Implementation of Cyber Network's Attacks Detection System with Deep Learning Designing Algorithms

Original Scientific Paper

Lubna Emad Kadhim
University of Imam Al-Kadhum,
College of Imam Al-Kadhum (IKC), Department of Computer Techniques Engineering
10011, Baghdad, Iraq
lubnaemad@alkadhum-col.edu.iq

Saif Aamer Fadhil
University of Imam Al-Kadhum,
College of Imam Al-Kadhum (IKC), Department of Computer Techniques Engineering
10011, Baghdad, Iraq
saifaamer@alkadhum-col.edu.iq

Sumaia M. Al-Ghuribi
Prince Sattam bin Abdulaziz University,
Faculty of Computer Engineering & Sciences,
Department of Software Engineering
Alkharj 11942, Riyadh, Saudi Arabia
Taiz University, Faculty of Applied Sciences,
Department of Computer Science, Taiz, Yemen
s.alghuribi@psau.edu.sa

Amjed Abbas Ahmed*
Universiti Kebangsaan Malaysia (UKM),
Faculty of Information Science and Technology,
Center for Cyber Security, 43600, Bangi, Malaysia
University of Imam Al-Kadhum,
College of Imam Al-Kadhum (IKC),
Department of Computer Techniques Engineering
10011, Baghdad, Iraq
amjedabbas@alkadhum-col.edu.iq

Mohammad Kamrul Hasan
Universiti Kebangsaan Malaysia (UKM),
Faculty of Information Science and Technology,
Center for Cyber Security, 43600, Bangi, Malaysia
mkh@ukm.edu.my

Shahrul A. Mohd Noah
Universiti Kebangsaan Malaysia (UKM),
Faculty of Information Science and Technology,
Centre for Artificial Intelligence Technology (CAIT)
43600, Bangi, Malaysia
shahrul@ukm.edu.my

Fatima N. AL-Aswadi
UCSI University,
Institute of Computer Science and Digital Innovation
56000, Kuala Lumpur, Malaysia
Hodeidah University,
Faculty of Computer Science and Engineering
Al Hudaydah, Yemen
Fatima.Nadeem@ucsiuniversity.edu.my

*Corresponding author

Abstract *– The internet has become indispensable for modern communication, playing a vital role in the development of smart cities and communities. However, its effectiveness is contingent upon its security and resilience against interruptions. Intrusions, defined as unauthorized activities that compromise system integrity, pose a significant threat. These intrusions can be broadly categorized into host intrusions, which involve unauthorized access and manipulation of data within a system, and network intrusions, which target vulnerabilities within the network infrastructure. To mitigate these threats, system administrators rely on Network Intrusion Detection Systems (NIDS) to identify and respond to security breaches. However, designing an effective and adaptable NIDS capable of handling novel and evolving attack strategies presents a significant challenge. This paper proposes a deep learning-based approach for NIDS development, leveraging Self-Taught Learning (STL) and the NSL-KDD benchmark dataset for network intrusion detection. The proposed approach is evaluated using established metrics, including accuracy, F-measure, recall, and precision. Experimental results demonstrate the effectiveness of STL in the 5-class categorization, achieving an accuracy of 79.10% and an F-measure of 75.76%. This performance surpasses that of Softmax Regression (SMR), which attained 75.23% accuracy and a 72.14% F-measure. The paper concludes by comparing the proposed approach's performance with existing state-of-the-art methods.*

Keywords: *cyber network, deep learning, intrusion detection system, network intrusion*

Received: May 18, 2024; Received in revised form: July 23, 2024; Accepted: July 24, 2024

1. INTRODUCTION

An intrusion detection system (IDS) [1] analyzes and monitors an organization's network devices [2]. If an invasion is identified, this system notifies users and stops additional harm. The two types of intrusion sensing systems are anomaly-dependent network intrusion detection systems (ANIDS) [3] and signature-dependent network intrusion detection systems. SNIDS [4], such as Snort, come pre-configured with attack signatures. To detect a compromise in network security, traffic is compared to the signatures that have been applied. If an ADNIDS identifies a deviation from normal traffic patterns, the observed network traffic is labeled as an intrusion. SNIDS are useful for detecting known threats since they have a high detection accuracy and a low false alarm rate. Due to the limited number of attack signatures that can be pre-installed, it is difficult for an IDS to recognize new or inventive attacks [5-9].

ADNIDS, on the other hand, are very successful in locating previously undiscovered, unique threats. Even though they have a higher rate of false positives, ADNIDS have gained considerable recognition in the research community due to their potential for identifying new attacks. Two impediments hinder the development of effective and adaptable NIDS capable of protecting against unknown future threats. Firstly, the sheer volume of information available makes it challenging to select suitable criteria for identifying anomalies within network data. Since attack methods constantly change and evolve, traits effective for one type of attack may be ineffective for another. Currently, insufficient labeled traffic data from real networks is available for developing effective NIDS. Creating such a labeled dataset from raw, real-time network traffic traces requires significant effort and time. Network managers are very cautious about disclosing security breaches within their networks, as they strive to protect both individual user privacy and the organization's trade secrets related to internal network structure [5]. NIDS are designed to distinguish between normal and anomalous traffic patterns.

Many NIDS employ feature selection to achieve more accurate classifications, which involves selecting a subset of meaningful features from recorded traffic. Feature selection can help reduce the risk of overfitting during training by removing irrelevant features and noise [6]. Comprehending sounds, images, and speech using deep learning algorithms is a relatively recent development. hese approaches enable the construction of effective feature representations from large amounts of unlabeled data, which can then be applied to smaller, labeled datasets for supervised classification. Data from both labeled and unlabeled distributions may originate from various sources, but they should ideally be related [7].

Deep learning approaches were expected to address the challenges of building effective NIDS [8]. It is feasible to gather unlabeled network traffic data from various sources across the network and then apply deep learning methods to extract useful feature representations. These features can then be used for supervised classification on a smaller, labeled dataset containing both normal and anomalous traffic. Such a dataset could be used to analyze traffic trends. Collecting labeled traffic data is possible within a controlled, secure, and isolated network environment [10].

Self-taught learning (STL) is a powerful approach for Network Intrusion Detection Systems (NIDS), providing a robust mechanism for detecting anomalous patterns in network traffic. STL typically utilizes large amounts of unlabeled network data to train deep learning models without requiring annotations. This data, often unstructured, undergoes feature extraction, where important characteristics such as packet size, protocol type, and source/destination IP addresses are extracted. A deep learning model can then leverage unsupervised learning, often through an autoencoder architecture, for pretraining. During pretraining, the model aims to minimize the difference between the original input and its reconstruction, effectively learning to represent normal network behavior. This process allows the model to encode normal network behavior. During deployment, deviations from this learned behavior can indicate anomalies, such as intrusions or attacks.

Once deployed, the trained model continuously monitors all incoming network traffic. Leveraging the learned representations, the model can effectively identify anomalies based on emerging patterns in real-time. This proactive approach enables network administrators to preempt or promptly detect threats before they escalate into more serious security breaches. Moreover, the model can adapt to new threats as it continuously learns from new data, refining its understanding of normal network behavior. Therefore, STL, particularly when applied to deep learning models for NIDS, offers a promising approach to strengthening cybersecurity and defending against evolving network threats.

The following are our contribution towards this research work:

- To achieve this goal, we propose a novel deep learning approach for NIDS based on self-taught learning, utilizing sparse autoencoders and softmax regression.
- This approach enabled the development of our proposed NIDS. We evaluate its performance on the NSL-KDD intrusion dataset, a widely used benchmark derived from the original KDD Cup 1999 dataset.
- We evaluate the performance of our STL-based NIDS on the NSL-KDD dataset and compare its effectiveness with existing methods.

2. RELATED WORK

It is important to note that this discussion focuses solely on studies that utilized the NSL-KDD dataset to evaluate effectiveness. Henceforth, any mention of a dataset refers to NSL-KDD. One of the earliest studies [11] employed Artificial Neural Networks (ANNs) with enhanced backpropagation to develop an intrusion detection system. This research utilized the entire training dataset, allocating 70% for training, 15% for validation, and 15% for testing. As anticipated, performance degraded when evaluated on unlabeled data. The training dataset will be analyzed. Another study [12] employed the J48 decision tree classifier with 10-fold cross-validation. This experiment utilized a reduced feature set of 22 attributes instead of the complete set of 41. Similar research [13] demonstrated that the Random Forest model achieved the lowest false alarm rate, surpassing other supervised tree-based classifiers.

Numerous two-level classification schemes have also been proposed. One study [14] used a Discriminative Multinomial Naive Bayes (DMNB) model as the base classifier, with nominal features converted to binary using a controlled filtering approach at the second level and 10-fold cross-validation. This concept was further developed using Random Forest and Ensembles of Balanced Nested Dichotomies (END) [15]. END is an abbreviation for ensembles of balanced nested dichotomies. As expected, this approach resulted in an increased detection rate and a reduced false positive rate.

Another study [16] proposed a novel two-level technique that first applies Principal Component Analysis (PCA) for feature reduction and then utilizes a Support Vector Machine (SVM) for classification, achieving high detection accuracy. While this approach, using the full training dataset with 41 features, demonstrated promising results, reducing the feature set to 23 improved the detection accuracy for specific attack types, albeit with a slight decrease in overall performance. Building upon their previous work, the authors of [17] first ranked features based on information gain and then applied behavior-based feature selection to reduce the feature set to 20. This approach led to an improvement in the reported accuracy on the training dataset. The secondary group of experiments utilized both training and testing datasets. An earlier study [18] combined fuzzy classification with a genetic algorithm, achieving a detection accuracy of at least 80% with a 1% false positive rate.

One notable study [19] revealed a significant performance degradation when training and testing data were combined. This research employed unsupervised clustering techniques to address its research questions. Similarly, another study [20] utilizing both training and testing datasets employed a k-nearest neighbors approach, achieving slightly higher detection accuracy and a lower false positive rate. Compared to the SVM-RBF approach, the Optimum-Path Forest (OPF) strategy, which utilizes graph partitioning for feature classification, has been shown to achieve a higher detection rate, although it is not as widely adopted as other techniques. The study [21] employed a deep learning approach, utilizing a Deep Belief Network (DBN) for feature selection and a Support Vector Machine (SVM) for classification. This approach, when trained on the training data, achieved an accuracy of approximately 92.84%.

Our research, which also utilizes both training and testing datasets, builds upon these earlier works by exploring the application of deep learning for NIDS. The authors of [22] adopt a semi-supervised learning scheme, similar to the one used in [23]. Their technique was validated using real-world data from the KDD Cup 1999 dataset, which was also used for training. Our approach differs in that we specifically focus on the NSL-KDD dataset to evaluate the feasibility of using deep learning for NIDS. Furthermore, we employ a sparse autoencoder for completely unsupervised feature learning. The authors of [24] propose a deep learning approach for network traffic analysis based on sparse autoencoders. However, instead of focusing on intrusion detection, their research concentrates on identifying anomalous protocols within TCP traffic.

3. METHODOLOGY

3.1. SELF-TAUGHT LEARNING (STL)

Self-Taught Learning (STL) is a deep learning-based classification technique that operates in two stages. The first stage, known as Unsupervised Feature Learning (UFL), focuses on constructing robust feature representations from a large volume of unlabeled data. This type of learning, free from human supervision, relies solely on the inherent structure within the data. The learned representation is then utilized in the subsequent stage to categorize labeled data. A key assumption in STL is that even if the unlabeled and labeled data originate from different distributions, there should be some underlying relationship or shared features between them. Fig. 1 presents a diagrammatic representation of the STL architecture.

Two common approaches for UFL are Gaussian Mixtures and Sparse Autoencoders. This study employed a Sparse Autoencoder for feature learning due to its simplicity and efficiency. In a Sparse Autoencoder, the roles of the traditional neural network layers (input, hidden, output) are reinterpreted. The input layer of the neural network corresponds to the output layer of the Sparse Autoencoder, while the hidden layer remains the same. Finally, the output layer of the neural network aligns with the input layer of the Sparse Autoencoder. Both the output and input layers consist of "N" nodes, while the hidden layer is composed of "K" nodes. The Sparse Autoencoder aims to reconstruct the original input data at its output layer, thereby learning a compressed and meaningful representation in the hidden layer.

The sigmoid function, $g(z)=1/(1+exp(-z))$, is used to activate the nodes in the hidden and output layers, indicated by hW, b, respectively:

$$hw, b(x) = g(Wx + b) \quad (1)$$

$$J = \frac{1}{2m}\sum_{i=1}^{m} m \left| |x_i - \hat{x}_i||^2 + \frac{\lambda}{2}\left(\sum_{k,n} W^2 + \sum_{n,k} V^2 + \sum_{k} {b_1}^2 + \sum_{n} {b_2}^2\right) + \beta \sum_{j=1}^{K} KL(\rho||\hat{\rho_j}\right) \quad (2)$$

The cost function minimized during backpropagation in a sparse autoencoder is represented by Equation (2). This function consists of three key components:

- Reconstruction Error: The primary term represents the average sum-of-squared errors between the input and reconstructed output over all "m" training data points. This term encourages the autoencoder to learn a faithful representation of the input data.
- Weight Decay: The second term incorporates weight decay, controlled by a weight decay parameter. This term helps prevent overfitting by penalizing large weights, thus promoting a smoother and more generalizable model.
- Sparsity Penalty: The first term in the equation is the sparsity penalty factor. This term, crucial for enforcing sparsity in the hidden layer, encourages most hidden units to have low average activation levels.

Equation (3) defines the sparsity penalty using the Kullback-Leibler (KL) divergence:

$$KL(\rho||\hat{\rho_j}) = \rho log\frac{\rho}{\hat{\rho_j}} + (1-\rho) log\frac{1-\rho}{1-\hat{\rho_j}} \quad (3)$$

Where the sparsity penalty term β is governed by a sparsity limited parameter "ρ" having a measure ranging 0 to 1, where parameter's value can be any number between 0, 1. The $KL(\rho||\hat{\rho_j})$ reaches its minimum value when $\rho = \rho_j$, in which j denotes the average activation of hidden unit "j" over all training inputs "x". In other words, the penalty is minimized when the average activation of each hidden unit closely matches the desired sparsity level (ρ).

Once the optimal parameters $W, b1$ are learned from the unlabeled data (xu) using the sparse autoencoder, we can generate feature representations for the labeled data (xl). This is achieved by calculating:

$a = hW, b1\ (xl)$. The modified representation of the attributes is used in the next step.

The sparsity penalty factor, appearing as the first term in the equation, ensures that the hidden layer maintains relatively low average activation levels. This factor, formally known as the Kullback-Leibler (KL) divergence, is defined in Equation (3).

The sparsity penalty term, β, is governed by the sparsity parameter "ρ," which ranges from 0 to 1. The $KL(\rho||\hat{\rho_j})$ reaches its minimum value when $\rho = \rho_j$, where "j" represents the average activation of hidden unit "j" over all training inputs "x".

We first determine the optimal parameters $W, b1$ using the sparse autoencoder on the unlabeled data (xu). Subsequently, we assess the feature representation for the labeled data (xl) using these learned parameters. This representation, denoted as a, is calculated as: $a = W, b1\ (xl)$.

3.2. WORKING OF STL

STL has been successfully incorporated into deep learning models for Network Intrusion Detection Systems (NIDS). Below is a simplified explanation of the STL workflow:

Unlabeled Data Collection: Initially, a large dataset of raw network traffic data is collected. This data encompasses various network packets and flows but lacks any labels indicating whether the traffic is benign or malicious.

Feature Extraction: This stage involves extracting relevant features from the raw network data. These features may include packet size, protocol type, source and destination IP addresses, port numbers, and other characteristics. Feature extraction enables the model to identify patterns and predict future network behavior.

Pre-training on Unlabeled Data: A deep learning model, such as an autoencoder or another type of artificial neural network, is trained on the unlabeled data. The model's objective during pre-training is to reconstruct the input data accurately, thereby learning meaningful representations by minimizing the difference between the input and its reconstruction.

Fine-tuning on Labeled Data: The pre-trained model is then fine-tuned using a smaller labeled dataset. This step may not always be necessary, depending on the size of the labeled data and the pre-trained model's performance.

Intrusion Detection: Once trained, the model functions as an intrusion detection system. During inference, the model receives new network traffic data and classifies it as either benign or malicious based on learned patterns. If the model detects any anomalies or deviations from normal behavior, it triggers an alert.

Feedback Loop: The NIDS can incorporate a feedback mechanism where the generated alerts are used to further improve the model's performance over time. This feedback loop allows for continuous learning and adaptation to new threats.

3.3. DATASET

This study utilizes the NSL-KDD dataset, a modified and refined version of the original KDD Cup 99 dataset. The KDD Cup 99 dataset, based on network traffic collected during the 1998 DARPA IDS assessment program, has undergone significant changes. The original data-

set consists of raw network data gathered over seven weeks of training and two weeks of testing. The testing data includes several attack types absent from the training data. This deliberate omission aims to simulate real-world scenarios where novel attacks, often inspired by previous ones, emerge. This characteristic enhances the dataset's ability to evaluate the accuracy of intrusion detection systems in identifying unknown threats. The NSL-KDD dataset comprises five million TCP/IP connection records for training and two million for testing.

For many years, the KDD Cup dataset served as a standard benchmark for evaluating NIDS. However, a significant drawback is the high redundancy within the dataset. A substantial portion of the records in both the training (78%) and testing (75%) sets are duplicates. This redundancy biases learning algorithms towards the most frequent attack types, resulting in poorer performance on less common but potentially more dangerous attacks. For example, a simple machine learning model achieved a minimum accuracy of 98% on the training data but only 86% on the testing data. This discrepancy makes it challenging to compare different IDSs and training methods fairly. The NSL-KDD dataset addresses these limitations. It improves upon the KDD Cup dataset in two key ways:

1. Redundancy Removal: Duplicate records are removed from both the training and testing sets.
2. Difficulty-Based Sampling: Records are categorized based on their difficulty level for learning algorithms. The NSL-KDD dataset then samples records randomly from various difficulty levels, ensuring a more balanced and representative distribution of attack types.

The training data comprises 23 traffic classes, consisting of 22 attack classes and one normal class. The test data set is more diverse, containing 39 traffic classes. These include 21 attack classes present in the training data, 16 novel attack classes, and one normal class. Each attack class falls into one of four categories based on its intended impact: Probing, Denial of Service (DoS), Remote to Local (R2L), and User to Root (U2R). Table 1 presents the distribution of normal and attack traffic within the training and testing sets.

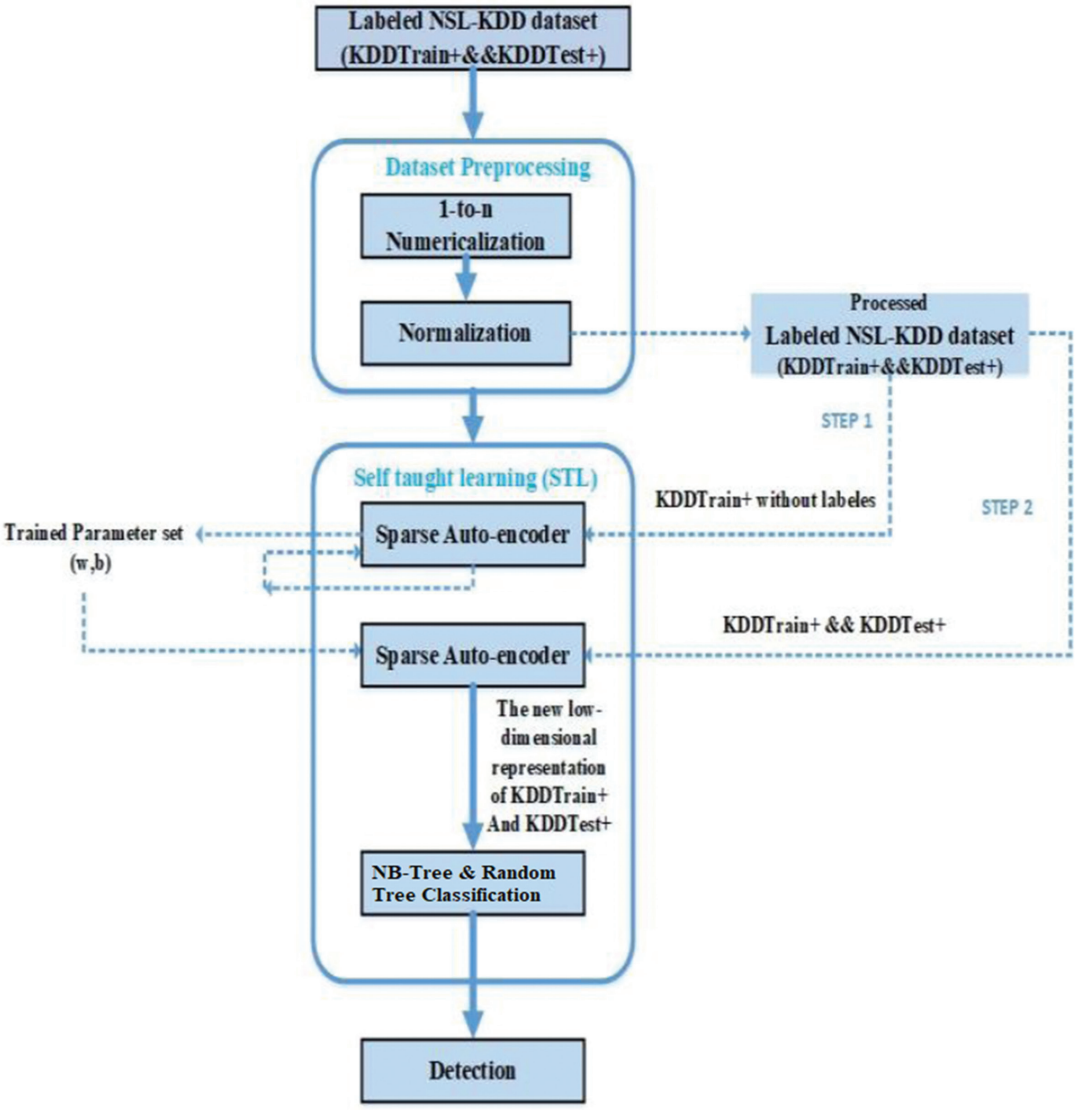

Fig. 1. STL NIDS architecture

Table 1. Distribution of Normal and Attack Traffic in the NSL-KDD Dataset

Traffic		Training	Test
Normal		67343	9711
Attack	DoS	45927	7458
	U2R	52	67
	R2L	995	2887
	probe	11656	2421

4. RESULTS AND DISCUSSION

The following describes the two methods employed to evaluate the NIDS performance. The first method utilizes the entire dataset for both training and testing, resulting in a high accuracy rate and a low false positive rate. However, this approach lacks independent evaluation. The second method addresses this limitation by splitting the dataset into separate training and testing sets. This independent evaluation, while more realistic, yields lower accuracy due to the differing conditions under which the training and testing data were collected. To ensure a comprehensive assessment, we prioritize the results obtained using the second, more realistic, method. However, for completeness, we also present the results from the first method.

4.1. PERFORMANCE ASSESSMENT, NSL IMPLEMENTATION

The dataset, as described in the previous section, contains numerous attributes, each of which can take on a range of values. Before employing self-taught learning, the dataset requires preparation. This involves converting nominal features into discrete attributes using '1-to-n encoding'. Additionally, the value of the 'num outbound cmds' feature is set to 0 for all entries in both the training and testing sets, as this feature is currently absent from the database. Following these steps, the dataset yields 121 features. Fig. 2 illustrates how the sigmoid function generates values in the output layer during the feature learning phase. As shown in the diagram, this function produces values ranging from 0 to 1.

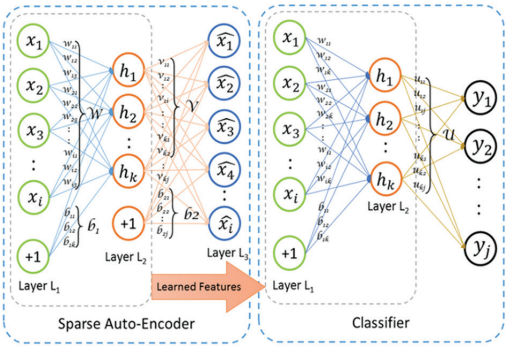

Fig. 2. STL based learning stages

4.1.1 Training phase

The classification accuracy of self-taught learning (STL) was evaluated on the training data using 10-fold cross-validation. Performance was assessed for two, five, and twenty-three class scenarios and compared against a baseline of soft-max regression (SMR) trained on the same data without prior knowledge. As illustrated in Fig 3, STL significantly outperforms SMR in the two-class classification task. However, for the five-class and twenty-three class scenarios, the performance of both methods is comparable. Furthermore, our analysis determined that STL achieves a consistently high classification accuracy exceeding 98% across all tested categorization scenarios.

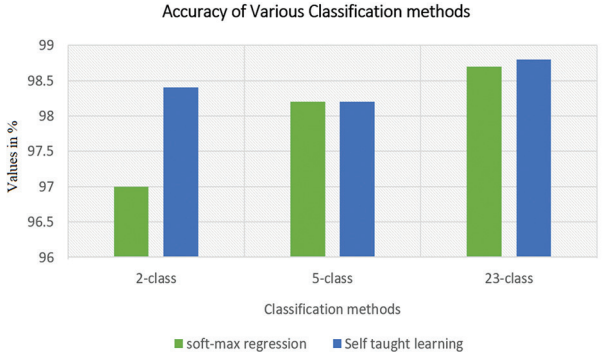

Fig. 3. Classification accuracy

Certain data categories were excluded from the 10-fold cross-validation evaluation of the five-class and twenty-three-class scenarios. Consequently, these metrics were only assessed for the two-class classification task. Our analysis revealed that STL consistently outperformed SMR across all evaluated metrics. Specifically, as depicted in Fig 4, STL achieved an F-measure of 98.84%, while SMR attained 96.79%. Notably, STL's performance on the training data approaches the highest accuracy levels reported in the literature for similar tasks.

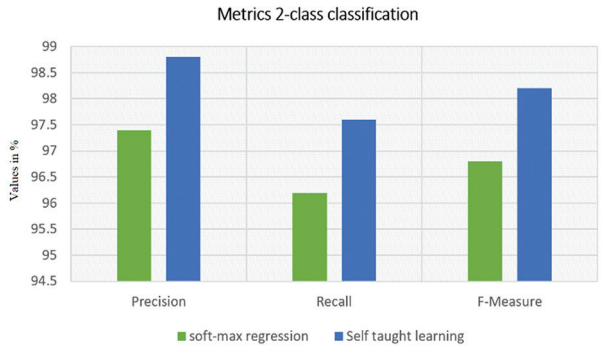

Fig. 4. Precision, recall, F-measure data

4.1.2. Testing phase

We evaluated the performance of STL and SMR on both two-class and five-class classification tasks using the held-out testing data. As illustrated in Fig. 5, STL consistently outperforms SMR in terms of accuracy. For the two-class scenario, STL achieved an accuracy of 88.39%, surpassing the 78.06% accuracy obtained by SMR. This result also compares favorably to previous studies, with the highest reported accuracy for a similar task using NB-

Tree being 82% [24]. In the five-class scenario, STL maintained its advantage with an accuracy of 79.10%, compared to 75.23% for SMR. Fig. 6 and 7 provide a detailed breakdown of the performance metrics for the five-class and two-class tasks, respectively, including F-measure, accuracy, and recall. Interestingly, while STL demonstrates superior overall accuracy in the two-class case, its accuracy (85.44%) is notably lower than that of SMR (96.56%) when considering only Figure 6. However, STL exhibits a significantly higher recall rate (95.95%) compared to SMR (63.73%), ultimately leading to a higher F-measure (90.4% for STL vs. 76.8% for SMR). This discrepancy highlights the importance of considering multiple evaluation metrics. For the five-class classification (Figure 7), the results follow a similar trend, with STL achieving a higher F-measure (75.76%) than SMR (72.14%).

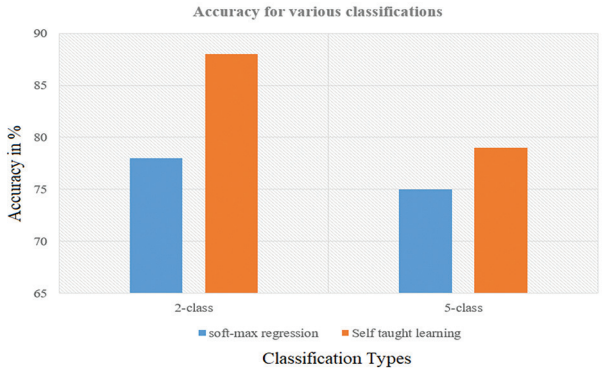

Fig. 5. Classification accuracy

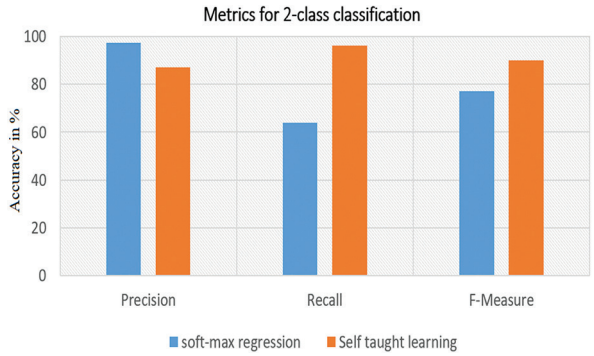

Fig. 6. An evaluation of accuracy metrics for two-class classification

Fig. 7. An evaluation of accuracy metrics for 5-class classification

Comparing Self-Taught Learning (STL) with Softmax Regression (SMR) provides researchers and practitioners valuable insights into the trade-offs between unsupervised and supervised learning for Network Intrusion Detection Systems (NIDS). STL excels in exploratory data analysis, uncovering patterns and anomalies within unlabeled data. Conversely, SMR offers a robust framework for classification when labeled data is available, enabling effective cross-situational categorization. Understanding these distinct strengths and weaknesses is crucial for selecting the most appropriate algorithm based on the specific requirements and constraints of the NIDS application.

Self-Taught Learning (STL), an unsupervised learning approach, demonstrates particular efficacy in scenarios where labeled data is limited, especially for binary anomaly detection in Network Intrusion Detection Systems (NIDS). STL's ability to independently learn meaningful representations from unlabeled data is crucial for identifying previously unseen threats, which often evade detection by supervised methods reliant on predefined labels. However, STL may exhibit limitations in multi-class classification tasks, potentially leading to lower recall rates compared to supervised learning. This stems from STL's lack of predefined intrusion classes, making it challenging to differentiate between various attack types during training. Additionally, interpreting the learned representations within STL models can be less intuitive than those in supervised learning models, potentially hindering explainability. Therefore, a thorough understanding of STL's operational characteristics and limitations, as documented in the literature, is essential for justifying its suitability and effectiveness for specific NIDS applications

Empirical studies consistently demonstrate that applying STL-based deep learning models to NIDS leads to significant improvements in intrusion detection and prevention rates. These enhancements are evident across several key aspects:

Improved Detection Accuracy: A primary evaluation metric for NIDS is the model's ability to accurately distinguish between benign and malicious network traffic. STL-based approaches consistently outperform conventional methods in this regard. This superior performance stems from their ability to leverage latent representations learned from unlabeled data, enabling the detection of novel attack patterns not encountered during training. Consequently, STL-based NIDS exhibit enhanced detection capabilities, strengthening overall network security.

Reduced False Positives: Minimizing false positives is crucial in NIDS, as excessive alerts can overwhelm security teams, leading to alert fatigue and potentially missed threats. STL-based models excel in this area due to their capacity to discern subtle anomalies within complex traffic patterns. This ability to effectively differentiate between benign and malicious events sig-

nificantly reduces false alarms, optimizing resource allocation for security teams.

Adaptability to New Threats: The dynamic nature of cybersecurity threats necessitates adaptable defense mechanisms. STL-based NIDS models possess inherent flexibility, continuously learning from incoming network data to refine their detection patterns. This adaptability enables them to effectively identify and respond to emerging attack types and evolving malicious tactics, ensuring the NIDS remains a relevant and effective security measure.

Scalability and Efficiency: Effective NIDS solutions must handle the demands of large-scale networks without compromising performance. STL-based models are well-suited for such environments, often designed with computational efficiency in mind. This allows them to analyze vast volumes of network traffic in real-time without imposing excessive overhead on system resources.

Overall, these findings highlight the substantial benefits of incorporating STL-based deep learning models into NIDS, paving the way for more robust and resilient network security solutions.

5. CONCLUSION

This paper presented an effective and adaptable deep learning-based approach for enhancing Network Intrusion Detection Systems (NIDS). The proposed NIDS leverages a sparse autoencoder for unsupervised feature learning, followed by a soft-max regression classifier for anomaly detection. The system's performance was rigorously evaluated using the NSL-KDD benchmark dataset, demonstrating its effectiveness in identifying network intrusions. Comparative analysis revealed that our NIDS outperforms existing methods for both normal and anomaly detection on the test data. While alternative approaches like Stacked Autoencoders and Deep Belief Networks, also derived from sparse autoencoders, show promise for unsupervised feature learning when combined with classifiers such as J48, NB-Tree, or Random Forest, these methods achieved superior results when applied directly to the dataset. Our experiments with Self-Taught Learning (STL) based deep learning models for NIDS highlight the significant advantages of incorporating STL. The results indicate that STL enhances network intrusion security by enabling: higher accurate detection rates, minimized false alarms, adaptive learning of new threats over time, and scalability to large networks. Future research directions include exploring techniques for effectively training STL-based NIDS while preserving data privacy. This could involve investigating privacy-preserving machine learning methods such as federated learning, differential privacy, and homomorphic encryption. These technologies can facilitate collaborative model training across distributed networks without requiring the sharing of sensitive raw data.

6. ACKNOWLEDGMENT

This work has been supported by the Universiti Kebangsaan Malaysia, Under the research grant scheme DIP 2022-021.

7. REFERENCES

[1] L. X. Ying, M. Aman, A. Hafizah, M. S. Jalil, T. M. Omar, Z. S. Attarbashi, M. A. Abuzaraida, "Malaysia Cyber Fraud Prevention Application: Features and Functions", Asia-Pacific Journal of Information Technology and Multimedia, Vol. 12, No. 2, 2023, p. 312.

[2] A. A. Ahmed, M. K. Hasan, I. Memon, A. H. M. Aman, S. Islam, T. R. Gadekallu, S. A. Memon, "Secure AI for 6G Mobile Devices: Deep Learning Optimization Against Side-Channel Attacks", IEEE Transactions on Consumer Electronics, Vol. 70, No. 1, 2024, pp. 3951-3959.

[3] F. Dehkordi, K. Manochehri, V. Aghazarian, "Internet of Things (IoT) Intrusion Detection by Machine Learning (ML): A Review", Asia-Pacific Journal of Information Technology and Multimedia, Vol. 12, No. 1, 2023, pp. 13-38.

[4] A. A. Ahmed, M. K. Hasan, N. S. Nafi, A. H. Aman, S. Islam, S. A. Fadhil, "Design of Lightweight Cryptography based Deep Learning Model for Side Channel Attacks", Proceedings of the 33rd International Telecommunication Networks and Applications Conference, Melbourne, Australia, 29 November - 1 December 2023, pp. 325-328.

[5] N. Jafri, M. M. Yusof, "Managing Data Security Risk in Model Software as a Service (SAAS)", Asia-Pacific Journal of Information Technology and Multimedia, Vol. 7, No. 1, 2018, pp. 99-117.

[6] A. K. Jakkani et al. "Design of a Novel Deep Learning Methodology for IOT Botnet based Attack Detection", International Journal on Recent and Innovation Trends in Computing and Communication, Vol. 11, No. 9, 2023, pp. 4922-4927.

[7] A. O. Alzahrani, M. J. Alenazi, "Designing a Network Intrusion Detection System Based on Machine Learning for Software Defined Networks", Future Internet, Vol. 13, No. 5, 2021, p. 111.

[8] P. Reddy, Y. Adetuwo, A. K. Jakkani, "Implementation of Machine Learning Techniques for Cloud Security in Detection of DDOS Attacks", Interna-

tional Journal of Computer Engineering and Technology, Vol. 15, No. 2, 2024, pp. 25-34.

[9] R. Kolandaisamy, K. Subaramaniam, A. B. Jalil, "A Study on Comprehensive Risk Level Analysis of IoT Attacks", Proceedings of the International Conference on Artificial Intelligence and Smart Systems, Coimbatore, India, 25-27 March 2021, pp. 1391-1396.

[10] A. Agbonyin, P. Reddy, A. K. Jakkani, "Utilizing Internet of Things (IOT), Artificial Intelligence, and Vehicle Telematics for Sustainable Growth in Small, and Medium Firms (SMES)", International Journal of Computer Engineering and Technology, Vol. 15, No. 2, 2024, pp. 182-191.

[11] Y.-W. Chen, J.-P. Sheu, Y.-C. Kuo, N. Van Cuong, "Design and Implementation of IoT DDoS Attacks Detection System Based on Machine Learning", Proceedings of the European Conference on Networks and Communications, Dubrovnik, Croatia, 15-18 June 2020, pp. 122-127.

[12] Q. Abu Al-Haija, S. Zein-Sabatto, "An Efficient Deep-Learning-Based Detection and Classification System for Cyber-Attacks in IoT Communication Networks", Electronics, Vol. 9, No. 12, 2020, p. 2152.

[13] J. Zhang, L. Pan, Q.-L. Han, C. Chen, S. Wen, Y. Xiang, "Deep Learning Based Attack Detection for Cyber-Physical System Cybersecurity: A Survey", IEEE/CAA Journal of Automatica Sinica, Vol. 9, No. 3, 2021, pp. 377-391.

[14] B. Susilo, R. F. Sari, "Intrusion Detection in IoT Networks Using Deep Learning Algorithm", Information, Vol. 11, No. 5, 2020, p. 279.

[15] T. H. Aldhyani, H. Alkahtani, "Attacks to Automatous Vehicles: A Deep Learning Algorithm for Cybersecurity", Sensors, Vol. 22, No. 1, 2022, p. 360.

[16] I. Ullah, Q. H. Mahmoud, "Design and Development of a Deep Learning-Based Model for Anomaly Detection in IoT Networks", IEEE Access, Vol. 9, 2021, pp. 103906-103926.

[17] A. A. Ahmed, W. A. Jabbar, A. S. Sadiq, H. Patel, "Deep Learning-Based Classification Model for Botnet Attack Detection", Journal of Ambient Intelligence and Humanized Computing, Vol. 13, 2022, pp. 3457-3466.

[18] V. Dutta, M. Choraś, M. Pawlicki, R. Kozik, "A Deep Learning Ensemble for Network Anomaly and Cyber-Attack Detection", Sensors, Vol. 20, No. 16, 2020, p. 4583.

[19] N. Elmrabit, F. Zhou, F. Li, H. Zhou, "Evaluation of Machine Learning Algorithms for Anomaly Detection", Proceedings of the International Conference on Cyber Security and Protection of Digital Services (Cyber Security), Dublin, Ireland, 15-19 June 2020, pp. 1-8.

[20] A. Khaleghi, M. S. Ghazizadeh, M. R. Aghamohammadi, "A Deep Learning-Based Attack Detection Mechanism Against Potential Cascading Failure Induced by Load Redistribution Attacks", IEEE Transactions on Smart Grid, Vol. 14, No. 6, 2023, pp. 4772-4783.

[21] J. Shareena, A. Ramdas, H. AP, "Intrusion Detection System for IoT Botnet Attacks Using Deep Learning", SN Computer Science, Vol. 2, No. 205, 2021, pp. 1-8.

[22] L. Liu, P. Wang, J. Lin, L. Liu, "Intrusion Detection of Imbalanced Network Traffic Based on Machine Learning and Deep Learning", IEEE Access, Vol. 9, 2020, pp. 7550-7563.

[23] J. Lansky, S. Ali, M. Mohammadi, M. K. Majeed, S. T. Karim, S. Rashidi, M. Hosseinzadeh, A. M. Rahmani, "Deep Learning-Based Intrusion Detection Systems: A Systematic Review", IEEE Access, Vol. 9, 2021, pp. 101574-101599.

[24] G. Kocher, G. Kumar, "Machine Learning and Deep Learning Methods for Intrusion Detection Systems: Recent Developments and Challenges", Soft Computing, Vol. 25, 2021, pp. 9731-9763.