classifier in [38] shows slightly better accuracy and F1-score, the proposed method offers a more comprehensive and resilient solution excelling in adversarial robustness while having almost the same accuracy as the reported one in [38].

8. Conclusion

In this paper, we proposed a novel intrusion detection system that combines the transformer architecture with Fast Gradient Sign Method (FGSM) adversarial training to enhance both accuracy and robustness against adversarial attacks. The core of the proposed method lies in the design of a transformer encoder, which efficiently processes network traffic data. The transformer encoder was modified by expanding its original structure, removing the decoder, and replacing embedding input data with a fully connected-based feature extractor and a patch embedding for better feature extraction. The transformer encoder is composed of two layers, each featuring a multi-head self-attention mechanism and a fully connected feed-forward layer. To prevent overfitting and ensure efficient training, dropout and normalization techniques were employed. Moreover, the use of residual connections in the network allowed for smoother information flow and improved the learning of complex data patterns.

The second key contribution of this work is the incorporation of FGSM adversarial training, which enhances the network's resistance to adversarial attacks. During training, adversarial examples were generated by introducing perturbations in the input data. The model was trained on both clean and adversarial data, allowing it to learn to identify adversarial patterns and become more resilient. The lambda parameter was introduced to balance the emphasis on clean versus adversarial data, and through experiments, a lambda value of 0.5 was found to provide the best trade-off.

The results demonstrated that the proposed approach outperforms conventional training methods in both normal and adversarial detection scenarios. With lambda = 0.5, the model achieved a detection accuracy of 99.60% on clean data and 99.56% on adversarial data. These results underscore the effectiveness of the transformer-based model in identifying complex network attacks while maintaining robust performance against adversarial examples. Future research will explore further enhancements, such as optimizing adversarial defense techniques and refining transformer architectures for even greater robustness.

Nomenclature:

x	The input patch matrix
W	The weight matrix
b	The bias term
P	The position embedding matrix
Q	The Query matrices
K	The Key matrices
V	The Value matrices
d_k	The dimension of the keys
W_i^Q	The learned weight matrices for queries
W_i^K	The learned weight matrices for keys

W_i^V	The learned weight matrices for values
σ	The standard deviation of the inputs
c	The learned parameters
β	The learned parameters
z_i	The raw score (logit) for class i
N	The total number of classes
$\nabla_x L(f_\theta(x), y)$	The gradient of the loss function
ϵ	Controls the magnitude of the perturbation
x_{adv}	the adversarial example
$J(\theta, x, y)$	The gradient of the loss function with respect to learnable parameters
$\nabla_x J(\theta, x, y)$	The gradient of the loss with respect to the input x
ϵ	A small constant controlling the magnitude of the perturbation
α	The boundary parameter that controls and limits the perturbation of the adversarial data
$L_{\text{clean}}(x, y)$	The loss on the clean example x
W_i^O	The learned weight matrices for output
μ	The mean of the inputs
$\hat{m}_t$	The bias-corrected first-moment estimate
η	The learning rate
FP	False Positives
FN	False Negatives
$L_{\text{adv}}(x_{\text{adv}}, y)$	The loss on the adversarial example x_{adv}
λ	The adversarial rate parameter
$\hat{v}_t$	The bias bias-corrected second-moment estimate
TP	True Positives
TN	True Negatives

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization, Bashar Alhadad and Mohsen Nickray; methodology, Bashar Alhadad and Mohsen Nickray; software, Bashar Alhadad; validation, Bashar Alhadad; formal analysis, Mohsen Nickray; investigation, Bashar Alhadad; resources, Bashar Alhadad; data curation, Bashar Alhadad; writing—original draft preparation, Bashar Alhadad; writing—review and editing, Bashar Alhadad and Mohsen Nickray; visualization, Bashar Alhadad; supervision, Mohsen Nickray.

References

[1] A. A. Ghorbani, W. Lu & M. Tavallaee. "Network intrusion detection and prevention: concepts and techniques", *Springer Science & Business Media*, Vol. 47, 2009.

[2] H. Qiu, T. Dong, T. Zhang, J. Lu, G. Memmi, & M. Qiu. "Adversarial attacks against network intrusion detection in IoT systems", *IEEE Internet of Things Journal*, Vol. 8, No. 13, pp. 10327-10335, 2020.

[3] D. D. Nguyen, M. T. Le, & T. L. Cung. "Improving intrusion detection in SCADA systems using stacking ensemble of tree-based models", *Bulletin of Electrical Engineering and Informatics*, Vol. 11, No. 1, pp. 119-127, 2022.

[4] M. Altaha, J. M. Lee, M. Aslam, and S. Hong, "An Autoencoder-Based Network Intrusion Detection System for the SCADA System", *Journal of Communications*, Vol. 16, No. 6, pp. 210-216, 2021.

[5] J. Gao, L. Gan, F. Buschendorf, L. Zhang, H. Liu, P. Li, and T. Lu, "Omni SCADA intrusion detection using deep learning algorithms", *IEEE Internet of Things Journal*, Vol. 8, No. 2, pp. 951-961, 2020.

[6] F. Mesadieu, D. Torre, and A. Chennameneni, "Leveraging Deep Reinforcement Learning Technique for Intrusion Detection in SCADA Infrastructure", *IEEE Access*, 2024.

[7] M. Mahalakshmi, M. P. Ramkumar, and E. S. GSR, "SCADA intrusion detection system using cost sensitive machine learning and SMOTE-SVM", In: *Proc. of 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pp. 332-337, 2022.

[8] A. Singh, J. Nagar, S. Sharma, and V. Kotiyal, "A Gaussian process regression approach to predict the k-barrier coverage probability for intrusion detection in wireless sensor networks", *Expert Systems with Applications*, Vol. 172, p. 114603, 2021.

[9] O. Ahmed, "Enhancing Intrusion Detection in Wireless Sensor Networks through Machine Learning Techniques and Context Awareness Integration", *International Journal of Mathematics, Statistics, and Computer Science*, Vol. 2, pp. 244-258, 2024.

[10] S. Jiang, J. Zhao, and X. Xu, "SLGBM: An intrusion detection mechanism for wireless

sensor networks in smart environments", *IEEE Access*, Vol. 8, pp. 169548-169558, 2020.

[11] L. Yang, J. Li, L. Yin, Z. Sun, Y. Zhao, and Z. Li, "Real-time intrusion detection in wireless network: A deep learning-based intelligent mechanism", *IEEE Access*, Vol. 8, pp. 170128-170139, 2020.

[12] H. Rajadurai and U. D. Gandhi, "A stacked ensemble learning model for intrusion detection in wireless network", *Neural Computing and Applications*, Vol. 34, No. 18, pp. 15387-15395, 2022.

[13] A. Ali and M. M. Yousaf, "Novel three-tier intrusion detection and prevention system in software defined network", *IEEE Access*, Vol. 8, pp. 109662-109676, 2020.

[14] A. H. Janabi, T. Kanakis, and M. Johnson, "Overhead reduction technique for software-defined network based intrusion detection systems", *IEEE Access*, Vol. 10, pp. 66481-66491, 2022.

[15] N. Satheesh, M. V. Rathnamma, G. Rajeshkumar, P. V. Sagar, P. Dadheech, S. R. Dogiwal, and S. Sengan, "Flow-based anomaly intrusion detection using machine learning model with software defined networking for OpenFlow network", *Microprocessors and Microsystems*, Vol. 79, p. 103285, 2020.

[16] O. J. Ibrahim and W. S. Bhaya, "Intrusion detection system for cloud based software-defined networks", In: *Proc. of Journal of Physics: Conference Series*, Vol. 1804, No. 1, p. 012007, IOP Publishing, 2021.

[17] C. Kumar, S. Biswas, M. S. A. Ansari, and M. C. Govil, "Nature-inspired intrusion detection system for protecting software-defined networks controller", *Computers & Security*, Vol. 134, p. 103438, 2023.

[18] M. Ahmad, Q. Riaz, M. Zeeshan, H. Tahir, S. A. Haider, and M. S. Khan, "Intrusion detection in internet of things using supervised machine learning based on application and transport layer features using UNSW-NB15 data-set", *EURASIP Journal on Wireless Communications and Networking*, Vol. 2021, No. 1, pp. 1-23, 2021.

[19] Y. Wu, L. Nie, S. Wang, Z. Ning, and S. Li, "Intelligent intrusion detection for internet of things security: A deep convolutional generative adversarial network-enabled approach", *IEEE Internet of Things Journal*, Vol. 10, No. 4, pp. 3094-3106, 2021.

[20] H. Lu, T. Wang, X. Xu, and T. Wang, "Cognitive memory-guided autoencoder for effective intrusion detection in internet of things", *IEEE Transactions on Industrial Informatics*, Vol. 18, No. 5, pp. 3358-3366, 2021.

[21] X. Fu, N. Zhou, L. Jiao, H. Li, and J. Zhang, "The robust deep learning–based schemes for intrusion detection in internet of things environments", *Annals of Telecommunications*, Vol. 76, No. 5, pp. 273-285, 2021.

[22] A. S. Dina, A. B. Siddique, and D. Manivannan, "A deep learning approach for intrusion detection in Internet of Things using focal loss function", *Internet of Things*, Vol. 22, p. 100699, 2023.

[23] R. Gopi, R. Sheeba, K. Anguraj, T. Chelladurai, H. M. Alshahrani, N. Nemri, and T. Lamoudan, "Intelligent Intrusion Detection System for Industrial Internet of Things Environment", *Computer Systems Science & Engineering*, Vol. 44, No. 2, 2023.

[24] J. B. Awotunde, S. O. Folorunso, A. L. Imoize, J. O. Odunuga, C. C. Lee, C. T. Li, and D. T. Do, "An ensemble tree-based model for intrusion detection in industrial internet of things networks", *Applied Sciences*, Vol. 13, No. 4, p. 2479, 2023.

[25] J. Long, W. Liang, K. C. Li, Y. Wei, and M. D. Marino, "A regularized cross-layer ladder network for intrusion detection in industrial internet of things", *IEEE Transactions on Industrial Informatics*, Vol. 19, No. 2, pp. 1747-1755, 2022.

[26] Y. Zhang, C. Yang, K. Huang, and Y. Li, "Intrusion detection of industrial internet-of-things based on reconstructed graph neural networks", *IEEE Transactions on Network Science and Engineering*, Vol. 10, No. 5, pp. 2894-2905, 2022.

[27] S. Soliman, W. Oudah, and A. Aljuhani, "Deep learning-based intrusion detection approach for securing industrial Internet of Things", *Alexandria Engineering Journal*, Vol. 81, pp. 371-383, 2023.

[28] A. Vaswani, "Attention is all you need", *Advances in Neural Information Processing Systems*, 2017.

[29] Z. Long, H. Yan, G. Shen, X. Zhang, H. He, and L. Cheng, "A Transformer-based network intrusion detection approach for cloud security", *Journal of Cloud Computing*, Vol. 13, No. 1, p. 5, 2024.

[30] B. Li, S. Wang, S. Jana, and L. Carin, "Towards understanding fast adversarial training", *arXiv:2006.03089*, 2020.

[31] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples", *arXiv:1412.6572*, 2014.

[32] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set", In: *Proc. of 2009 IEEE symposium on computational intelligence for security and defense applications*, pp. 1-6, 2009.

[33] L. Dhanabal and S. P. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms", *International Journal of Advanced Research in Computer and Communication Engineering*, Vol. 4, No. 6, pp. 446-452, 2015.

[34] NSL-KDD dataset. Retrieved [09/01/2024] from https://www.kaggle.com/datasets

[35] F. Türk, "Analysis of intrusion detection systems in UNSW-NB15 and NSL-KDD datasets with machine learning algorithms", *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, Vol. 12, No. 2, pp. 465-477, 2023.

[36] M. Zakariah, S. A. AlQahtani, A. M. Alawwad, and A. A. Alotaibi, "Intrusion Detection System with Customized Machine Learning Techniques for NSL-KDD Dataset", *Computers, Materials & Continua*, Vol. 77, No. 3, 2023.

[37] S. M. Kasongo, "A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework", *Computer Communications*, Vol. 199, pp. 113-125, 2023.

[38] A. H. Aljammal, I. Al-Oqily, M. Obiedat, A. Qawasmeh, S. Taamneh, and F. I. Wedyan, "Anomaly intrusion detection using machine learning-IG-R based on NSL-KDD dataset", *Bulletin of Electrical Engineering and Informatics*, Vol. 13, No. 6, pp. 4466-4474, 2024.