Empirical Evaluation of Autoencoder Models for Anomaly Detection in Packet-based NIDS

Soumyadeep Hore*
University of South Florida
Tampa, USA
soumyadeep@usf.edu

Quoc H. Nguyen*
University of South Florida
Tampa, USA
nguyenq29@usf.edu

Yulun Xu
University of South Florida
Tampa, USA
yulun@usf.edu

Ankit Shah
University of South Florida
Tampa, USA
ankitshah@usf.edu

Nathaniel D. Bastian
United States Military Academy
West Point, USA
nathaniel.bastian@westpoint.edu

Trung Le
University of South Florida
Tampa, USA
tqle@usf.edu

***Abstract*—Anomaly detection is critical for network security. Unsupervised learning models trained on benign network traffic data aim to detect anomalies without relying on attack data sets. Autoencoder-based models have emerged as a promising approach for detecting anomalies in network intrusion data. While autoencoder models have predominantly been utilized in flow-based approaches, which are suitable for offline analysis, there is a notable gap in research concerning unsupervised learning, particularly autoencoder-based techniques, for packet-based network intrusion detection. Packet-based network intrusion detection systems (NIDS) enable real-time detection at a granular level, making this area of investigation crucial.**

This paper compares autoencoder models for anomaly detection in packet-based NIDS. A methodological framework is presented for implementing an autoencoder-based network intrusion detection mechanism with packet data. A novel reconstruction error metric is proposed for autoencoders, which is evaluated at different threshold levels to compare the detection accuracies of network traffic anomalies. The effectiveness of autoencoder models is demonstrated on various network attacks and adversarial samples obtained from publicly available network intrusion data sets. The analysis highlights the strengths and limitations of different autoencoders for network traffic anomaly detection. The insights obtained from the empirical evaluation offer valuable guidance to researchers and practitioners aiming to develop an autoencoder-based network intrusion detection mechanism.

***Index Terms*—Autoencoders, network intrusion detection system, packet-based NIDS, anomaly detection, per-byte reconstruction error metric**

I. Introduction

Network intrusion detection systems (NIDS) play a crucial role by continuously monitoring an organization's network traffic to detect cyber attacks. Machine learning (ML) and deep learning (DL) algorithms power anomaly-based NIDS to timely detect network intrusions. ML/DL-based NIDS rely on extracting features from the network traffic data, which can be achieved using flow- or packet-based approaches [1], [2]. Flow-based approaches extract information from network traffic flow, making them more suitable for offline analysis where a holistic view of network behavior is required. On the other hand, packet-based approaches focus on extracting features from individual network packets, enabling real-time detection and analysis at a more granular level.

Developing NIDS using a supervised learning approach requires a labeled data set of benign and network attack samples to train the ML/DL models, such as tree-based or deep neural network classifiers [3]. However, supervised learning models can only detect attack classes encountered during training. In contrast, unsupervised learning approaches like autoencoder-based models do not rely on an attack data set for training. Instead, they are trained on the organization's benign traffic data and aim to detect anomalies from the normal traffic patterns observed in the network environment without being restricted to specific attack types. While there is a significant body of literature on unsupervised learning approaches, particularly autoencoder-based models, for flow-based NIDS [4]–[7], there is a lack of studies focusing on unsupervised learning for packet-based NIDS.

In this paper, we focus on comparing different types of autoencoder models from the literature for the anomaly detection task in NIDS using packet-based features. We present a methodological framework for implementing an autoencoder-based network intrusion detection mechanism. Through this framework, we provide insights into the strengths and limitations of different autoencoders in detecting various types of network attacks. We perform an empirical evaluation using publicly available network intrusion data. By exploring the potential of an autoencoder-based framework in the context of packet-based NIDS, we aim to assist researchers and practitioners in the cybersecurity community in developing more robust and accurate intrusion detection mechanisms.

The paper makes several notable contributions, which are summarized as follows:

1) Comparative evaluation of autoencoder-based methods: The study presents a methodological framework for comparing different autoencoder-based methods used in network traffic anomaly detection. This analysis serves

* These authors contributed equally to this work.

as a guide for building effective anomaly detectors using autoencoders.

2) Analysis of approach: The study examines an approach for anomaly detection in network intrusion using packet data. This study highlights its strengths and limitations, emphasizing the importance of selecting appropriate autoencoder methods for effective detection.
3) Novel metric for evaluating autoencoder performance: The study proposes a novel metric that overcomes challenges related to varying packet length in network intrusion detection. Unlike traditional autoencoder techniques, this metric computes a per-byte reconstruction error for non-zero elements in the reconstructed data, thereby mitigating issues like zero-padding and evasion attempts by an adversary with crafting of shorter attack packets.
4) Performance evaluation: The study assesses the performance of various autoencoder models under normal and adversarial conditions, including the use of adversarial samples. This evaluation provides insights into the effectiveness and robustness of these models for detecting anomalies in network intrusion scenarios.

The remainder of the paper is structured as follows. Section II provides a detailed description of the various autoencoder methods from the literature. Section III describes the autoencoder-based methodological framework for network traffic anomaly detection that can be utilized by the research community as well as cybersecurity practitioners. The numerical experiment setup is also presented in this section. The results and analysis of the experiments conducted using the framework are presented in Section IV. Finally, Section V provides the conclusions from this study and potential future research directions.

II. Review of Different Autoencoder Models

In this section, we examine various autoencoder architectures discussed in the literature pertaining to the task of network traffic anomaly detection. We select a set of frequently encountered autoencoder models in the NIDS domain [5], [6], [8], [9]. The notations used in explaining the autoencoder-based techniques are summarized in Table I.

TABLE I
Notations Used in Autoencoder-based Techniques

Symbol	Description
$\mathcal{D}$	The data set, $\mathcal{D} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}\}$, contains n data samples; $\lvert\mathcal{D}\rvert = n$.
$\mathbf{x}^{(i)}$	Each data point is a vector of d dimensions, $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)}]$.
$\mathbf{x}$	One data sample from the data set, $\mathbf{x} \in \mathcal{D}$.
$\tilde{\mathbf{x}}$	The corrupted version of $\mathbf{x}$.
$\mathbf{x}'$	The reconstructed version of $\mathbf{x}$.
$\mathbf{z}$	The compressed code learned in the bottleneck layer.
$a_j^{(l)}$	The activation function for j-th neuron in l-th hidden layer.
$g_\phi(.)$	The encoding function parameterized by ϕ.
$f_\theta(.)$	The decoding function parameterized by θ.
$q_\phi(\mathbf{z}\lvert\mathbf{x})$	Estimated posterior probability function (probabilistic encoder).
$p_\theta(\mathbf{x}\lvert\mathbf{z})$	Likelihood of generating true data sample given the latent code (probabilistic decoder).

A. Basic Autoencoder (AE)

The autoencoder is a neural network architecture that learns an identity function by compressing input data into an efficient and compact representation without supervision [10]. It consists of an *encoder* network that performs dimensionality reduction by translating the high-dimensional input into a low-dimensional code, and a *decoder* network that recovers the data from the code. Unlike principal component analysis or matrix factorization, the autoencoder is specifically designed to optimize the reconstruction of the original input from the compressed representation. A high-quality intermediate representation captures latent variables and enhances the decompression process.

The AE model comprises an encoder function $g(.)$ and a decoder function $f(.)$, both parameterized by ϕ and θ, respectively. The *encoder* network compresses the original high-dimensional input $\mathbf{x}$ into a low-dimensional code denoted by $\mathbf{z} = g_\phi(\mathbf{x})$, within the bottleneck layer. The *decoder* network then reconstructs the input by generating $\mathbf{x}' = f_\theta(g_\phi(\mathbf{x}))$. During the training phase, the parameters (θ, ϕ) are jointly learned to produce a reconstructed data sample that closely matches the original input, i.e., $f_\theta(g_\phi(\mathbf{x})) \approx \mathbf{x}$, effectively learning an identity function. Various metrics can be employed to measure the dissimilarity between two vectors, such as cross-entropy in the case of a sigmoid activation function or the simpler mean squared error (MSE) loss, given by:

$$\mathcal{L}_{\text{AE}}(\theta, \phi) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}^{(i)} - f_\theta(g_\phi(\mathbf{x}^{(i)})))^2 \tag{1}$$

B. Denoising Autoencoder (DAE)

The basic autoencoder or AE is designed to learn the identity function, which can lead to overfitting when the number of network parameters exceeds the available data points. To address this issue, the DAE was proposed as a modification to the AE [11]. In this technique, the input is partially corrupted by introducing noise or masking certain values in a stochastic manner, resulting in a partially corrupted input denoted by $\tilde{\mathbf{x}}^{(i)} \sim \mathcal{MD}(\tilde{\mathbf{x}}^{(i)}|\mathbf{x}^{(i)})$, where $\mathcal{MD}$ represents the mapping from true data samples to the noisy or corrupted ones. The type of noise introduced in the DAE is not limited to any specific form, which includes masking noise, Gaussian noise, and salt-and-pepper noise, among others. For instance, in [11], the noise was introduced by randomly selecting a fixed proportion of input dimensions and setting their values to zero. Such an approach allows customization of the corruption process based on prior knowledge or specific requirements. The DAE model is trained to reconstruct the original input, enhancing its robustness and reducing the risk of overfitting. The MSE

loss function for DAE is similar to that of the AE, as follows:

$$\mathcal{L}_{\text{DAE}}(\theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}^{(i)} - f_\theta(g_\phi(\tilde{\mathbf{x}}^{(i)})))^2 \tag{2}$$

In data sets characterized by high dimensionalities and redundancy, such as network intrusion or image data sets, the model tends to rely on evidence from multiple input dimensions to reconstruct the denoised version rather than overfitting to a single dimension. By considering information from various dimensions, the model can capture meaningful patterns and dependencies in the data, leading to a more reliable and effective representation. Such an approach enhances the model's ability to generalize to new, unseen data instances, resulting in a more robust performance in critical tasks such as network intrusion detection.

C. Sparse Autoencoder (SAE)

SAE is particularly effective for data sets that possess sparse features, as it imposes a "sparse" constraint on the activation of hidden units to prevent overfitting and enhance robustness [12]. This constraint ensures that only a small subset of hidden neurons is activated at any given time, with the aim of keeping most hidden neurons inactive for the majority of the time. In the l-th hidden layer, which consists of s_l neurons, the activation function for the j-th neuron is denoted as $a_j^{(l)}(.)$ for $j = 1, \ldots, s_l$. The activation fraction of this neuron, represented as $\hat{\rho}_j$, is expected to be a small value ρ (commonly, set to 0.05), which is referred to as the sparsity parameter. By enforcing sparsity, the SAE encourages the hidden units to capture and represent only the most relevant and discriminative features of the input data, leading to more efficient and meaningful representations. This sparsity-driven approach improves the model's ability to generalize and enhances its robustness, particularly in data sets where sparsity is a prevalent characteristic. The sparsity constraint is enforced by minimizing the following cost function:

$$\mathcal{L}_{\text{SAE}}(\theta) = \mathcal{L}(\theta) + \beta \sum_{l=1}^{L} \sum_{j=1}^{s_l} D_{\text{KL}}(\rho \| \hat{\rho}_j^{(l)}) \tag{3}$$

$$= \mathcal{L}(\theta) + \beta \sum_{l=1}^{L} \sum_{j=1}^{s_l} \rho \log \frac{\rho}{\hat{\rho}_j^{(l)}} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j^{(l)}}, \tag{4}$$

where $D_{\text{KL}}(\rho \parallel \hat{\rho}_j^{(l)})$ is the Kullback-Leibler divergence between the desired sparsity ρ and the average activation $\hat{\rho}_j^{(l)}$ of the j-th neuron in the l-th hidden layer, which is estimated over the training data set as follows:

$$\hat{\rho}_j^{(l)} = \frac{1}{n} \sum_{i=1}^{n} [a_j^{(l)}(\mathbf{x}^{(i)})] \approx \rho \tag{5}$$

D. Variational Autoencoder (VAE)

VAE is a generative neural network model that learns the underlying structure of the input data distribution and enables the generation of new data samples that resemble the original data [13]. The primary objective of VAE is to acquire a compressed representation of the data in a latent space using a probabilistic encoder and a probabilistic decoder. The probabilistic encoder $q_\phi(\mathbf{z}|\mathbf{x})$ takes the input data $\mathbf{x}$ and maps it to a probability distribution over the latent code $\mathbf{z}$. Specifically, the probabilistic encoder outputs the mean μ and the variance σ^2 of a multivariate normal distribution, where the parameters are determined by $\mathbf{x}$.

$$\mu, \sigma^2 = q_\phi(\mathbf{z}|\mathbf{x}) \tag{6}$$

To generate a sample from the latent space, $\mathbf{z}$ is sampled from the normal distribution, $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$. The probabilistic decoder $p_\theta(\mathbf{x}|\mathbf{z})$ then maps the sampled latent code $\mathbf{z}$ back to the data space. The output of the decoder is the reconstructed data $\mathbf{x}'$, as given by:

$$\mathbf{x}' = p_\theta(\mathbf{x}|\mathbf{z}) \tag{7}$$

The loss function for VAE consists of two terms, one for the reconstruction loss and the other one for the KL divergence loss. The reconstruction loss quantifies the dissimilarity between $\mathbf{x}$ and $\mathbf{x}'$. On the other hand, the KL divergence loss captures the disparity between the posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$ and the prior distribution $p(\mathbf{z})$, typically chosen as the standard normal distribution $\mathcal{N}(0, \mathbf{I})$. By minimizing both these losses, the VAE aims to simultaneously enhance the fidelity of the reconstructed data and ensure that the latent space distribution adheres to the desired prior distribution. The loss function for VAE is given by:

$$\mathcal{L}_{\text{VAE}}(\theta, \phi; \mathbf{x}) = \mathbb{E} q_\phi(\mathbf{z}|\mathbf{x})[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})) \tag{8}$$

$$= -\frac{1}{2} \sum_{j=1}^{d} (1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2) + \sum_{i=1}^{k} \log p_\theta(x_i|\mathbf{z}), \tag{9}$$

where k is the dimensionality of the latent code $\mathbf{z}$, μ_j and σ_j are the j-th elements of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ respectively, and $p_\theta(x_i|\mathbf{z})$ is the conditional probability of the i-th dimension of the input data given the latent code. Throughout the training process, the VAE optimizes the parameters (θ, ϕ) by minimizing the aforementioned loss function. The dual objective enables the VAE to learn a meaningful latent representation that facilitates effective data generation and exploration. Once the training is complete, the trained encoder and decoder networks can be applied to perform critical tasks, including network traffic anomaly detection.

E. Ensemble Autoencoder (EAE)

EAE harnesses the strengths of various types of autoencoders and mitigates their individual limitations. The ensemble approach involves training multiple autoencoder-based models using the same data set, and averaging their outputs during the testing phase. To reconstruct test data points, all the autoencoders are trained in the ensemble. Each autoencoder

independently reconstructs the input test data points, utilizing its unique architecture and learned representations. These reconstructions serve as individual outputs from the different autoencoder models within the ensemble. To obtain the final reconstruction from the ensemble autoencoder, an element-wise averaging technique is employed, which entails computing the average value for each element of the reconstructed data points obtained from the various models. By combining the outputs in this manner, a consolidated and refined reconstruction from the ensemble is derived.

III. Methodology and Experiments

A. An Autoencoder-based Framework for Network Traffic Anomaly Detection

The objective of this study is to perform an evaluation of autoencoder-based models to determine their effectiveness in detecting network traffic anomalies by inspecting network packets. We develop a methodological framework using different types of autoencoders to achieve this goal. Figure 1 shows the proposed autoencoder-based framework for network intrusion detection with several processes, including data preparation, model training, novel threshold computation for detecting anomalies, and model evaluation. This framework can be used by cybersecurity researchers as well as practitioners to perform anomaly detection in network traffic using packet data and gain valuable insights. We have made the source code[1] available to the cybersecurity community. The methodological framework begins with the initial step of data preparation, transformation, and labeling. Here, the raw packet capture (pcap) data are processed to extract relevant information. The data set is then divided into two categories: benign and malicious packet samples. To train and evaluate the autoencoders, the extracted benign data is further divided into two sets: benign train samples and benign test samples. The autoencoders are trained using the benign train samples, allowing them to learn the underlying patterns and structure of normal network traffic. Once the autoencoders are trained, the trained models are utilized to process the benign test samples. This step computes the reconstruction errors, which measure the dissimilarity between the original benign samples and their reconstructed versions. By analyzing the reconstruction errors, the framework enables the computation of the benign threshold at various percentile values. These thresholds serve as indicators of the acceptable range of reconstruction errors for benign samples. This information is crucial for distinguishing between normal and anomalous network traffic. To evaluate the performance of the trained autoencoders, a mixture of benign and malicious samples from different attack classes is passed through the models. The reconstruction errors for these samples are calculated, and based on a chosen threshold value, each sample is classified as either benign or attack (malicious). The evaluation metrics used to gauge the performance of each autoencoder-based model are as follows:

[1]https://github.com/quocnh/AE_Framework

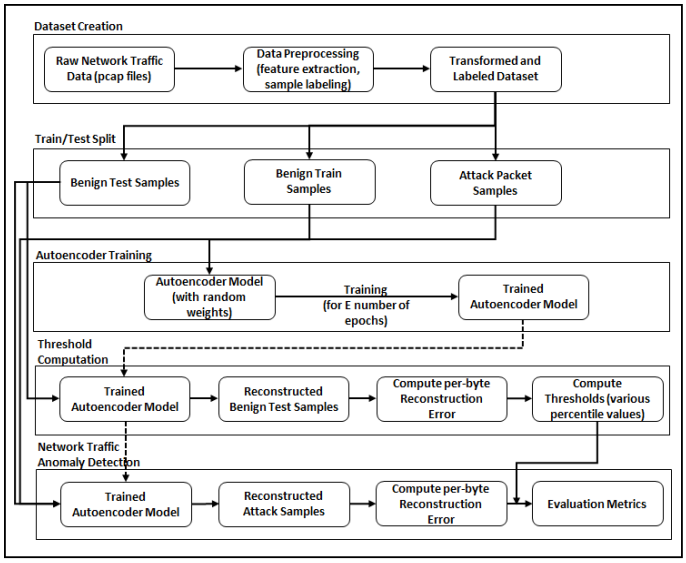

Fig. 1. Network traffic anomaly detection framework using autoencoders

- The true positive rate (TPR), also referred to as recall or detection rate, is defined as the ratio of correctly classified attack samples to the total number of attack samples.
- The true negative rate (TNR) is determined by the ratio of correctly classified benign samples to the total number of benign samples.
- The false negative rate (FNR) is calculated as the ratio of incorrectly classified attack samples to the total number of attack samples.
- The false positive rate (FPR), also known as the false alarm rate, is determined by the ratio of incorrectly classified benign samples to the total number of benign samples.
- Accuracy is defined as the ratio of the total number of correctly classified samples from both classes to the total number of samples.

Next, we describe the numerical experiments performed to evaluate different autoencoders for the network traffic anomaly detection task in NIDS.

B. Numerical Experiments

We first discuss the details related to the training and testing data sets, followed by the training setup and hyperparameter optimization of the various autoencoder models. Later, we introduce a novel metric to determine the threshold for measuring the dissimilarity between benign and attack network traffic packets.

1) Data Description: Anomaly-based NIDS utilizes features extracted from the flows, through the aggregation of information from packet headers, or directly from the network traffic packets. Packet-based NIDS are better suited for real-time detection. In our study, we conducted experiments using packet-level data. Specifically, we selected the CICIDS-2017 data set [14] due to its resemblance to modern network traffic and the availability of raw pcap files [15]. In our

experiments, we also crafted adversarial samples from the CICIDS-2017 attack communications to test the robustness of the autoencoder-based models to evasion attacks. First, we describe the extraction process of benign and attack data from the CICIDS-2017 data set, followed by the details of creating the adversarial samples.

The data extraction process involves obtaining both benign and attack communications from the raw CICIDS-2017 pcap files made publicly available by the Canadian Institute of Cybersecurity (CIC). We followed a similar approach for processing the pcap files and extracting features as in [16]. We extracted byte information from each packet and represented each byte as a feature. To ensure consistency in the number of extracted features, we considered the maximum possible packet length of 1525 and zero-padded packets with fewer features while converting hexadecimal byte values to normalized decimal values. We trained the autoencoder models using the benign data and used the attack data to evaluate the anomaly detection performance of the trained models.

To evaluate the robustness of the trained autoencoders against evasion attacks, we crafted adversarial examples by making subtle modifications to attack packets, specifically targeting ML/DL algorithms to misclassify them as benign. The modifications were constrained to a subset of features that preserved the functionality and maliciousness of the packets and were based on insights from literature studies [17]–[19] and discussions with practitioners. Details of the modifications and their respective byte locations for crafting adversarial attack packets from the original ones are provided in Table II. The generated adversarial packets, categorized by the original attack type, were collected for testing against the trained autoencoders.

2) Training: We trained the autoencoders for the anomaly detection task using 70% of the extracted benign data from the CICIDS-2017 pcap files. The training lasted for 1000 epochs. To establish a fair point of comparison, we used a similar structure and hyperparameter setting for each autoencoder. We used a symmetric architecture for the encoder and decoder networks. We experimented with various hyperparameter values. Finally, our encoder network consists of three hidden layers with a gradually decreasing number of perceptrons $(150, 100, 50)$. The activation function used in the hidden layers and the output layer is the rectified linear unit (ReLU). The chosen latent dimension is 20. The decoder network is a mirror image of the encoder network with the number of perceptrons in the hidden layers gradually increasing $(50, 100, 150)$. We used a batch size of 256 with 'adam' optimizer and 0.001 learning rate.

3) Testing and Evaluation: The autoencoder performance is typically evaluated using the reconstruction error, which measures the dissimilarity between the original sample and its reconstructed counterpart. For an original sample x and a reconstructed sample x', the reconstruction error can be represented as $f(x, x')$, where $f(.)$ signifies any function that measures the difference between x and x'. This function value can be computed using the Euclidean distance, the Manhattan distance, or the Chebyshev distance, among others. In this work, the Manhattan distance or L-1 norm was chosen as the difference function. However, the L-1 norm's reliance on the number of non-zero elements in the packet representation posed challenges due to zero-padding and potential evasion by shorter attack packets. To address this, a per-byte reconstruction error approach was adopted, focusing only on non-zero elements in the reconstructed data, thereby mitigating the packet length bias. We first collect a new set of indices K for all $x'_{i,j} \neq 0$. We then calculate their absolute byte-wise differences. Finally, we take an average of the aggregated error by dividing it with the cardinality of the set, $|K|$. This new metric score for per-byte reconstruction error (*pbRE*) is computed from the non-zero bytes, as follows:

$$pbRE = \frac{\sum_{j \in K} abs(x_{i,j} - x'_{i,j})}{|K|}, \tag{10}$$

where $x_{i,j}$ represents the value of the j-th feature from the i-th original sample and $x'_{i,j}$ represents that of the i-th reconstructed sample.

The evaluation of the trained autoencoders involves two main tasks: threshold determination and anomaly detection. To determine the threshold, the remaining 30% of benign test data samples are passed through the autoencoders to compute the pbRE scores. Three thresholds are constructed based on the allowable FPRs at 1%, 5%, and 10%, respectively. These thresholds are used to classify unseen test samples as anomalies if their pbRE score exceeds the threshold. The performance of the autoencoders is evaluated by testing them against *Port Scan*, *DoS*, *DDoS*, *Infiltration*, *Web Attack*, and *Brute Force* attack packets. Additionally, the autoencoders are subjected to adversarial samples to assess their robustness against evasion attacks.

IV. Results and Analysis

In this section, we discuss the results of the conducted experiments and their analysis. First, we provide a comparison of the pbRE scores computed for the various autoencoders described in Section II for the benign and different attack network packets. Then, we analyze the attack detection performance of these autoencoder models. Finally, we evaluate the robustness of the trained autoencoder models in detecting various types of adversarial samples, crafted for evading the DL architecture in the NIDS.

Table III presents statistics of the pbRE scores for the test benign samples. The basic autoencoder (AE) exhibits the lowest average score, while the VAE shows similar reconstruction performance. On the other hand, the SAE achieves the highest pbRE score. We considered different pbRE percentile values to establish an anomaly detection threshold, which is explained in detail in the next two paragraphs. The similarity between the mean and 90th percentile scores suggests the presence of outliers in the data set. Specifically, we can infer that only 10% of the test samples reconstructed using a basic autoencoder have a higher reconstruction error than the mean.

TABLE II
SUMMARY OF THE MODIFICATIONS USED FOR CRAFTING ADVERSARIAL PACKETS

Modification	Location	Description	Affected Bytes
Changing time-to-live (TTL) values	IP Header	Increasing or decreasing the TTL value of the packets by +/- δ	TTL byte (IP header byte 9), IP checksum bytes (IP header bytes 11 and 12)
Changing window size	TCP Header	Increasing or decreasing the window size value of the packets by +/- δ	Window size bytes (TCP header bytes 15 and 16), TCP checksum (TCP header bytes 17 and 18)
Fragmentation	IP Header	Changing 'do not fragment' to 'do fragment'	Fragmentation bytes (IP header bytes 7 and 8), TTL byte (IP header byte 9), IP checksum bytes (IP header bytes 11 and 12)
Payload addition	TCP Segment	Injecting dead payload bytes at the end of the original packet data	TCP segment bytes, TCP checksum bytes (TCP header bytes 17 and 18)
TCP Options addition	TCP Header	Adding or changing elements of TCP Options such as maximum segment size (MSS), No Operation bytes (NOP), window scale	TCP option bytes, TCP checksum bytes (TCP header bytes 17 and 18)

TABLE III
PBRE SCORES OF BENIGN SAMPLES FOR THE DIFFERENT AUTOENCODERS

Type	pbRE Score			
	Mean	99 percentile	95 percentile	90 percentile
AE	0.005	0.086	0.011	0.005
DAE	0.010	0.141	0.036	0.012
SAE	0.032	0.125	0.059	0.052
VAE	0.007	0.093	0.024	0.007
EAE	0.014	0.107	0.028	0.018

TABLE IV
MEAN PBRE SCORES OF VARIOUS ATTACK TYPES FOR DIFFERENT AUTOENCODERS

Type	Mean pbRE Score					
	Portscan	DoS	DDoS	Infiltration	Web Attack	Brute Force
AE	0.165	0.146	0.059	0.137	0.151	0.195
DAE	0.141	0.140	0.108	0.139	0.142	0.162
SAE	0.187	0.181	0.142	0.140	0.188	0.228
VAE	0.124	0.157	0.089	0.155	0.170	0.179
EAE	0.154	0.156	0.099	0.142	0.162	0.191

Table IV displays the pbRE scores of different attacks when reconstructed by the various autoencoders. It can be noted that the SAE exhibits the highest score, indicating a larger reconstruction error, across all attack types. The calculation of benign and attack reconstruction errors is a crucial step in anomaly detection. Hence, to assess the anomaly detection performance, we quantify the autoencoders' performance at different benign thresholds. For instance, setting the threshold at the 99th percentile of benign pbRE scores for a basic autoencoder (as shown in Table III), any sample with a score exceeding 0.086 is considered an anomaly. By selecting the 99th percentile threshold, we allow for a 1% FPR. Next, we present the performance comparison of the trained autoencoders in detecting various attacks.

Table V presents the performance comparison of the autoencoders at three different levels of allowable FPR. At a 1% FPR, the VAE demonstrates the highest success in detecting attacks across all attack types. The AE, SAE, and EAE exhibit comparable performance in terms of average TPR, achieving 83.66%, 82.66%, and 88.33%, respectively, while the DAE performs the poorest with an average TPR of 45%. Among the attack types, *Brute Force* was the easiest to detect across all autoencoders at an average TPR of 91%, followed by *Port Scan, Web Attack, DoS,* and *Infiltration* with average TPRs of 88.2%, 87.6%, 84.5%, and 81.4%, respectively. Notably, all autoencoders struggle to differentiate *DDoS* attack packets from benign traffic at a 1% FPR, achieving an average TPR of 35.88% across all models. The SAE performs the best in detecting *DDoS* packets with a TPR of 64%. However, at a 5% FPR, the *DDoS* detection significantly improves with an average TPR of 99% across all autoencoders.

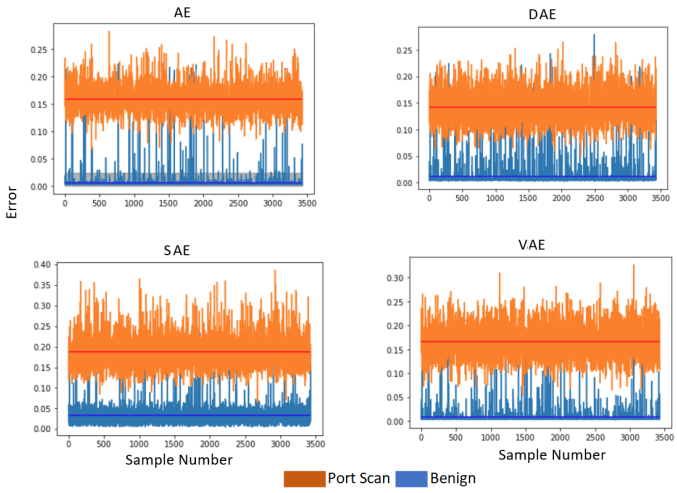

Fig. 2. pbRE score comparison among different autoencoders for Benign vs Port Scan attack packets

Figures 2 and 3 illustrate the difference in pbRE scores between 3500 randomly selected benign packets and *Port Scan* packets, as well as between 3500 randomly selected benign packets and *DDoS* attack packets, respectively. The x-axis represents the number of benign and attack samples, while the y-axis represents the pbRE score for each packet. The red and blue lines in the figures indicate the mean error for attack and benign packets, respectively. We selected the *Port Scan* and *DDoS* attacks as they represent the easiest and hardest to detect, respectively, according to Table V. In Figure 2, a noticeable gap is observed between the benign samples reconstruction error and that of *Port Scan* attack packets.

TABLE V
PERFORMANCE COMPARISON OF DIFFERENT AUTOENCODERS AT VARIOUS FPR VALUES

Type	Attack Type	1% FPR				5% FPR				10% FPR			
		TPR (%)	TNR (%)	FNR (%)	Accuracy (%)	TPR (%)	TNR (%)	FNR (%)	Accuracy (%)	TPR (%)	TNR (%)	FNR (%)	Accuracy (%)
AE	Port Scan	99	99	1	98	100	95	0	95	100	90	0	90
	DoS	95.2	99	4.8	94.2	100	95	0	95	100	90	0	90
	DDoS	14.1	99	85.9	13.1	100	95	0	95	100	90	0	90
	Infiltration	98.7	99	1.3	97.7	100	95	0	95	100	90	0	90
	Web Attack	98	99	2	97	100	95	0	95	100	90	0	90
	Brute Force	97	99	3	96	100	95	0	95	100	90	0	90
DAE	Port Scan	47	99	53	46	100	95	0	95	100	90	0	90
	DoS	48	99	52	47	100	95	0	95	100	90	0	90
	DDoS	14	99	86	13	99	95	1	94	100	90	0	90
	Infiltration	47	99	53	46	100	95	0	95	100	90	0	90
	Web Attack	53	99	47	52	100	95	0	95	100	90	0	90
	Brute Force	61	99	39	60	100	95	0	95	100	90	0	90
SAE	Port Scan	97	99	3	96	100	95	0	95	100	90	0	90
	DoS	87	99	13	86	100	95	0	95	100	90	0	90
	DDoS	64	99	36	63	97	95	3	92	98	90	2	88
	Infiltration	64	99	36	63	100	95	0	95	100	90	0	90
	Web Attack	89	99	11	88	100	95	0	95	100	90	0	90
	Brute Force	95	99	5	94	100	95	0	95	100	90	0	90
VAE	Port Scan	99	99	1	98	100	95	0	95	100	90	0	90
	DoS	96	99	4	95	100	95	0	95	100	90	0	90
	DDoS	49	99	51	48	99	95	1	94	100	90	0	90
	Infiltration	98	99	2	97	100	95	0	95	100	90	0	90
	Web Attack	99	99	1	98	100	95	0	95	100	90	0	90
	Brute Force	99	99	1	98	100	95	0	95	100	90	0	90
EAE	Port Scan	99	99	1	98	100	95	0	95	100	90	0	90
	DoS	96	99	4	95	100	95	0	95	100	90	0	90
	DDoS	38	99	62	37	100	95	0	95	100	90	0	90
	Infiltration	99	99	1	98	100	95	0	95	100	90	0	90
	Web Attack	99	99	1	98	100	95	0	95	100	90	0	90
	Brute Force	99	99	1	98	100	95	0	95	100	90	0	90

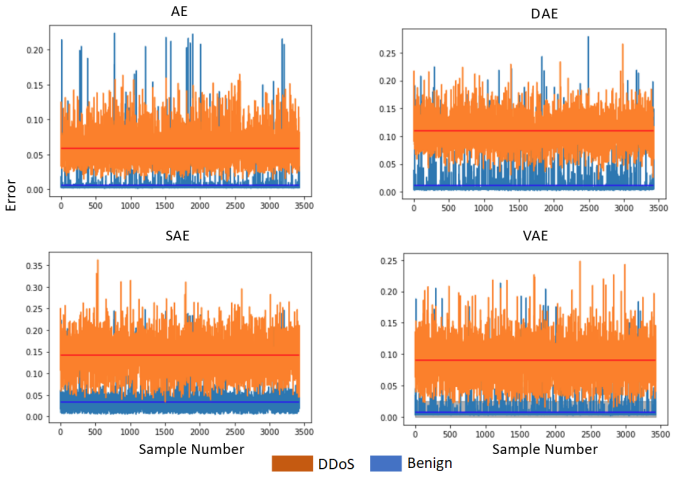

Fig. 3. pbRE score comparison among different autoencoders for Benign vs DDoS attack packets

However, the difference between the reconstruction error of benign and *DDoS* packets is less distinct in Figure 3.

Finally, we evaluate the performance of the autoencoders when subjected to carefully crafted adversarial attack samples. We compare their performance with popular ML/DL-based NIDS, including decision tree, random forest, support vector machine, and deep neural networks [20], [21]. Our observations reveal evasion rates ranging from 40% to 99% against these ML/DL-based NIDS for different attack types. In contrast, the autoencoders, except for the DAE, effectively minimize the evasion rate between 0% to 6%. Table VI presents the TPRs for different autoencoders against adversarial examples generated from *Port Scan, DoS, DDoS,* and *Infiltration* attack packets. Note that the ensemble model (EAE) also achieves a similar performance due to the strong detection accuracy of all individual autoencoders. The AE (basic autoencoder) demonstrates the highest success rate, with an average TPR exceeding 99% across these attack types at 1% FPR. The VAE and SAE also perform well, achieving average TPR values of 98% and 96% respectively, at 1% FPR. At 5% and 10% FPR, all autoencoders successfully detect 100% of the adversarial packets, highlighting their superior robustness.

V. CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

We compared different autoencoder models for anomaly detection in NIDS using packet-based features. We presented a methodological framework for implementing an autoencoder-based network intrusion detection mechanism and provided insights into the anomaly detection performance of different autoencoders across various network attacks using publicly available network intrusion data. Below, we summarize the insights obtained from this study:

1) Reconstruction performance on benign samples: The basic autoencoder (AE) stands out for its superior per-

TABLE VI
TPR FOR DIFFERENT AUTOENCODERS AGAINST ADVERSARIAL PACKETS

Type	AE			DAE		
FPR (%)	1	5	10	1	5	10
Port Scan	100	100	100	74	100	100
DoS	99	100	100	71	100	100
DDoS	99	100	100	31	100	100
Infiltration	98	100	100	45	100	100
Type	SAE			VAE		
FPR (%)	1	5	10	1	5	10
Port Scan	99	100	100	99	100	100
DoS	95	100	100	97	100	100
DDoS	96	100	100	99	100	100
Infiltration	94	100	100	97	100	100

formance in reconstructing benign samples compared to the other autoencoders.

2) Detection performance on different attacks: At a 1% FPR, the VAE shows the highest success in detecting attacks across all attack types. On the other hand, the DAE exhibits the lowest performance with an average TPR of 45%. Our results identify the *DDoS* attack as the most challenging to detect across all of the evaluated autoencoders. The SAE performs the best among them in detecting *DDoS* attack packets with a TPR of 64%.
3) Performance at different FPR levels: At higher FPR levels of 5% and 10%, all autoencoders successfully detect 100% of nearly all attack types, including both original and perturbed (adversarial) packets. However, such thresholds come with a trade-off of increased workload for security analysts, as they require investigations of a larger number of false positives.
4) Performance against adversarial samples: The AE appears to be the best autoencoder type for handling carefully crafted adversarial attack samples, as it demonstrates the highest success rate in detecting these attacks. The VAE also exhibits strong performance and can be considered as a viable alternative.

As future work, we aim to evaluate the robustness of the autoencoder-based network traffic anomaly detection framework with the novel per-byte reconstruction error metric in various network environments and against AI/ML-based adversarial agents. This evaluation would provide valuable insights into the framework's effectiveness and resilience in real-world scenarios.

ACKNOWLEDGMENTS

This work was supported in part by the U.S. Military Academy (USMA) under Cooperative Agreement No. W911NF-22-2-0045, as well as the U.S. Army Combat Capabilities Development Command C5ISR Center under Support Agreement No. USMA21056. The views and conclusions expressed in this paper are those of the authors and do not reflect the official policy or position of the U.S. Military Academy, U.S. Army, U.S. Department of Defense, or U.S. Government.

REFERENCES

[1] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on sdn based network intrusion detection system using machine learning approaches," *Peer-to-Peer Networking and Applications*, vol. 12, pp. 493–501, 2019.
[2] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, p. e4150, 2021.
[3] X. Li, W. Chen, Q. Zhang, and L. Wu, "Building auto-encoder intrusion detection system based on random forest feature selection," *Computers & Security*, vol. 95, p. 101851, 2020.
[4] G. Muhammad, M. S. Hossain, and S. Garg, "Stacked autoencoder-based intrusion detection system to combat financial fraudulent," *IEEE Internet of Things Journal*, 2020.
[5] K. N. Rao, K. V. Rao, and P. R. PVGD, "A hybrid intrusion detection system based on sparse autoencoder and deep neural network," *Computer Communications*, vol. 180, pp. 77–88, 2021.
[6] H. Choi, M. Kim, G. Lee, and W. Kim, "Unsupervised learning approach for network intrusion detection system using autoencoders," *The Journal of Supercomputing*, vol. 75, pp. 5597–5621, 2019.
[7] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *arXiv preprint arXiv:1802.09089*, 2018.
[8] Y.-D. Lin, Z.-Q. Liu, R.-H. Hwang, V.-L. Nguyen, P.-C. Lin, and Y.-C. Lai, "Machine learning with variational autoencoder for imbalanced datasets in intrusion detection," *IEEE Access*, vol. 10, pp. 15247–15260, 2022.
[9] S. A. H. Ayubkhan, W.-S. Yap, E. Morris, and M. B. K. Rawthar, "A practical intrusion detection system based on denoising autoencoder and lightgbm classifier with improved detection performance," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–26, 2022.
[10] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[11] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, 2008.
[12] A. Ng *et al.*, "Sparse autoencoder,"
[13] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
[14] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "A detailed analysis of the cicids2017 data set," in *Information Systems Security and Privacy: 4th International Conference, ICISSP 2018, Funchal-Madeira, Portugal, January 22-24, 2018, Revised Selected Papers 4*, pp. 172–188, Springer, 2019.
[15] H. Hindy, D. Brosset, E. Bayne, A. K. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy of network threats and the effect of current datasets on intrusion detection systems," *IEEE Access*, vol. 8, pp. 104650–104675, 2020.
[16] S. Hore, J. Ghadermazi, D. Paudel, A. Shah, T. K. Das, and N. D. Bastian, "Deep packgen: A deep reinforcement learning framework for adversarial network packet generation," *arXiv preprint arXiv:2305.11039*, 2023.
[17] M. Nasr, A. Bahramali, and A. Houmansadr, "Defeating dnn-based traffic analysis systems in real-time with blind adversarial perturbations.," in *USENIX Security Symposium*, pp. 2705–2722, 2021.
[18] A. M. Sadeghzadeh, S. Shiravi, and R. Jalili, "Adversarial network traffic: Towards evaluating the robustness of deep-learning-based network traffic classification," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1962–1976, 2021.
[19] B. Yan and G. Han, "Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system," *IEEE Access*, vol. 6, pp. 41238–41248, 2018.
[20] B.-E. Zolbayar, R. Sheatsley, P. McDaniel, M. J. Weisman, S. Zhu, S. Zhu, and S. Krishnamurthy, "Generating practical adversarial network traffic flows using nidsgan," *arXiv preprint arXiv:2203.06694*, 2022.
[21] H. Yan, X. Li, W. Zhang, R. Wang, H. Li, X. Zhao, F. Li, and X. Lin, "Automatic evasion of machine learning-based network intrusion detection systems," *IEEE Transactions on Dependable and Secure Computing*, 2023.