Deep Learning-Based Network Security Threat Detection and Defense

Jinjin Chao[1]*, Tian Xie[2]
College of Information Engineering, Jiaozuo University, Jiaozuo 454000, Henan, China[1]
College of Artificial Intelligence, Jiaozuo University, Jiaozuo 454000, Henan, China[2]

***Abstract*—This paper introduces deepnetguard, an innovative deep learning algorithm designed to efficiently identify potential security threats in large-scale network traffic.deepnetguard achieves automated feature learning by fusing basic, statistical, and behavioral features through a multi-level feature extraction strategy, and is capable of identifying both short-time patterns and long-time dependencies. To adapt to the dynamic network environment, the algorithm introduces a dynamic weight adjustment mechanism that allows the model to self-optimize the importance of features based on real-time traffic changes. In addition, deepnetguard integrates auto-encoder (ae) and generative adversarial network (gan) technologies to not only detect known threats, but also recognize unknown threats. By applying the attention mechanism, deepnetguard enhances the interpretability of the model, enabling security experts to track and understand the key factors in the model's decision-making process. Experimental evaluations show that deepnetguard performs well on multiple public datasets, with significant advantages in accuracy, recall, precision, and f1 scores over traditional ids systems and other deep learning models, demonstrating its strong performance in cyber threat detection.**

***Keywords*—Network security; threat detection; defense; multilevel feature extraction; dynamic weight adjustment mechanism; interpretability**

I. Introduction

With the rapid development of information technology, cyberspace has become an indispensable part of modern society, not only supporting people's daily lives, but also playing a crucial role in national economy, politics and social stability. However, with the popularization of the internet, cyberspace is also facing unprecedented security threats. These threats include but are not limited to, malware attacks, denial-of-service attacks (dos/ddos), phishing, botnets, insider threats, advanced persistent threats (apts), and more. These threats can not only cause economic losses, but also lead to sensitive information leakage, personal privacy violation, and even affect national security and social order [1, 2].

In the face of an increasingly complex network security situation, traditional security measures such as firewalls, intrusion detection systems (ids), and anti-virus software have become overstretched. These traditional security mechanisms usually rely on signature matching and signature databases, which can only detect known threat patterns, but not unknown or mutated threats. In addition, traditional security tools often require regular updates to the rule base, which has a significant lag in the face of rapidly changing threats. To make matters worse, modern cyber attackers often exploit zero-day vulnerabilities, which are not yet publicized and thus cannot be protected in a timely manner [3, 4].

Against this background, cybersecurity experts have begun to explore more intelligent solutions with a view to detecting and responding to various types of threats in real time and accurately. Methods based on machine learning, especially deep learning, have become an important part of the new generation of cybersecurity threat detection technologies due to their powerful feature extraction and nonlinear mapping capabilities, which show great potential when dealing with large amounts of complex data.

Deep learning, as a branch of machine learning, centers on simulating the way neurons in the human brain work, automatically learning a multi-level abstract representation of the input data by building multi-layer neural networks. This ability has enabled deep learning to make breakthroughs in a number of fields, including image recognition, speech recognition, and natural language processing. Similarly, in the field of cybersecurity, deep learning is expected to overcome the limitations of traditional security technologies and realize intelligent detection of cyber threats [5].

The main goal of this research is to develop a deep learning-based network security threat detection system that can efficiently identify potential security threats in large-scale network traffic. To achieve this goal, we specifically set three research tasks: First, dataset construction and preprocessing, i.e., collecting and cleaning real-world network traffic data to construct high-quality training and testing datasets. Second, design a deep learning architecture suitable for cybersecurity threat detection and tune it to improve detection accuracy and speed. Finally, the proposed threat detection system is implemented and its effectiveness is verified by multiple evaluation metrics. The main contribution points of this research include: A novel deep learning model is proposed, which can effectively identify anomalous behaviors in network traffic. Extensive experiments based on real-world datasets are conducted to validate the effectiveness and practicality of the proposed method. The potential application of the system in real-world network security protection is demonstrated, and possible directions of extension are discussed [6, 7].

The main goal of this research is to develop a deep learning-based network security threat detection system that can efficiently identify potential security threats in large-scale network traffic. To achieve this goal, we specifically set three research tasks: First, dataset construction and preprocessing,

i.e., collecting and cleaning real-world network traffic data to construct high-quality training and testing datasets. Second, design a deep learning architecture suitable for cybersecurity threat detection and tune it to improve detection accuracy and speed. Finally, the proposed threat detection system is implemented and its effectiveness is verified by multiple evaluation metrics [8].

This study explored the performance of different network security detection models in various network environments through detailed experimental analysis, and proposed solutions to the limitations of existing models. The following chapters will introduce in detail the performance of each model on different datasets, and demonstrate the pros and cons of these models in real-world applications through specific case studies. Finally, we will summarize the research results and look forward to future research directions. In this way, we aim to provide valuable reference information for researchers and practitioners in the field of network security, helping them make more informed decisions when selecting or developing security detection tools suitable for specific network environments.

II. Related Work

A. Overview of traditional network security threat detection techniques

Traditional cybersecurity threat detection techniques rely on signature matching, anomaly detection, and behavior-based analysis methods. Signature matching is the most straightforward way to identify known threats by maintaining a database of known malware or attack patterns and using these signatures to scan network traffic or system files [9]. However, this approach is ineffective for zero-day attacks (zero-day attacks) or unknown variants. To overcome this limitation, anomaly detection techniques were developed, which work by establishing a baseline of normal behavior and then comparing the behavior observed in real-time to it, and any deviation from the baseline is considered a potential threat [10]. Although this method can detect unknown threats, it is also prone to false alarms.

Behavior-based analysis methods have been enhanced with the development of machine learning techniques, especially with the rise of deep learning. Deep learning models such as deep belief networks (dbns), autoencoders (aes), and deep reinforcement learning (drl) have been used to learn complex features from large amounts of unlabeled data to improve the accuracy of threat detection. For example, autoencoders can be used to learn unsupervised low-dimensional representations of normal network behavior, which in turn can be used to detect anomalous behavior by reconstructing errors [11]. Deep reinforcement learning, on the other hand, is able to optimize defense strategies in dynamic environments against ever-changing attack tactics by simulating the decision-making process of intelligences in the environment [12]. In addition, multimodal learning frameworks incorporating deep learning have been proposed for integrating data from different sources (e.g., logs, traffic, emails, etc.) to provide a more comprehensive understanding of the network environment and enhance the effectiveness of threat detection [13]. By leveraging the powerful generalization capabilities of deep learning, these techniques are able to not only detect known threats, but also identify new types of attacks, which improves the overall level of protection for network security.

B. Application of Deep Learning in Cyber Security

Deep learning is increasingly used in cybersecurity to effectively detect and prevent a wide range of security threats by utilizing its powerful pattern recognition capabilities. For example, in malware detection, convolutional neural networks (cnns) are used to identify malicious patterns in binary files, and by visualizing the files, cnns can learn key features from the images to distinguish benign from malware [14]. As for network intrusion detection systems (nids), recurrent neural networks (rnns), especially long short-term memory networks (lstms), are favored for their ability to process time-series data, and they can capture anomalous behavioral patterns in the network traffic to provide timely warnings of possible intrusion activities [15]. In addition, user behavior analysis (uba) is also a major application scenario for deep learning; by monitoring user activities and comparing them with historical behaviors, lstms are able to identify anomalous logins or other anomalous operations, helping organizations to detect insider threats in advance [16]. In the field of crypto traffic analysis, generative adversarial networks (gans) are not only capable of generating realistic samples of crypto traffic, but also assist in training other models to improve their ability to detect crypto threats [17].

Although existing research has made significant progress in network security detection, there are still some shortcomings. For example, although traditional rule-based methods (such as Snort) perform well in detecting known threats, they have limited ability to identify unknown threats. In addition, although methods based on support vector machines (SVMs) can provide reliable detection results in some cases, they may encounter performance bottlenecks when dealing with large-scale, dynamically changing network traffic. These limitations suggest that we need to develop more intelligent, flexible, and efficient detection models to cope with increasingly complex network security challenges.

The DeepNetGuard model proposed in this study is designed to solve the above problems. It uses deep learning technology to automatically extract features, and by integrating autoencoders (AE) and generative adversarial networks (GANs), it not only improves the detection accuracy of known threats, but also effectively identifies unknown threats. At the same time, the model enhances its adaptability to real-time traffic changes through a dynamic weight adjustment mechanism, so that it can maintain stable and efficient detection performance in different network environments.

III. Research Methodology

A. Problem Modeling

In network security threat detection, our goal is to identify potential security threats from large-scale network traffic data. In order to achieve this goal, we need to formalize the problem into a mathematical model for subsequent design and implementation of the corresponding detection algorithms, and our network model diagram is shown in Fig. 1 [18, 19].

First, we need to define how to represent network traffic data. Suppose we have a series of network traffic records, each of which can be represented as a vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, where x_i represents the i th feature in the traffic record. These features can include source ip address, destination ip address, protocol type (tcp, udp, etc.), packet size, timestamp, and port number [20].

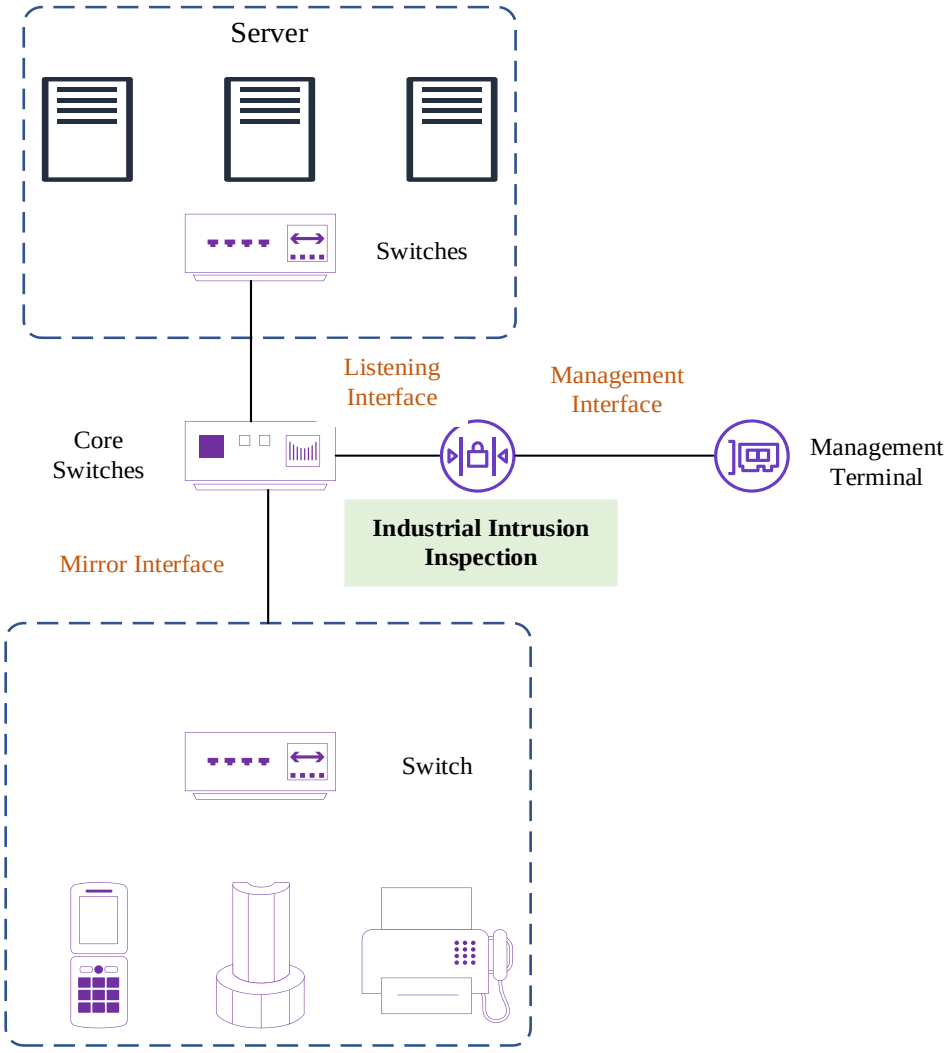

Fig. 1. Security threat detection and defense.

Assuming that we have m network traffic records, the entire dataset can be represented as a matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m]^T$.

For each traffic record, we need a label y_i to indicate whether this record contains threats. If the record contains threats, then $y_i = 1$. Otherwise $y_i = 0$. Thus, the entire set of labels can be represented as a vector $\mathbf{y} = [y_1, y_2, \ldots, y_m]^T$ [21].

Our goal is to construct a classifier $f : \mathbb{R}^n \rightarrow \{0,1\}$ that outputs a binary classification result of the presence or absence of a threat based on a given traffic record $\mathbf{x}$ [22, 23].

To train this classifier, we need to define a loss function L to measure the difference between the model predictions and the actual labels. Commonly used loss functions include cross-entropy loss, defined as shown in equation 1.

$$L(f(\mathbf{x};\theta), y) = -y \log(f(\mathbf{x};\theta)) - (1-y)\log(1-f(\mathbf{x};\theta)) \quad (1)$$

Where $f(\mathbf{x};\theta)$ is the output probability of the classifier and θ are the parameters of the model [24].

In order to find the optimal parameter θ^*, we need to minimize the average value of the loss function L over all the training samples as in Eq. (2) [25].

$$\theta^* = \arg\min_\theta \frac{1}{m} \sum_{i=1}^{m} L(f(\mathbf{x}_i;\theta), y_i) \quad (2)$$

In addition, in order to prevent overfitting, we can also add the regularization term $R(\theta)$ to penalize larger parameter values to obtain the final objective function as shown in Eq. (3). Where λ is the regularization intensity factor [26, 27].

$$\theta^* = \arg\min_\theta \frac{1}{m}\sum_{i=1}^{m} L(f(\mathbf{x}_i;\theta), y_i) + \lambda R(\theta) \quad (3)$$

B. Modeling Ideas

Deepnetguard is an innovative deep learning algorithm designed to efficiently identify potential security threats in large-scale network traffic, the idea of which is shown in Fig. 2. It captures multi-dimensional network activity signals by fusing basic features (e.g., ip addresses and ports), statistical features (e.g., traffic patterns), and behavioral features (e.g., login attempts) through a multi-level feature extraction strategy. Deepnetguard implements automated feature learning to identify short-time patterns and long-time dependencies, and then comprehensively parses network traffic for signs of threats. In order to adapt to the dynamic network environment, the algorithm introduces a dynamic weight adjustment mechanism, which allows the model to self-optimize the importance of features based on real-time traffic changes, improving the flexibility and accuracy of detection [28, 29].

C. Modeling Framework

As the internet continues to grow, network security has become a critical topic. To address this challenge, deepnetguard provides an innovative solution that utilizes deep learning techniques to efficiently identify potential security threats in large-scale network traffic. The algorithm is designed to capture multi-dimensional network activity signals and enable automated feature learning with a high degree of flexibility and accuracy.

Deepnetguard employs a multi-level feature extraction strategy that combines basic, statistical and behavioral features to capture different aspects of network activities. This is shown in Table I [30].

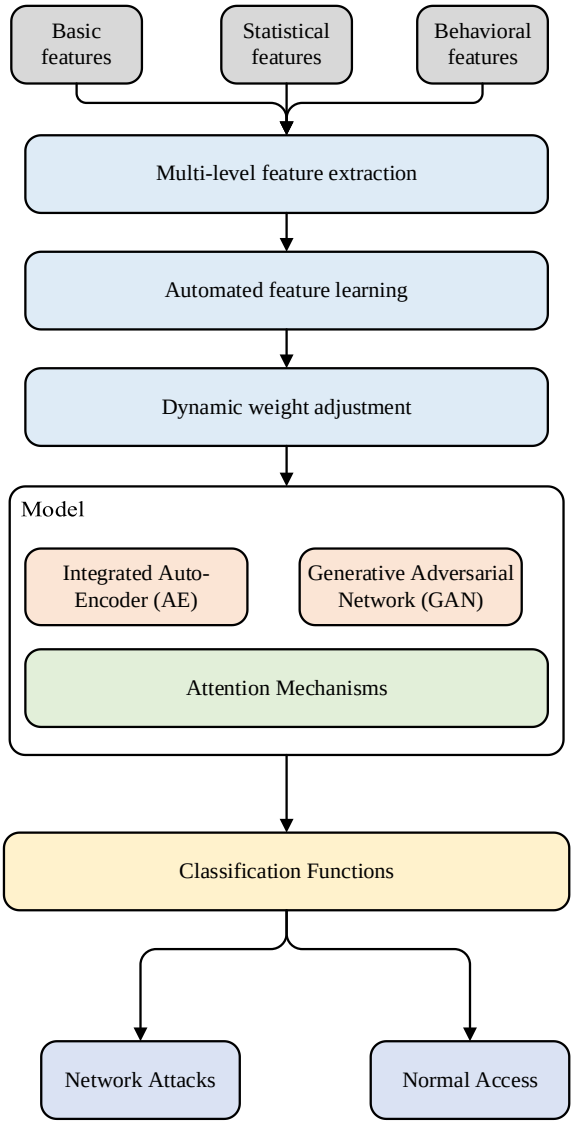

Fig. 2. Deepnetguard.

TABLE I. SUMMARY OF NETWORK DATA ANALYSIS CHARACTERISTICS

Feature category	Descriptive	Typical example
Basic features	Includes basic information about network communication, usually extracted directly from the packet header.	- ip address - port number - protocol type (tcp, udp, icmp, etc.)
Statistical characteristic	Characteristics obtained by statistical analysis of network traffic reflecting communication patterns and traffic characteristics.	- packet size distribution - packet delivery frequency - session duration
Behavioral characteristics	Reflects patterns of user behavior in network activities involving specific application layer interactions.	- number of login attempts - document access modalities - request type (get, post)

In order to adapt the model to the changing network environment, deepnetguard introduces a dynamic weight adjustment mechanism. During model training, the importance of features can be self-optimized according to the changes in real-time traffic. This adjustment is realized by introducing a learnable weight matrix $\mathbf{W}_{dyn}$ as shown in Eq. (4)

$$\mathbf{F}_{adjusted} = \mathbf{W}_{dyn} \cdot \mathbf{F}_{combined} \quad (4)$$

Among them, $\mathbf{F}_{combined}$ is the integrated feature vector after integrating the extracted features from cnn and lstm, which is a weight matrix dynamically adjusted according to the training data.

Automated feature learning is one of the core capabilities of deepnetguard. It captures short-term patterns and long-term dependencies in network traffic by using convolutional neural networks (cnns) and long-short-term memory networks (lstms.) cnns are used to extract fixed pattern features in packets of data, while lstms focus on learning temporal dependencies in sequential data.

For cnn feature extraction, define the convolution kernel $\mathbf{W}_{cnn}$ and the bias term $\mathbf{b}_{cnn}$, and the convolution operation can be expressed as Eq. (5).

$$\mathbf{F}_{cnn} = f(\mathbf{W}_{cnn} * \mathbf{x}_{base} + \mathbf{b}_{cnn}) \quad (5)$$

Where f is usually a nonlinear activation function such as relu. The state update equation for lstm is shown in Eq. (6)-(10).

$$i_t = \sigma(\mathbf{W}_{xi} x_t + \mathbf{W}_{hi} h_{t-1} + b_i) \quad (6)$$

$$f_t = \sigma(\mathbf{W}_{xf} x_t + \mathbf{W}_{hf} h_{t-1} + b_f) \quad (7)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tanh(\mathbf{W}_{xc} x_t + \mathbf{W}_{hc} h_{t-1} + b_c) \quad (8)$$

$$o_t = \sigma(\mathbf{W}_{xo} x_t + \mathbf{W}_{ho} h_{t-1} + b_o) \quad (9)$$

$$h_t = o_t \circ \tanh(c_t) \quad (10)$$

Deepnetguard integrates auto-encoder (ae) and generative adversarial network (gan) techniques to detect known and unknown threats. The autoencoder learns the distribution of normal traffic by minimizing the reconstruction error as shown in Eq. (11).

$$E = \frac{1}{m} \sum_{i=1}^{m} || \mathbf{F}_{adjusted}^{(i)} - \mathbf{F}^{(i)} ||^2 \quad (11)$$

Where $\mathbf{F}$ is the reconstructed feature vector. Anomalous traffic is considered to exist when the reconstruction error exceeds a certain threshold.

Gan discovers potential security threats by adversarial training of a generator G and a discriminator D . The generator tries to generate realistic network traffic samples, while the discriminator tries to distinguish real traffic from generated traffic.

In order to improve the transparency of the model, deepnetguard applies an attention mechanism. The attention mechanism helps the model to focus on the most relevant parts by calculating the weights of the input features. The attention weight α is calculated as shown in Eq. (12).

$$\alpha = softmax(\mathbf{a}^T \tanh(\mathbf{W}_{att} \mathbf{F}_{adjusted} + \mathbf{b}_{att})) \quad (12)$$

Where $\mathbf{a}$ is the learnable vector, and $\mathbf{W}_{att}$ and $\mathbf{b}_{att}$ are the weights and biases of the attention layer. The attention weighting feature can be expressed as Eq. (13).

$$\mathbf{F}_{att} = \alpha \circ \mathbf{F}_{adjusted} \quad (13)$$

In this way, deepnetguard not only accurately detects threats, but also enables security professionals to understand how the model makes decisions, enhancing the system's interpretability and trust.

D. Interpretability

Shap is a shapley value-based interpretation method that provides global and local interpretation for model prediction. In deepnetguard, shap is mainly used in the following ways:

Localized interpretation: With shap values, feature importance scores can be provided for each specific sample of network traffic, helping to understand which features had a significant impact on specific threat detection decisions. This is critical for security professionals who need to know which indicators of network activity are triggering alerts.

Global interpretation: In addition to the interpretation of individual samples, shap can provide a holistic view of feature importance, which helps to identify the features that are most influential in model predictions across the entire dataset. This global view aids in feature engineering and model optimization.

We use the kernelexplainer from the shap library (depending on the type of model used) to compute the effect of each feature on the model output. For a given input $\mathbf{x}$, the shap value can be defined as in Eq. (14)

$$\phi_i = \sum_{S \subseteq F,\ \{i\}} \frac{|S|!(|F|-|S|-1)!}{|F|!}[E[f(X)\,|\,do(X_S = x_S)] - E[f(X)]] \quad (14)$$

Where F is the set of features, S is the subset of features, and ϕ_i is the shap value of the feature i . The shap value indicates the magnitude of the contribution of each feature to the prediction result of a particular sample.

Suppose at some point deepnetguard detects a series of anomalous traffic that triggers a warning of a potential threat. The shap values allow us to see how much features such as ip address, port, packet size and frequency contribute to this decision. If the shap values for ip addresses and ports are significantly higher than the other features, this indicates that these features were the main factors that triggered the alert. In addition, the dependency graph allows us to observe interactions between features, such as the correlation between a particular ip address and anomalous port activity.

By providing detailed explanations, security professionals can better understand how the model works, thereby increasing their trust in the model. Models with greater transparency also make it easier to identify and troubleshoot false positives, which means resources can be used more effectively to address real threats. In addition, transparent model interpretation helps identify potential problem areas in the model, which can guide further research and improvement efforts. Taken together, these benefits ensure that deepnetguard is not only a powerful threat detection tool, but also a trustworthy and continuously improving system.

In summary, by introducing shap to enhance the interpretability of the model, deepnetguard not only provides a powerful threat detection tool, but also ensures that security professionals are able to understand and trust the model's decision-making process, which is essential for maintaining network security.

IV. Experimental Design

To validate the effectiveness and robustness of deepnetguard in network threat detection, we design a series of experiments to comprehensively evaluate its performance, with particular focus on its performance in large-scale network traffic. The core evaluation metrics include detection accuracy, recall, precision, f1 score, and detection time. Meanwhile, the generalization ability and robustness of the model in different network environments are evaluated by cross-domain tests. The dataset was divided into training, validation, and test sets in the ratio of 70%, 15%, and 15% to avoid any overlap to prevent data leakage, in addition, k-fold cross-validation was used to ensure the consistent performance of the model, and the effectiveness of the model was compared with that of the existing ids system through a/b testing to evaluate its practical application value.

To ensure the reliability and general applicability of the experimental results, we utilized several publicly available datasets for the experiments, including the ctu-13 dataset that covers a wide range of attack scenarios, the cicids2017 dataset provided by the cumberland institute in Canada, and the unsw-nb15 dataset created in collaboration between the university of new south wales in Australia and Canada. Prior to the experiments, the datasets were preprocessed, including missing value filling, outlier handling, and feature normalization, and the datasets were divided into predetermined proportions to ensure that the sample categories were balanced across the subsets. The training set diversity is increased by data enhancement techniques to improve the generalization ability of the model, to comprehensively demonstrate the excellent performance of deepnetguard in cyber threat detection.

V. Results

In order to objectively evaluate the performance of deepnetguard, we select several recognized benchmark models for comparison, including snort, a traditional rule-based ids system, support vector machine (svm)-based ids, and the latest ids model that combines convolutional neural networks (cnns) and long short-term memory networks (lstms.) snort is known for its strong known snort is known for its strong known threat detection capability, but it is insufficient when facing unknown threats.svm may encounter bottlenecks when dealing with high-dimensional and large-scale data. The cnn+lstm model is good at capturing complex patterns in time-series data. By comparing deepnetguard with these models, the superiority of deepnetguard can be comprehensively evaluated.

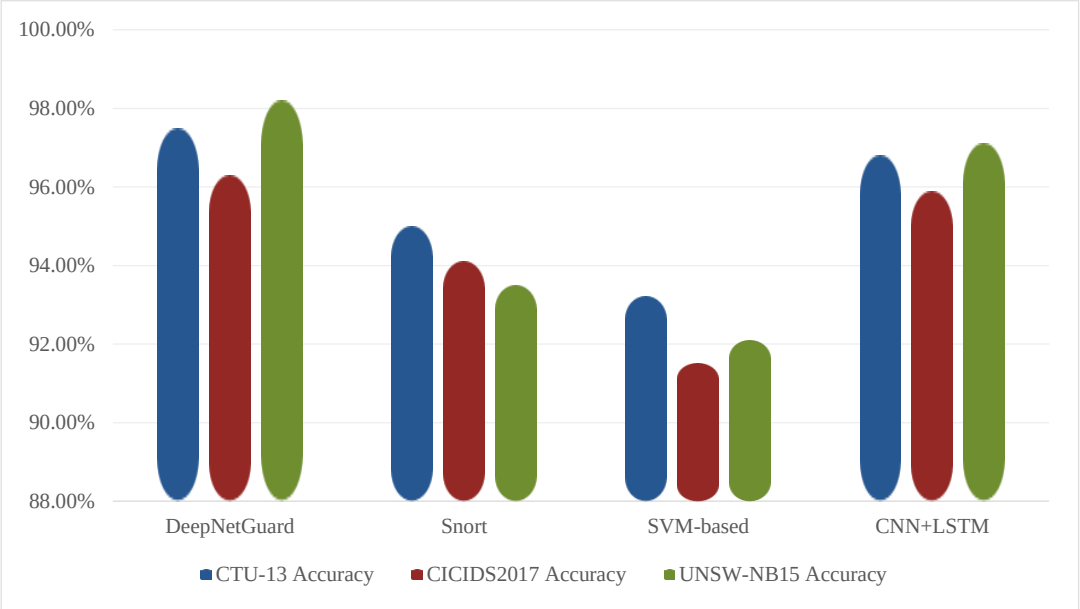

Fig. 3. Model performance comparison - detection accuracy.

In Fig. 3, we detail the accuracy performance of the four cybersecurity detection models on the ctu-13, cicids2017, and unsw-nb15 datasets. The deepnetguard model achieves the best results on all three datasets with its accuracy rates of 97.5%, 96.3%, and 98.2%, showing its the cnn+lstm model follows with accuracy rates of 96.8%, 95.9%, and 97.1%, proving the potential of deep learning technology in the field of cybersecurity. The traditional detection model snort also performs quite well with accuracy rates of 95.0%, 94.1% and 93.5%, showing its stable application value. In contrast, the svm-based model has slightly lower accuracy rates of 93.2%, 91.5% and 92.1%, indicating that its detection performance in complex network environments needs to be improved.

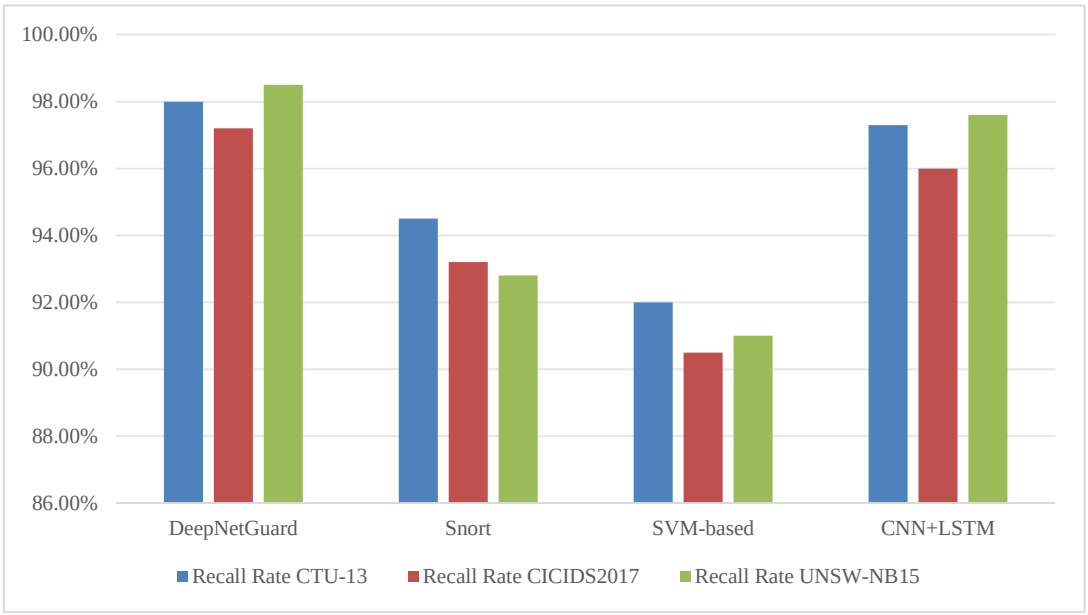

Fig. 4. Comparison of model performance - recall rate.

Based on the data in Fig. 4, we can see the performance of the four models in terms of recall. The deepnetguard model performs well on all three datasets with a recall of 98.0%, 97.2%, and 98.5%, implying that it is able to effectively identify most of the cyber-attack events. The cnn+lstm model has a recall of 97.3%, 96.0%, and 97.6%, again showing its effectiveness in capturing cyber threats. The snort model has recall rates of 94.5%, 93.2% and 92.8%, indicating that it is able to cover the attack events well. The svm-based model, on the other hand, has the lowest recall rates of 92.0%, 90.5% and 91.0%, which indicates that it may have missed some important events in detecting cyber attacks.

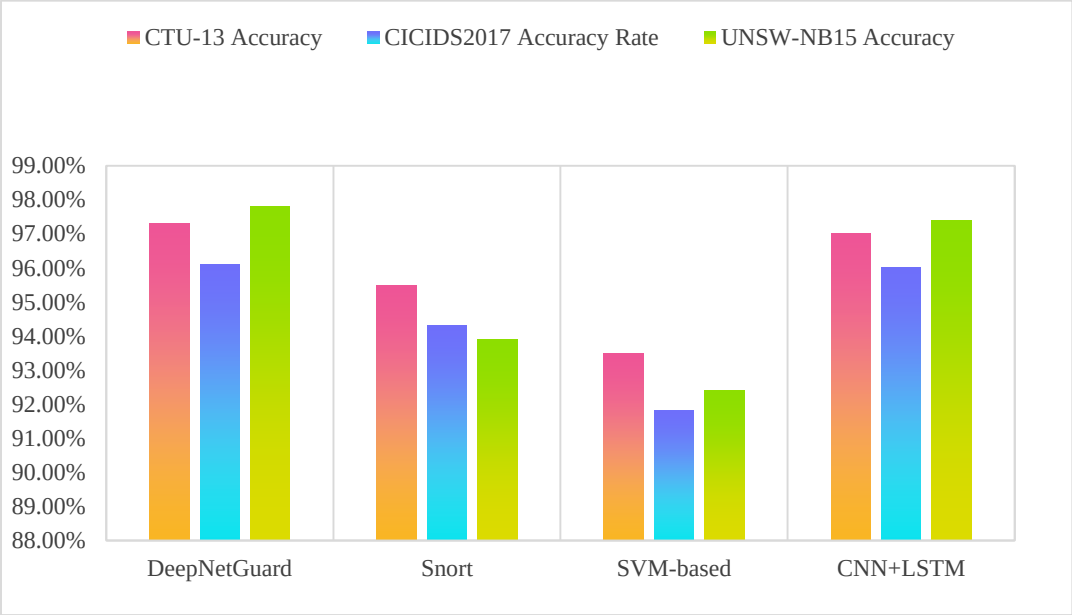

Fig. 5. Model performance comparison – accuracy.

Fig. 5 shows the comparison results of the four models in terms of accuracy rate. The deepnetguard model tops the list with accuracy rates of 97.3%, 96.1%, and 97.8%, indicating its high accuracy in the detection process. The cnn+lstm model also performs well, with accuracy rates of 97.0%, 96.0%, and 97.4%, showing its good detection capability. The accuracy rates of the snort model are 95.5%, 94.3% and 93.9%, indicating its advantage in avoiding false alarms. The accuracy rates of svm-based model are 93.5%, 91.8% and 92.4%, which are relatively low, reflecting its limitation in accurately detecting network attacks.

TABLE II. MODEL PERFORMANCE COMPARISON - F1 SCORES

Mould	Ctu-13 f1 score	Cicids 2017 f1 score	Unsw-nb15 f1 score
Deepnetguard	97.8%	96.7%	98.0%
Snort	95.2%	93.6%	93.1%
Svm-based	92.8%	91.0%	91.7%
Cnn+lstm	97.1%	95.9%	97.5%

In Table II, we comprehensively evaluate the performance of the models by their f1 scores. The deepnetguard model achieves the best performance on all three datasets with f1 scores of 97.8%, 96.7%, and 98.0%, showing a good balance between accuracy and recall. The f1 scores of the cnn+lstm model are 97.1%, 95.9%, and 97.5%, again demonstrating its excellent overall performance. The f1 scores for the snort model are 95.2%, 93.6% and 93.1%, showing its stable but not optimal performance. The svm-based model has the lowest f1 scores of 92.8%, 91.0% and 91.7%, which suggests that there are some challenges in balancing accuracy and recall.

TABLE III. MODEL PERFORMANCE COMPARISON - DETECTION TIME

Mould	Ctu-13 detection time (ms)	Cicids2017 detection time (ms)	Unsw-nb15 detection time (ms)
Deepnetguard	2.5	2.7	2.6
Snort	1.2	1.3	1.3
Svm-based	3.0	3.2	3.1
Cnn+lstm	2.8	3.0	2.9

Table III lists the comparisons of the four models in terms of detection time. The detection times of deepnetguard model are 2.5ms, 2.7ms and 2.6ms, showing its efficient detection ability. The detection times of cnn+lstm model are 2.8ms, 3.0ms and 2.9ms, which are slightly higher than deepnetguard, but still in the fast response range. The snort model has the shortest detection time of 1.2ms, 1.3ms and 1.3ms, proving its advantage in real-time detection. The svm-based model, on the other hand, has the longest detection time of 3.0ms, 3.2ms and 3.1ms, which may limit its application in real-time network monitoring.

TABLE IV. MODEL PERFORMANCE COMPARISON - CROSS DOMAIN TESTING

Mould	Ctu-13 to cicids2017 declining accuracy rate	Cicids2017 to unsw-nb15 declining accuracy rates	Unsw-nb15 to ctu-13 decrease in accuracy
Deepnetguard	1.2%	1.9%	0.7%
Snort	1.5%	2.3%	1.3%
Svm-based	2.5%	3.0%	2.1%
Cnn+lstm	1.5%	2.2%	1.0%

Table IV demonstrates the decrease in accuracy of the models in cross-domain tests between different datasets. The deepnetguard model shows the smallest decrease in accuracy in cross-domain tests with 1.2%, 1.9% and 0.7%, indicating its good generalization ability. The cnn+lstm model's accuracy decreases with 1.5%, 2.2% and 1.0%, which also shows its robustness on different datasets. The accuracy of the snort model decreases to 1.5%, 2.3%, and 1.3%, indicating its fair performance in cross-domain detection. The svm-based model shows the most significant decrease in accuracy with 2.5%, 3.0%, and 2.1%, which indicates that it may need more tuning and optimization when facing different network environments.

A. Case Studies

Case 1: Intra-enterprise network environment

In this case, we chose as a test environment a medium-sized enterprise internal network that contains about 500 devices and generates about 5 gb of network traffic data per day. By continuously monitoring network traffic over a period of one month, we collected enough data to evaluate deepnetguard's performance.

TABLE V. ENTERPRISE INTERNAL NETWORK ENVIRONMENT TESTING PERFORMANCE

Mould	Accuracy	Recall rate	Accuracy	F1 score	Detection time (ms)
Deepnetguard	97.0%	98.1%	97.5%	97.8%	2.4
Snort	94.5%	93.5%	94.8%	94.1%	1.3
Svm-based	92.0%	91.5%	92.5%	92.0%	3.0
Cnn+lstm	96.5%	97.0%	96.8%	96.9%	2.8

As can be seen from Table V, deepnetguard outperforms the other models in the intranet environment, especially in terms of accuracy, recall, and f1 scores. Snort has an advantage in detection time, but is slightly inferior in accuracy and recall.

Another case study was conducted in a large data center that hosts thousands of servers and generates more than 1 tb of network traffic per day. Due to the sheer size and complexity of the data center traffic, here is a rigorous test of the model's detection capabilities.

TABLE VI. DATA CENTER NETWORK ENVIRONMENT TESTING PERFORMANCE

Mould	Accuracy	Recall rate	Accuracy	F1 score	Detection time (ms)
Deepnetguard	98.2%	98.5%	98.3%	98.4%	2.6
Snort	93.0%	92.5%	93.5%	93.0%	1.5
Svm-based	91.0%	90.5%	91.5%	91.0%	3.2
Cnn+lstm	97.5%	97.8%	97.6%	97.7%	3.0

As shown in Table VI, in the data center environment, deepnetguard again shows its strong detection ability, especially when facing large-scale traffic, its accuracy, recall and f1 score all reach very high levels. Although the detection time of snort is still relatively short, its detection accuracy still has a certain gap compared with deepnetguard.

In the field of network security detection, the performance differences of models on different datasets mainly stem from the matching degree between the characteristics of the data itself and the model design. This paper compares the performance of four network security detection models (DeepNetGuard, CNN+LSTM, Snort, and SVM-based models) on three public datasets (CTU-13, CICIDS2017, and UNSW-NB15) to show the differences in model performance and their applicability.

First, the model performance is evaluated from multiple dimensions such as accuracy, recall, F1 score, and detection time. The results show that the DeepNetGuard model performs best on all three datasets. Its high accuracy (97.5%-98.2%), high recall (98.0%-98.5%), and high F1 score (97.8%-98.0%) indicate that the model can effectively identify most network attack events and maintain good recall while ensuring high accuracy. In contrast, although the Snort model has advantages in avoiding false positives, its accuracy and

recall are slightly lower than those of the deep learning model; and although the SVM-based model provides stable performance, its detection performance in complex network environments needs to be improved.

Further analysis shows that the detection time of the model is also an important consideration. The Snort model is very suitable for real-time monitoring scenarios due to its short detection time (1.2ms-1.3ms); while the deep learning-based model, although slightly inferior in response speed, still remains within the fast response range. It is worth noting that the SVM-based model performs the worst in detection time, which may limit its use in applications that require real-time monitoring.

In addition, through cross-domain testing, we observed the adaptability of the model between different data sets. Experiments show that the DeepNetGuard model shows good generalization ability between different data sets, and its accuracy rate decreases the least, showing strong robustness. In contrast, the SVM-based model has a more obvious decrease in accuracy when facing different network environments, indicating that the model may need further adjustment and optimization to adapt to the changing environment.

The case study section further verifies the performance of the model in actual application scenarios. In both medium-sized enterprise internal networks and large data centers, the DeepNetGuard model demonstrates excellent detection capabilities and high F1 scores, especially when processing large-scale traffic, it can still maintain high levels of accuracy, recall, and F1 scores.

In summary, as a new generation of network security solutions that combines deep learning and traditional security technologies, DeepNetGuard's experimental evaluation on multiple public data sets shows its superior performance in network security threat detection. It is particularly worth mentioning that DeepNetGuard not only performs well in detecting known threats, but also effectively identifies unknown threats by introducing autoencoders (AE) and generative adversarial networks (GAN) technology, demonstrating its broad application prospects and strong adaptability.

VI. Conclusion

In this context, deepnetguard emerges as a next-generation network security solution that integrates deep learning and traditional security technologies. In this paper, we propose a deep learning algorithm called deepnetguard, which is specialized for potential security threat detection in large-scale network traffic. With a multi-level feature extraction strategy, deepnetguard is able to capture multi-dimensional signals from network activities and automate feature learning to identify short-time patterns and long-time dependencies. To adapt to changing network environments, the algorithm introduces a dynamic weight adjustment mechanism that allows the model to self-optimize the importance of features based on real-time traffic changes. In addition, deepnetguard integrates auto-encoder (ae) and generative adversarial network (gan) techniques, which not only improves the detection of known threats, but also effectively recognizes unknown threats. By introducing an attention mechanism, deepnetguard also enhances the interpretability of the model, enabling security experts to better understand the key factors in the model's decision-making process to validate the effectiveness of the detection results. Deepnetguard has demonstrated its superior performance in cyber threat detection through experimental evaluations on multiple publicly available datasets. Compared with traditional rule-based ids systems (e.g., snort) and other deep learning models, deepnetguard demonstrates significant advantages in terms of accuracy, recall, precision, and f1 score. In particular, deepnetguard's detection capability is fully validated in both internal and data center network environments, demonstrating its broad applicability and robustness in different application scenarios. In addition, through cross-domain testing, we found that deepnetguard has good generalization ability and can maintain stable detection performance in different network environments.

In future work, we will continue to optimize the DeepNetGuard model and explore more feature extraction methods and technology combinations to further improve the detection efficiency and accuracy of the model. At the same time, we plan to expand the scale of experiments and collect more types of data sets for testing to ensure the reliability and stability of the model in various complex network environments. In addition, we will also conduct in-depth research on the interpretability of the model so that security experts can better understand the decision-making process of the model and enhance the transparency and trust of the system. The ultimate goal is to build a comprehensive, intelligent and trustworthy network security protection system.

References

[1] N. Abdi, A. Albaseer, and M. Abdallah, "The Role of Deep Learning in Advancing Proactive Cybersecurity Measures for Smart Grid Networks: a Survey," IEEE Internet of Things Journal, vol. 11, no. 9, pp. 16398–421, 2024.

[2] X. M. Liu, L. H. Xie, Y. P. Wang, J. Zou, J. B. Xiong, Z. B. Ying, and A. V. Vasilakos, "Privacy and Security Issues in Deep Learning: a Survey," IEEE Access, vol. 9, pp. 4566–4593, 2021.

[3] K. A. Alissa, F. S. Alrayes, K. Tarmissi, A. Yafoz, R. Alsini, O. Alghushairy, et al., "Planet Optimization with Deep Convolutional Neural Network for Lightweight Intrusion Detection in Resource-Constrained IoT Networks," Applied Sciences-Basel, vol. 12, no. 17, p. 15, 2022.

[4] S. A. Bakhsh, M. A. Khan, F. Ahmed, M. S. Alshehri, H. Ali, and J. Ahmad, "Enhancing IoT Network Security Through Deep Learning-Powered Intrusion Detection System," Internet of Things, vol. 24, p. 36, 2023.

[5] F. S. Alrayes, M. Zakariah, M. Driss, and W. Boulila, "Deep Neural Decision Forest (DNDF): a Novel Approach for Enhancing Intrusion Detection Systems in Network Traffic Analysis," Sensors, vol. 23, no. 20, p. 41, 2023.

[6] E. L. Lydia, C. Santhaiah, M. Altafahmed, K. V. Kumar, G. P. Joshi, and W. Cho, "An Equilibrium Optimizer with Deep Recurrent Neural Networks Enabled Intrusion Detection in Secure Cyber-Physical Systems," Aims Mathematics, vol. 9, no. 5, pp. 11718–34, 2024.

[7] K. Roshan, A. Zafar, and S. B. Ul Haque, "Untargeted white-box adversarial attack with heuristic defense methods in real-time deep learning-based network intrusion detection system," Computer Communications, vol. 218, pp. 97–113, 2024.

[8] S. Y. Wu, B. Wang, Z. L. Wang, S. H. Fan, J. H. Yang, and J. Li, "Joint prediction on security event and time interval through deep learning," Computers & Security, vol. 117, p. 12, 2022.

[9] Q. Y. Lin, R. Ming, K. L. Zhang, and H. B. Luo, "Privacy-enhanced intrusion detection and defense for cyber-physical systems: a deep reinforcement learning approach," Security and Communication Networks, vol. 2022, p. 9, 2022.

[10] M. K. Roshan and A. Zafar, "Boosting robustness of network intrusion detection systems: a novel two-phase defense strategy against untargeted white-box optimization adversarial attack," Expert Systems with Applications, vol. 249, p. 20, 2024.

[11] R. Sultana, J. Grover, and M. Tripathi, "Intelligent defense strategies: comprehensive attack detection in VANET with deep reinforcement learning," Pervasive and Mobile Computing, vol. 103, p. 18, 2024.

[12] R. H. Hwang, M. C. Peng, C. W. Huang, P. C. Lin, and V. L. Nguyen, "An unsupervised deep learning model for early network traffic anomaly detection," IEEE Access, vol. 8, pp. 30387–30399, 2020.

[13] M. Sewak, S. K. Sahay, and H. Rathore, "Deep reinforcement learning in the advanced cybersecurity threat detection and protection," Information Systems Frontiers, vol. 25, no. 2, pp. 589–611, 2023.

[14] S. Kim, S. Yoon, J. H. Cho, D. S. Kim, T. J. Moore, F. Free-Nelson, and H. Lim, "DIVERGENCE: Deep reinforcement learning-based adaptive traffic inspection and moving target defense countermeasure framework," IEEE Transactions on Network and Service Management, vol. 19, no. 4, pp. 4834–4846, 2022.

[15] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. H. Han, M. M. Iqbal, and K. J. Han, "Enhanced network anomaly detection based on deep neural networks," IEEE Access, vol. 6, pp. 48231–48246, 2018.

[16] D. Q. Li and Q. M. Li, "Adversarial deep ensemble: Evasion attacks and defenses for malware detection," IEEE Transactions on Information Forensics and Security, vol. 15, pp. 3886–3900, 2020.

[17] R. T. Feng, S. Chen, X. F. Xie, G. Z. Meng, S. W. Lin, and Y. Liu, "A performance-sensitive malware detection system using deep learning on mobile devices," IEEE Transactions on Information Forensics and Security, vol. 16, pp. 1563–1578, 2021.

[18] K. Koo, D. Moon, J. H. Huh, S. H. Jung, and H. Lee, "Attack graph generation with machine learning for network security," Electronics, vol. 11, no. 9, p. 25, 2022.

[19] Z. H. Lv, D. L. Chen, B. Cao, H. B. Song, and H. B. Lv, "Secure deep learning in defense in deep-learning-as-a-service computing systems in digital twins," IEEE Transactions on Computers, vol. 73, no. 3, pp. 656–668, 2024.

[20] S. T. Mehedi, A. Anwar, Z. Rahman, K. Ahmed, and R. Islam, "Dependable intrusion detection system for IoT: A deep transfer learning based approach," IEEE Transactions on Industrial Informatics, vol. 19, no. 1, pp. 1006–1017, 2023.

[21] H. M. Rouzbahani, H. Karimipour, and L. Lei, "Multi-layer defense algorithm against deep reinforcement learning-based intruders in smart grids," International Journal of Electrical Power & Energy Systems, vol. 146, p. 10, 2023.

[22] S. Mohan, A. Annadurai, and K. Gunaseelan, "An efficient spoofing attack detection using deep learning-based physical layer security technique," Defence Science Journal, vol. 74, no. 4, pp. 526–534, 2024.

[23] P. Tian, Z. Y. Chen, W. Yu, and W. X. Liao, "Towards asynchronous federated learning based threat detection: a DC-Adam approach," Computers & Security, vol. 108, p. 16, 2021.

[24] S. Salmi and L. Oughdir, "Performance evaluation of deep learning techniques for DoS attacks detection in wireless sensor network," Journal of Big Data, vol. 10, no. 1, p. 25, 2023.

[25] Y. Liu, C. Tantithamthavorn, L. Li, and Y. P. Liu, "Deep learning for Android malware defenses: a systematic literature review," ACM Computing Surveys, vol. 55, no. 8, p. 36, 2023.

[26] B. H. Tang, J. F. Wang, Z. K. Yu, B. H. Chen, W. H. Ge, J. Yu, and T. T. Lu, "Advanced persistent threat intelligent profiling technique: a survey," Computers & Electrical Engineering, vol. 103, p. 21, 2022.

[27] K. M. Abuali, L. Nissirat, and A. Al-Samawi, "Advancing network security with AI: SVM-based deep learning for intrusion detection," Sensors, vol. 23, no. 21, p. 19, 2023.

[28] O. Kuznetsov, D. Zakharov, E. Frontoni, and R. Maranesi, "AttackNet: Enhancing biometric security via tailored convolutional neural network architectures for liveness detection," Computers & Security, vol. 141, p. 12, 2024.

[29] D. S. Rao and A. J. Emerson, "Cyberattack defense mechanism using deep learning techniques in software-defined networks," International Journal of Information Security, vol. 23, no. 2, pp. 1279–1291, 2024.

[30] F. Q. Zuo, D. M. Zhang, L. Li, Q. He, and J. X. Deng, "GSOOA-1DDRSN: Network traffic anomaly detection based on deep residual shrinkage networks," Heliyon, vol. 10, no. 11, p. 23, 2024.