AI-Driven Anomaly Prediction in Encrypted Network Traffic

Sina Ahmadi
National Coalition of Independent Scholars (NCIS), Seattle, USA

***Abstract*—The rapid growth of computer networks has increased demand for more sophisticated tools for network traffic analysis and monitoring. The increasing reliance on networks has amplified the need for robust security and intrusion detection mechanisms. Numerous studies have sought to develop efficient methods for fast and accurate intrusion detection, each addressing the challenge from different perspectives. A common limitation among these approaches is their reliance on expert-engineered features extracted from network traffic. This dependency makes them less adaptable to emerging attack techniques and changes in normal traffic patterns, often resulting in suboptimal performance. In this study, we propose a method leveraging recent advancements in artificial neural networks and deep learning, specifically using recurrent neural networks (RNNs), for network traffic analysis and intrusion detection. The key advantage of this approach is its ability to autonomously extract features from network traffic without human intervention. Trained on the ISCX IDS 2012 dataset, the proposed model achieved an accuracy of 0.99 in distinguishing between malicious and normal traffic.**

Keywords—Machine learning; deep learning; recurrent neural networks; intrusion detection

I. Introduction

With the expansion of communication technology, computer security has become an essential part of everyday life. The widespread use of computer systems has increased the importance of security in computer systems, as well as the increasing number of threats, such as malware and mobile phone threats. Despite ongoing research in cybersecurity, security challenges remain unresolved. The rapid advancement of computer networks and the widespread interconnection of systems have heightened concerns, particularly in Internet security, where numerous devices are interconnected, increasing vulnerability risks.

Since the Internet Protocol (IP) is not designed to ensure the highest security standards, network administrators are faced with a huge flood of intrusion attempts by people with malicious intent [1]. According to statistics provided by Symantec [2] on Internet security, more than three billion malware attacks were reported in 2010, and the number of denial-of-service attacks has increased significantly. Anomaly detection plays a crucial role in data analysis. The goal of anomaly detection is to find anomalous data in a dataset. This has been widely studied in statistics and machine learning, as anomalies can trigger critical and potentially harmful events, making their detection essential.

For example, in computer networks, unusual traffic can mean infiltration of a computer where a compromised system transmits data to an unauthorized entity [3]. Conventional approaches for anomaly detection include deep packet inspection, statistical, and machine learning. Statistical and deep packet inspection methods do not perform well against emerging attacks in real-life scenarios.

Moreover, these methods are highly dependent on expert knowledge. Research in this field aims to develop anomaly detection systems that not only perform well but also operate autonomously while maintaining their high accuracy [4]. Understanding the characteristics of modern network traffic requires a deep knowledge of the structure and dynamics of Internet traffic, which also plays a crucial role in managing and monitoring Internet service provider (ISP) networks [5]. Although the rationale for intrusion detection in the context of computer networks is clear, several practical problems still exist with intrusion detection systems (IDS) [6]. One of the most pressing concerns in network security is the increasing sophistication and frequency of cyberattacks, coupled with evolving user behavior and software usage patterns. Traditional models that rely on historical attack data often become obsolete as new threats emerge, highlighting the need for adaptive intrusion detection systems that can be continuously updated to maintain effectiveness [7].

A significant challenge in intrusion detection is packet encryption and encapsulation. Encryption alters network traffic characteristics, making it difficult to differentiate between normal and malicious activity. Traditional intrusion detection methods rely on identifiable traffic patterns, which encryption effectively conceals, thereby reducing their accuracy. Additionally, packet encapsulation and encryption make traditional expert-defined feature extraction ineffective. The randomized nature of encrypted traffic prevents manual feature engineering, emphasizing the necessity for fully automated systems capable of independently capturing complex patterns in encrypted traffic [8].

Another important aspect is the high-speed nature of modern networks, which also makes real-time intrusion detection difficult. Anomaly detection, or outlier detection, examines normal data trends in a network and marks any deviations as a potential threat. One widely used approach is anomaly-based intrusion detection, where normal network traffic is first recorded, and any data deviating from learned normal behavior is classified as an anomaly [9].

Among deep learning techniques, deep convolutional networks, stacked self-encryptors, and recurrent neural networks (RNNs) have demonstrated tremendous success. These methods have been continuously used for various applications such as voice processing [10], computer vision [11], natural language processing [12], network traffic classification [13], and cyberattack detection [14], delivering strong performance. It is clear from different studies that deep learning performance is far beyond the performance of traditional methods in the aforementioned applications. Perhaps the greatest advantage of deep learning is the autonomous extraction and learning of features [15], which eliminates the need for manual feature engineering. There are other deep learning methods available, including unsupervised and quasi-supervised methods, which increase adaptability even further.

In the next sections, we will discuss two particular types of networks that were used for this study, and we will also describe the key principles for these networks in more detail.

The ability to recognize temporal order and sequence is what makes recurrent neural networks (RNNs) one of the most powerful models in deep learning. It also makes them especially good at natural language processing and network traffic processing. These networks are suitable to learn patterns in sequence data such as genes, texts, and time sequences [16].

II. Research Background

Ji et al. (2024) conducted a systematic review of AI-driven anomaly detection methods for encrypted network traffic. The study established key research questions and selected relevant literature based on specific eligibility criteria. The findings confirmed that a variety of AI techniques have been applied to anomaly detection in encrypted traffic. While some of these methods resemble those used for unencrypted traffic, others employ distinct approaches tailored to the challenges posed by encryption [17].

Zeng et al. [18] demonstrated the importance of user engagement regarding the effectiveness of AI-based cybersecurity systems by discussing the concept of technology acceptance and human-AI interaction. The study proposes building a security-conscious culture via ongoing education and skills training, which can help to better protect smart cities against new forms of cyber-attacks. The framework functions as a base for future empirical studies as well as an important tool for policymakers and city planners concerned with the digital borders of smart cities of the future.

Aakash [19] explores the use of machine learning to enhance network monitoring capabilities. The study outlines key network monitoring tools and examines their current approaches to anomaly detection. It then discusses machine learning techniques for developing predictive models based on historical data. Additionally, the research proposes a framework for integrating trained models as extensions to existing monitoring tools. The findings indicate that AI-driven methods offer more precise and automated anomaly detection compared to traditional techniques.

Alwhbi et al. [20] examined the modern application of machine learning to encrypted traffic analysis, as well as its classification, through a comprehensive survey. These goals break down into two objectives. First, the overall workflow is illustrated to clarify how machine learning is used to analyze and classify encrypted network traffic. And second, perform a literature review of the most sophisticated approaches to traffic analysis. This research attempts to shed light on contemporary practices and give directions for future works in the realm of encrypted traffic analysis, and especially, in the field of machine learning.

III. Proposed Methods

Our proposed methods use deep learning and specifically a deep recurrent neural network (RNN) architecture for intrusion detection in network traffic. As highlighted in the introduction, one of the key challenges in this domain is the extraction of appropriate features for attack detection. Traditional methods rely on features extracted by an expert, which limits their ability to detect new and evolving attacks. Such models that rely on manual feature selection often face generalization problems and bear the risk of failing against zero-day attacks or attacks that slightly deviate from established patterns. Deep learning solves this issue by extracting features automatically from raw data. In this case, regular traffic patterns and attacks are provided as input to a deep learning model so that it can learn on its own. Among deep neural network architectures, recurrent neural networks (RNNs) are particularly effective for sequence-based learning, making them well-suited for analyzing network traffic, which consists of sequential packet flows. RNNs can process network flows as sequential data, allowing them to extract long-term dependencies within the traffic.

We designed and evaluated three different approaches based on recurrent neural networks, each of which is explored in detail:

1) Anomaly detection using a recurrent self-encryptor: In this approach, a recurrent self-encryptor neural network model is trained on normal network traffic data. In case new data is provided to the system and the reconstruction error exceeds a certain threshold, the data will be classified as an attack.

2) Anomaly detection using an RNN: In this approach, both normal and attack traffic are incorporated into an RNN for training. The model is trained as a classifier to differentiate between normal and attack traffic based on the learned patterns.

3) Anomaly detection using multiple learning (Hybrid): This approach considers the above methods in tandem. A recurrent self-encryptor network is first trained, and the output of the cryptographic part of the self-encryptor network is then fed into a classifier neural network, which learns to classify normal and attack traffic. These two networks are connected and trained simultaneously, leveraging both anomaly detection and classification techniques for enhanced accuracy.

The novelty of this work lies in integrating a self-encrypting autoencoder with a supervised RNN classifier in a hybrid architecture. Unlike conventional deep learning methods that treat anomaly detection and classification separately, this dual-task framework allows simultaneous feature learning for both unsupervised anomaly detection and supervised classification, optimizing performance for encrypted traffic scenarios.

In the subsequent sections, we will examine each of the proposed methods and discuss their implementation and effectiveness in network intrusion detection.

A. *Anomaly Detection by Recursive Self-Encryptor*

In this method, the objective is to train a recurrent self-encryptor for anomaly detection. As explained in Chapter One, anomaly detection involves training a model on normal network data so that it learns to extract key features of normal traffic. Any data that deviates significantly from this learned pattern is classified as an anomaly. One way to achieve this is by using a self-encryptor, which is trained to reconstruct normal data. Once trained, if the network encounters abnormal data, its reconstruction error will be significantly higher compared to normal traffic. By defining a threshold, data with a reconstruction error above this limit is identified as an anomaly.

Intrusion detection in networks can be effectively modeled as an anomaly detection problem. Given that recurrent neural networks (RNNs) are particularly suited for processing sequential data, we implemented a self-encrypting network composed of LSTM layers. The general architecture of this recurrent self-encrypting network is shown in Fig. 1.

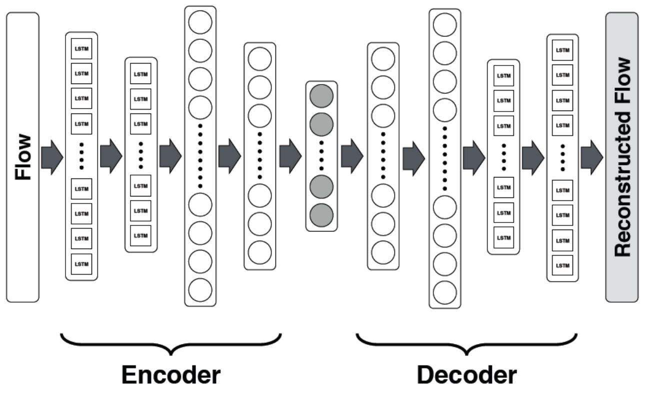

Fig. 1. General model of recursive self-encryption.

The designed neural network consists of two primary components: the encryption part and the decryption part. The encryption section is responsible for compressing the input data, while the decryption section reconstructs the original data. The encryption part begins with two LSTM layers, which extract long-term dependencies from the network stream. The output of these LSTM layers is then passed through a smoothing layer, preparing the data for input into fully connected layers. The two fully connected layers further reduce data dimensionality and extract higher-level abstract features.

Once the encryption process is completed, the decryption phase begins, where the model reconstructs the original data. The decryption section has a structure mirroring the encryption part, starting with two fully connected layers, followed by two LSTM layers that reconstruct long-term dependencies and restore the original traffic data. If the network can successfully reconstruct normal network data, it can serve as a robust anomaly detection system. To prevent overfitting between the fully connected layers, we incorporate random dropout layers. The full details of this implemented model for intrusion detection will be discussed in the following sections.

B. *Intrusion Detection by a Recurrent Neural Network*

Our second proposed solution is to train a recurrent neural network for classification. This network utilizes multiple recurrent layers of the LSTM type, where sequence return is active, meaning that each LSTM cell produces an output that is aggregated and passed as input to the next layer. Following the LSTM layers, the network includes a series of fully connected layers, and finally, two neurons with a smooth maximum activation function, with each neuron representing either normal or attack traffic label. The network learns to extract features from the sequence of packets in a flow, gradually refining these features into more complex features, to distinguish between normal and attack traffic. The structure of this approach is illustrated in Fig. 2.

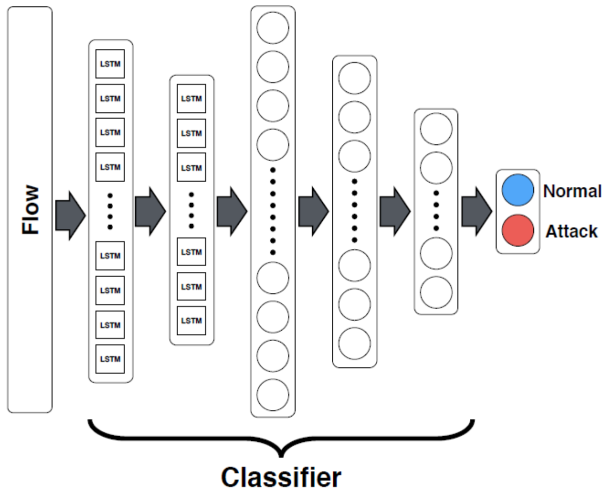

Fig. 2. Attack detection by a recurrent neural network.

The designed network begins with two LSTM layers, which extract long-term dependencies from the neural network input. The output of these layers is then processed by a smoothing layer before being passed through four fully connected layers, which refine the extracted features by extracting more abstract features and enhance the model's ability to classify traffic. The final output is processed through a smooth maximum function with two neurons, each corresponding to normal or attack traffic. To prevent overfitting between the fully connected layers, we incorporate random dropout layers. Further details on the implementation and evaluation of this model for intrusion detection are discussed in Section V.

C. *Intrusion Detection by Multiple Learning*

In the third approach, a deep self-encrypting recurrent neural network is first built, followed by the integration of a fully connected network into its middle layer. This additional network is designed to classify traffic into attack or normal categories.

Both networks are trained simultaneously with their errors propagating through the system to refine their predictions. This multi-task learning approach allows the recurrent neural network to extract generalized features that can both reconstruct network traffic and distinguish between normal and malicious activity at the same time. The high-dimensional nature of byte-level packet sequences and the temporal dependencies inherent in traffic flows necessitate deeper architectures. Multiple LSTM layers are essential to capture long-range patterns in encrypted traffic, while dense layers refine the latent representations for more accurate classification. Simpler models lack the capacity to extract such hierarchical features.

Solving these two tasks together enhances the performance of both. The auto-encrypting network learns comprehensive traffic features, while the classification network guides feature extraction toward detecting anomalies. These two combined are more comprehensive, as they capture more complete flow information, allowing them to reconstruct network traffic while also identifying attacks. As a result, this model performs more effectively in real-world scenarios, adapting better to new types of traffic and emerging attacks. An overview of this approach is illustrated in Fig. 3.

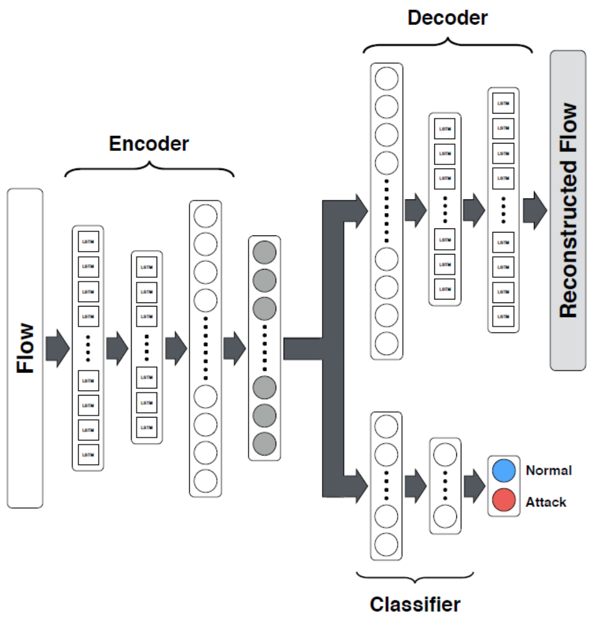

Fig. 3. Attack detection by multiple learning.

Like the first approach, this neural network consists of an auto-encrypting component that processes network flows. The input flows are first passed through two LSTM layers, which extract long-term dependencies from the sequence data. The processed data is then flattened and passed through three fully connected layers that reduce dimensionality while preserving meaningful patterns. The output of this section is then fed into two separate neural networks. One of these networks serves as a decoder, which mirrors the structure of the encrypting network. It consists of two fully connected layers, whose output is fed into two LSTM layers that are responsible for generating long-term dependencies and reconstructing the original data. Meanwhile, the second network is responsible for classification. This network consists of three fully connected layers, whose output is fed into a maximum smoothing layer with two neurons, each representing either the normal traffic or attack classes. To prevent overfitting between the fully connected layers, we apply random dropout layers. By combining self-encrypting and classification in this architecture, the model can both learn meaningful traffic representations and effectively detect anomalies in encrypted and unencrypted network traffic.

A major advantage of this approach is its adaptability. If the model misclassifies normal traffic as an attack or vice versa, the misclassified samples can be added to the training set, and the network can be retrained over several epochs. This self-improving mechanism ensures that the model continuously adapts to new traffic patterns and emerging threats, enhancing its ability to detect previously unseen attacks.

IV. Results

This section highlights the procedures carried out for data preprocessing and preparing the dataset for training and evaluation. Afterwards, three proposed methods will be discussed, followed by an examination of the outputs from the methods and the evaluation of the methods' effectiveness.

A. *Preprocessing*

As mentioned in the previous section, this study uses the ISCX IDS 2012 dataset, which contains PCAP files with both normal and attack network traffic [21]. The preprocessing step was important to clean and structure the dataset for training and evaluation, resulting in a consistent form to be presented as input to the neural network.

The first stage involved capturing network flows from the PCAP files. This was done by determining the source and destination IP address pairs and their relative port numbers. After extraction, the corresponding flows were tagged as normal or attack traffic based on the information available in the ISCX Laboratory's metadata.

Then, each packet within a flow was processed at the byte level, with each byte serving as an input feature for the neural network.

While flow-level modeling provides contextual information across an entire session, we adopt a packet-level approach to focus on fine-grained byte-level patterns. In encrypted traffic, metadata and payload features are heavily obfuscated, making it critical to extract local-level byte transitions. Future work may explore combining both packet and flow-level models to balance context and granularity.

The base-10 value of each byte was computed and normalized by dividing it by 255, ensuring that feature values were scaled between 0 and 1. Additionally, the time interval between each packet and its preceding packet was calculated and incorporated into the feature vector.

Since neural networks require fixed-size inputs, but network traffic varies in length, standardizing stream and packet lengths was necessary. As shown in Fig. 4, each stream was decided to be set to a fixed length of 100 packets. If a stream contained more than 100 packets, the excess packets were removed, whereas if it contained fewer, zero-padding was applied to ensure uniformity.

Similarly, each packet was limited to 200 bytes, consistent with prior research [5], which shows that most informative features in encrypted traffic, including TLS handshake and early payload indicators, are embedded within the first 200 bytes. Fig. 5 illustrates the length of the packets within the streams. Any packets exceeding this length were truncated, while shorter packets were zero-padded to maintain consistency. Choosing other packet sizes (e.g., 1,500 bytes) would significantly increase memory overhead and training time without measurable gains in accuracy.

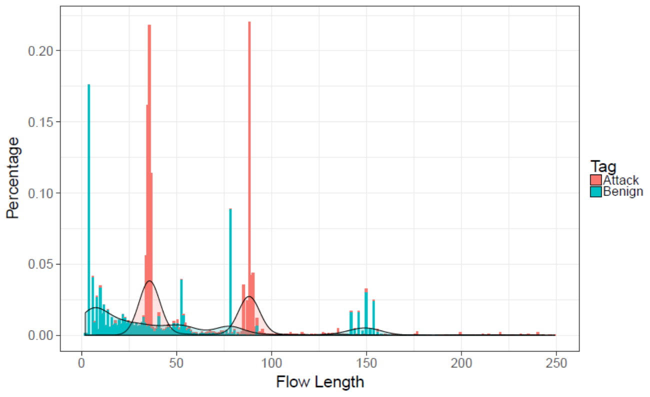

Fig. 4. Graph of the number of packets within each flow.

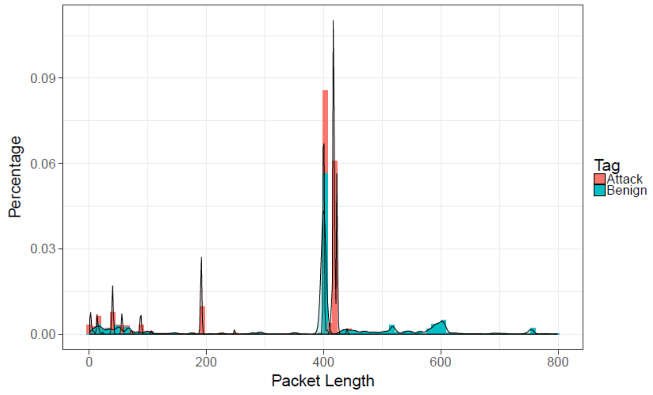

Fig. 5. Packet size diagram.

Deep neural networks are capable of learning highly complex features, but this can sometimes lead to overfitting on non-informative attributes. To prevent the model from extracting misleading patterns, non-relevant fields such as IP addresses and checksums were zeroed out. These fields do not contribute meaningfully to intrusion detection and could introduce unwanted biases into the model.

After completing these preprocessing steps, each network stream was stored as a separate file, with each line representing the details of a single packet within that stream. This structured format ensures that the neural network processes sequential packet data efficiently, preserving the temporal relationships within each stream. By following these steps, the dataset was standardized and optimized, allowing the deep learning model to effectively detect patterns in both normal and attack traffic.

To ensure robust evaluation, the ISCX IDS 2012 dataset was partitioned into three distinct subsets: 70% for training, 15% for validation, and 15% for testing. The splitting was done at the flow level to prevent data leakage across sets. The model was trained using the training set, hyperparameters were tuned using the validation set, and final performance was measured on the unseen test set. This approach ensures generalization and prevents overfitting to specific traffic patterns.

B. Anomaly Detection by Self-Encrypting Network

As discussed in the previous section, if a self-encrypting network that is trained on normal network data learns to accurately reconstruct normal traffic patterns, when presented with a new stream, if the reconstruction error exceeds a predefined threshold, the stream is identified as an anomaly. To implement this approach, we designed a recurrent self-encrypting network, the architecture of which is shown in Fig. 6.

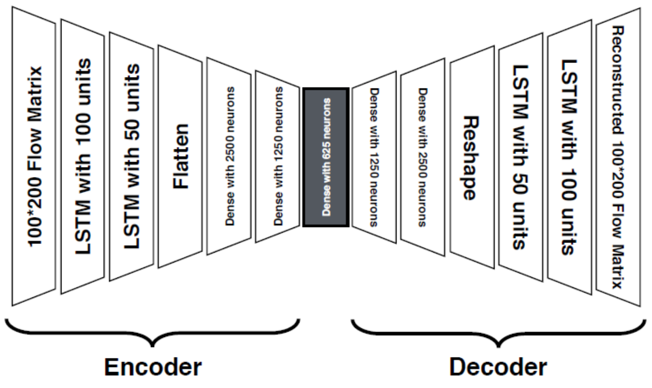

Fig. 6. Self-encrypting network architecture.

Each self-encrypting neural network consists of two main components: the encryptor and the decryptor. As illustrated in Fig. 6, the network begins with two LSTM layers responsible for extracting long-term dependencies from the input stream. The extracted features are then passed through a smoothing layer, followed by three fully connected layers with 2,500, 1,250, and 650 neurons, which reduce dimensionality and extract meaningful features for data reconstruction. The decryption network follows an inverse structure of the encryption network. It starts with two fully connected layers (1,250 and 2,500 neurons), after which the data is processed through two LSTM layers that reconstruct the original traffic patterns by capturing long-term dependencies between packets. To mitigate overfitting, random dropout layers were applied between the fully connected layers. Given the large volume of normal network data required for training, we randomly selected a subset from each protocol, totaling 30,000 samples, always keeping a 20% reserved for validation. Additionally, 10,000 samples (including both attack and normal traffic) were used for testing. The reduction in reconstruction error during training is depicted in Fig. 7.

Upon evaluating the model with test data, we calculated reconstruction errors, as shown in Fig. 8, for both normal and attack traffic. However, as seen in Fig. 8, the reconstruction error distributions for attack and normal traffic are too close, making them difficult to separate with a fixed threshold. The optimal threshold determined for this dataset was 975, based on which the test data was classified.

The results, presented in Table I, indicate that this approach did not achieve high accuracy. The primary limitation was the insufficient volume of normal network data for training. Due to hardware constraints, we were only able to train the model on 30,000 normal flows, which impacted its performance. Future improvements should focus on scaling the dataset and enhancing feature extraction techniques to improve detection accuracy.

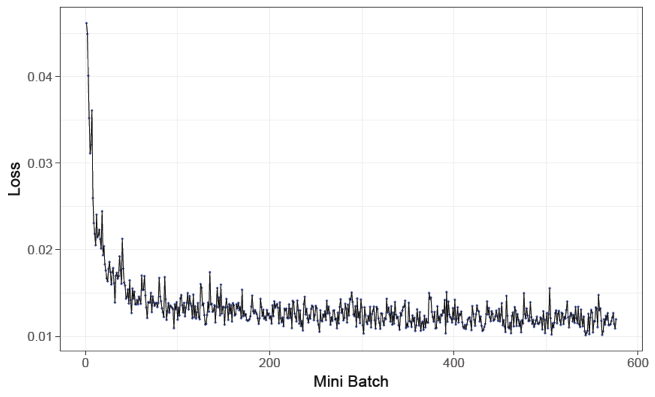

Fig. 7. Reduction of reconstruction error during training.

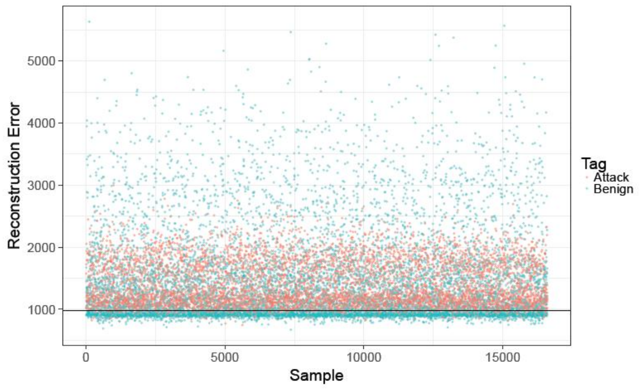

Fig. 8. Test data reconstruction error.

TABLE I. TEST DATA CLASSIFICATION RESULTS WITH SELF-ENCRYPTING NEURAL NETWORK

Class	Precision	Recall	F1-Score
Normal	0.74	0.61	0.58
Attack	0.52	0.95	0.67
Total	0.74	0.61	0.58

C. Intrusion Detection by a Recurrent Neural Network

In this method, we aim to train a classifier to distinguish between attack and normal traffic. To achieve this, we developed a recurrent neural network (RNN) with LSTM layers and trained it using all available attack data along with 20,000 normal flows, ensuring coverage of all protocols present in the dataset. Fig. 9 shows the network architecture. As depicted in this figure, this network architecture consists of nine layers designed to effectively capture sequential patterns in network traffic and improve classification accuracy.

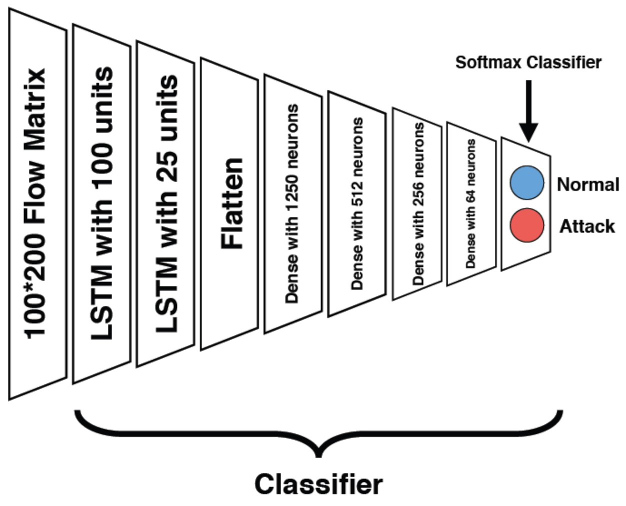

Fig. 9. Recurrent neural network structure.

First, the data is fed into two LSTM layers, which consist of 100 and 50 neurons, respectively. These are responsible for extracting long-term dependencies from the data. The output of the above layers goes to the smoothing layer and is then fed as input into four fully connected layers consisting of 1250, 512, 256, 64 neurons to extract abstract features from it. Then, the output of the fully connected layers is fed into a smooth maximum with two neurons, which are responsible for detecting normal versus attack traffic. Random dropout layers are placed between the fully connected layers with a probability of 25% to prevent overfitting. Fig. 10 depicts the chart of the reduction of the classification error of the training data. In general, the best way to find the most suitable neural network for training data is to perform a complete search on the parameters of the neural network, such as the number of neurons in the layers and the number of layers, etc. However, since our neural network is of the recurrent type and has a large size, training this network takes a lot of time, and it was not feasible to perform a comprehensive search on its parameters. Through experimentation with logically structured architectures based on prior research in this field, we developed a neural network design that demonstrated strong practical performance. After training the model, we evaluated its effectiveness using test data, with the corresponding evaluation metrics presented in Table II.

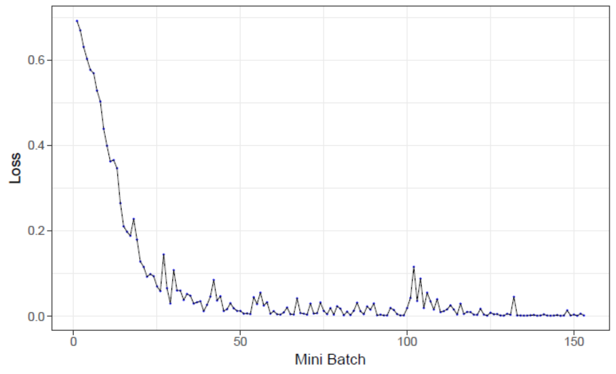

Fig. 10. Reduction of classification error during training.

TABLE II. RESULTS OF CLASSIFICATION OF TEST DATA BY THE NEURAL NETWORK

Class	Precision	Recall	F1-Score
Normal	0.98	1.0	0.99
Attack	1.0	0.97	0.99
Total	0.99	0.99	0.99

As shown in Table II, the classifier demonstrated high accuracy. To enhance processing speed, we optimized the network by reducing its size, resulting in a more efficient architecture, as illustrated in Fig. 11. This modification significantly improved the training and inference speed compared to the original model.

This change, as shown in Table III, however, led to a decrease in accuracy. The trade-off between speed and accuracy highlights the need for further optimizations to balance efficiency and detection accuracy.

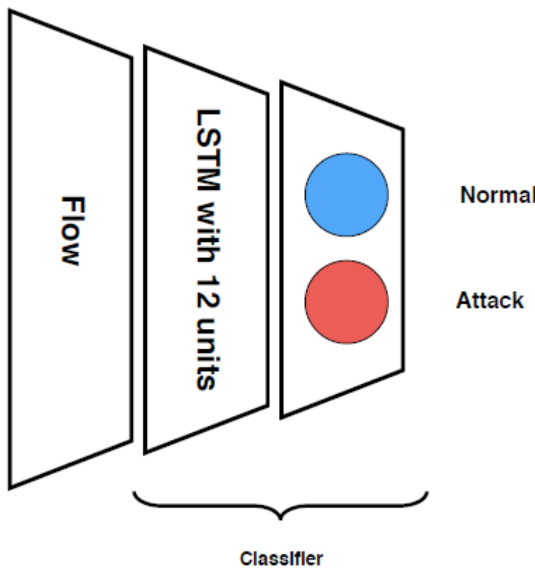

Fig. 11. Reduction of classification error during training.

TABLE III. RESULTS OF CLASSIFICATION OF TEST DATA IN THE REDUCED CLASSIFIER NEURAL NETWORK

Class	Precision	Recall	F1-Score
Normal	0.95	0.94	0.95
Attack	0.93	0.95	0.94
Total	0.94	0.94	0.94

D. Evaluation

As demonstrated in the previous section, the proposed methods achieved high accuracy in detecting attacks using the ISCX IDS 2012 dataset. To further evaluate their performance, we simulated 390,000 attack flows across HTTPS, HTTP, DNS, and FTP protocols using the 111-Spirent device. From these, 7,000 attack flows were randomly selected and added to the training set, while another 40,000 flows were randomly selected from the remaining flows to assess the multi-layer neural network model.

The evaluation results showed an attack detection accuracy of 0.99, confirming the model's effectiveness. However, one limitation is processing speed, as the neural network requires 0.039 seconds per flow. While this speed is sufficient for certain applications, real-world intrusion detection systems demand faster processing, making computational efficiency an area for further optimization.

Unlike prior approaches relying solely on handcrafted features or static classifiers, our method combines both reconstruction-based and classification-based techniques within a hybrid deep learning architecture. This dual-model approach demonstrated F1 scores of 0.99, showing significant improvement over baseline RNNs and self-encoder models in similar datasets. Table IV shows the results of the evaluation.

TABLE IV. RESULTS OF CLASSIFICATION OF TEST DATA IN MULTIPLE NEURAL NETWORKS

Class	Precision	Recall	F1-Score
Normal	0.99	0.99	0.99
Attack	0.99	0.99	0.99
Total	0.99	0.99	0.99

E. Implementation Tools

We utilized the dpkt library in Python for network data processing. All neural networks and deep learning models were implemented using the Keras library [6], which operates on TensorFlow [22]. To evaluate the performance of our classifiers, we employed the scikit-learn machine learning library [23].

For hardware acceleration during model training, we used an NVIDIA GeForce GTX 1080 GPU with 8GB of memory, significantly improving computational efficiency.

V. CONCLUSION

This study addresses the issue of applying deep learning in anomaly detection. First, we reviewed the literature on the subject and the research conducted on each of the parameters discussed in this area. Our analysis of existing research identified both key focus areas and underexplored domains with the potential for innovative solutions. One such area is automatic feature extraction from both the header and payload of network packets, including encrypted traffic such as HTTPS. Building on this, we proposed three deep learning-based approaches for intrusion detection in computer networks, all of which automatically extract features from packet headers and payloads. In this study, we utilized the latest available dataset and further enhanced it by incorporating additional attack samples to improve its diversity and robustness. The proposed methods were then rigorously evaluated using established performance metrics.

Looking ahead, we plan to pursue the following research directions:

A. Optimizing Data Reduction Methods

Our first goal is to enhance the data reduction process. Currently, flows are categorized based on their protocol, and a subset of packets is randomly selected from each protocol.

While this is a straightforward approach, it can be improved with a more intelligent selection strategy. For instance, we can extract handcrafted features from network flows, cluster the data based on these features, and eliminate clusters that are least relevant to attack detection.

B. Developing a Standardized Dataset

One of the most impactful contributions in this field would be the creation of a comprehensive, standardized dataset.

While ISCX IDS 2012 is dated, it remains widely used due to its balance of normal and attack traffic, packet-level granularity, and availability of labels. Its consistent structure allows comparative benchmarking, but it has several limitations:

- Limited protocol diversity, requiring an expansion to reflect real-world traffic more accurately.
- Insufficient attack variety, with attack traffic constituting only a small fraction of the dataset, necessitating better balance between normal and malicious traffic.
- High redundancy, as the dataset contains a large number of repetitive flows that could be filtered out to retain only the most informative samples.
- As ISCX IDS 2012 was collected over a decade ago, it is less representative of modern network traffic patterns and attack techniques.

Given these limitations, we plan to develop a new standardized dataset with greater protocol diversity, an improved balance of normal and attack traffic, and a reduced volume of redundant data, ensuring its effectiveness for future research in intrusion detection.

Future work will validate the model across multiple datasets (e.g., CIC-IDS2017, TON_IoT) to assess generalizability and adaptability to modern encrypted traffic beyond ISCX.

C. Enhancing Detection of Multi-Stream Attacks (DDoS Detection)

As outlined in previous sections, our current approach processes individual network streams, making it ineffective in detecting attacks that involve multiple concurrent streams, such as Distributed Denial-of-Service (DDoS) attacks. Since DDoS attacks are a significant category of cyber threats, developing a detection mechanism for them is a key focus of our future work.

One possible approach is to aggregate traffic directed to a specific host within a predefined time window and use this as input data. By analyzing traffic patterns over time, we aim to design a solution that can effectively identify DDoS attacks and other multi-stream-based intrusions.

D. Leveraging Generative Models for Attack and Normal Traffic Simulation

Another promising research direction is the development of a generative model for both attack traffic and normal network data. Such a model would learn the probability distribution of real-world network traffic and be capable of generating synthetic attack flow samples and normal traffic patterns.

This approach offers several advantages:

- Enhances dataset diversity by generating realistic attack scenarios, improving model robustness.
- Facilitates zero-day attack detection, as the generative model can create unseen attack variations, allowing the intrusion detection system to generalize better.
- Reduces data collection limitations, enabling continuous updates to training data without relying solely on real-world attack logs.

By pursuing these advancements, we aim to develop a highly adaptive intrusion detection system capable of automatically extracting attack features and effectively detecting zero-day attacks in real-world network environments.

REFERENCES

[1] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," Journal of Network and Computer Applications, vol. 60, pp. 19–31, 2016.

[2] M. Lotfollahi, R. S. H. Zade, M. J. Siavoshani, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," CoRR, vol. abs/1709.02656, 2017. [Online]. Available: http://arxiv.org/abs/1709.02656

[3] J. Khalife, A. Hajjar, and J. Diaz-Verdejo, "A multilevel taxonomy and requirements for an optimal traffic-classification model," International Journal of Network Management, vol. 24, no. 2, pp. 101– 120, 2014.

[4] H. Park, S.-h. Shin, B.-h. Roh, and C. Lee, "Identification of hosts behind a nat device utilizing multiple fields of ip and tcp," in Information and Communication Technology Convergence (ICTC), 2016 International Conference on. IEEE, 2016, pp. 484–486.

[5] H. Alizadeh and A. Zúquete, "Traffic classification for managing applications' networking profiles," Security and Communication Networks, vol. 9, no. 14, pp. 2557–2575, 2016.

[6] G. Sajeev and L. M. Nair, "Laser: A novel hybrid peer to peer network traffic classification technique," in Advances in Computing, Communications and Informatics (ICACCI), 2016 International Conference on. IEEE, 2016, pp. 1364–1370.

[7] Y.-n. Dong, J.-j. Zhao, and J. Jin, "Novel feature selection and classification of internet video traffic based on a hierarchical scheme," Computer Networks, vol. 119, pp. 102–111, 2017.

[8] W. De Donato, A. Pescapé, and A. Dainotti, "Traffic identification engine: an open platform for traffic classification," IEEE Network, vol. 28, no. 2, pp. 56–64, 2014.

[9] M. Stevanovic and J. M. Pedersen, "Detecting bots using multi-level traffic analysis," International Journal on Cyber Situational Awareness (ijcsa), 2016.

[10] Đ. T. Grozdić, S. T. Jovičić, and M. Subotić, "Whispered speech recognition using deep denoising autoencoder," Engineering Applications of Artificial Intelligence, vol. 59, pp. 15–22, 2017.

[11] W. Yu, K. Yang, H. Yao, X. Sun, and P. Xu, "Exploiting the complementary strengths of multi-layer cnn features for image retrieval," Neurocomputing, vol. 237, pp. 235–241, 2017.

[12] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for natural language processing," arXiv preprint arXiv:1606.01781, 2016.

[13] M. Lotfollahi, R. S. H. Zade, M. J. Siavoshani, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," CoRR, vol. abs/1709.02656, 2017. [Online]. Available: http://arxiv.org/abs/1709.02656

[14] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, “A survey of deep learning-based network anomaly detection,” Cluster Computing, Sep 2017. [Online]. Available: https://doi.org/10.1007/s10586-017-1117-8

[15] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016, http://www. deeplearningbook.org.

[16] G. Dorffner, “Neural networks for time series processing,” in Neural network world. Citeseer, 1996.

[17] Ji, I. H., Lee, J. H., Kang, M. J., Park, W. J., Jeon, S. H., & Seo, J. T. (2024). Artificial Intelligence-Based Anomaly Detection Technology over Encrypted Traffic: A Systematic Literature Review. Sensors, 24(3), 898. https://doi.org/10.3390/s24030898

[18] Heng Zeng, Manal Yunis, Ayman Khalil, Nawazish Mirza,(2024). Towards a conceptual framework for AI-driven anomaly detection in smart city IoT networks for enhanced cybersecurity, Journal of Innovation & Knowledge, Volume 9, Issue 4, 2024, https://doi.org/10.1016/j.jik.2024.100601.

[19] Aluwala, Aakash. (2024). AI-Driven Anomaly Detection in Network Monitoring Techniques and Tools. Journal of Artificial Intelligence & Cloud Computing. 1-6. 10.47363/JAICC/2024(3)310.

[20] Alwhbi, I. A., Zou, C. C., & Alharbi, R. N. (2024). Encrypted Network Traffic Analysis and Classification Utilizing Machine Learning. Sensors (Basel, Switzerland), 24(11), 3509. https://doi.org/10.3390/s24113509

[21] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, Ali A. Ghorbani, Toward developing a systematic approach to generate benchmark datasets for intrusion detection, Computers & Security, Volume 31, Issue 3, May 2012, Pages 357-374, ISSN 0167-4048, 10.1016/j.cose.2011.12.012.

[22] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.