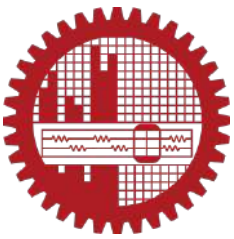
M.Sc. Engg. Thesis

CONDITIONAL VARIATIONAL LAPLACE AUTOENCODER BASED NETWORK INTRUSION DETECTION SYSTEM

by

Shuhana Azmin (0416052062)

Submitted to
Department of Computer Science and Engineering
(In partial fulfillment of the requirements for the degree of
Master of Science in Computer Science and Engineering)



Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka 1000

September, 2022

Dedicated to my loving parents

Author's Contact

Shuhana Azmin

Email: shuhana.azmin@gmail.com

The thesis titled CONDITIONAL VARIATIONAL LAPLACE AUTOENCODER BASED NETWORK INTRUSION DETECTION SYSTEM, submitted by Shuhana Azmin, Roll No. **0416052062**, Session April 2016, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents. Examination held on September 19, 2022.

Board of Examiners

1. _____
Dr. A. B. M. Alim Al Islam — Chairman (Supervisor)
Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka.

2. _____
Dr. Mahmuda Naznin — Member (Ex-Officio)
Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka.

3. _____
Dr. Abu Sayed Md. Latiful Hoque — Member
Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka.

3. _____
Dr. Rezwana Reaz — Member
Assistant Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka.

4. _____
Dr. M. Kaykobad — (External)
Distinguished Professor
Department of Computer Science and Engineering
BRAC University, Dhaka.

Candidate's Declaration

This is hereby declared that the work titled CONDITIONAL VARIATIONAL LAPLACE AUTOENCODER BASED NETWORK INTRUSION DETECTION SYSTEM is the outcome of research carried out by me under the supervision of Dr. A. B. M. Alim Al Islam, in the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka 1000. It is also declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

Shuhana Azmin

Shuhana Azmin

Candidate

Acknowledgment

First, I would like to express my paramount gratitude to my supervisor, Dr. A. B. M. Alim Al Islam, for allowing me to be a part of this research project. During this time, his guidance helps me go beyond my level and bring the best in my work. His limitless patience gives me the courage to think out of the box and make this thesis successful.

Besides, I would like to thank the honourable members of my thesis committee: Prof. Dr. Mahmuda Naznin, Prof. Dr. Abu Sayed Md. Latiful Hoque, Asst. Prof. Dr. Rezwana Reaz, and Prof. Dr. M. Kaykobad, for their encouragement, comments, and valuable feedback.

I am also thankful to Sajeda Akter (M.Sc. student, CSE, BUET), Kazi Sharmin Dina (M.Sc. student, CSE, BUET), and Tarik Reza Toha (M.Sc. student, CSE, BUET). I sought help from them several times in different phases of the study. I am also grateful to all honourable teachers of the department for their comments and suggestions.

Most importantly, I am grateful for the constant support from my beloved parents, who always remains to inspire me in every aspect of my life.

Abstract

Network Intrusion Detection System (NIDS) is an essential tool for network administrators to detect security breaches. Currently, the existing anomaly-based intrusion detection methods rely on traditional machine learning models such as Support Vector Machine and Random Forest. However, current machine learning-based NIDS applications often do not perform well due to the diversity of attacks and imbalanced datasets having less data pertinent to attack events. Therefore, it is important to synthesize data in a probabilistic manner that is similar to original attack event-related data. Accordingly, in this paper, we propose a new paradigm of the synthesizing task based on Variational Laplace AutoEncoder (VLAE) and Deep Neural Network. We exploit the paradigm to develop a new intrusion detection method. Here, we go beyond the existing VLAE model through incorporating class labels as an input in the VLAE model. Hence, the latent representation of samples of different class labels separate in the latent space and we can generate attack samples based on the class labels. We term the enhanced model as Conditional Variational Laplace Autoencoder (CVLAE). We further extend our proposed model by adding attention mechanism to better reconstruct features, named Conditional Variational Laplace Attention AutoEncoder (CVLAAE). We employ CVLAE and CVLAAE to learn latent variable representations of network data features and to synthesize data in a probabilistic manner. To do so, we use a Deep Neural Network (DNN) classifier, which is trained on the original and synthesized data. The DNN classifier is used to classify the attack samples. We evaluate our model on different benchmark datasets namely NSL-KDD and KDD CUPP 99 datasets. Here, we demonstrate the efficacy of our proposed method through showing that our method achieves higher performance in the cases of minority attacks compared to other existing methods in our experimentation. The experimental results further demonstrate that adding the attention mechanism in CVLAE resulting in CVLAAE has the best overall performance in terms of precision, recall, specificity, and F1 score.

Contents

List of Figures

List of Tables

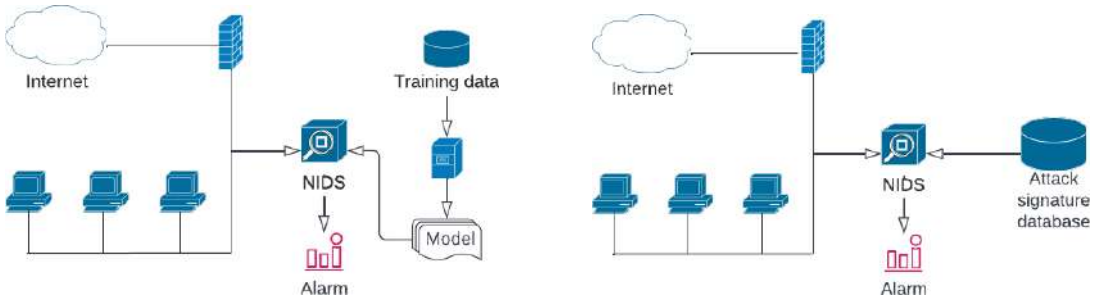
Chapter 1

Introduction

With the rapid advancement in communications and information technology, the Internet has reached all aspects of our life. More and more people are now using online services and applications, which results in a high volume of data transferred over the network everyday. However, with the advancement of technology and network systems, network attacks are growing rapidly and becoming more sophisticated [3]. Besides the Internet, enterprise networks are also facing the attacks because of their complex network environment [4]. Therefore, ensuring network security has become a huge challenge for both public as well as enterprise networks.

For example, in July 2019, a hacker gained access to 100 million Capital One credit card applications and accounts [5]. In the first half of 2019, Kaspersky detected more than 100 million attacks on smart devices [6]. In March 2020, hackers compromised a commercial software application made by SolarWinds and gained access to the systems running the SolarWinds products [7]. In May 2021, a significant DDoS attack disabled the ISP used by Belgium's government, which cut off Internet access to more than 200 organizations, including Belgium's Parliament [8]. These incidents suggest that hackers can now exploit different systems' security holes to gain access to confidential data and financial accounts.

As network protocols and services have increased rapidly over the last decades, the risk of exploiting them is also growing. Security attacks through such exploitation bring substantial financial loss, reputational damage, and legal consequences to organizations. According to UK Cyber Security Breaches Survey [9], in 2021, among the 39% of businesses that identified attacks, 21% of them ended up losing money, data, or other assets. 35% of them experienced a negative impact, such



(a) Anomaly-based NIDS (b) Signature-based NIDS

Figure 1.1: Categories of NIDS based on its detection strategies

as loss of customers, business disruption, etc., as a result of the attacks. Therefore, protecting the network from security intrusions has become an essential and challenging task for security experts. A Network Intrusion Detection System or NIDS is one of several security mechanisms to manage security intrusions. An NIDS detects malicious activities by analyzing incoming and outgoing network traffic. Based on the detection strategies adopted by NIDSs, there can be two primary categories of detection: i) signature-based detection and ii) anomaly-based detection [10]. A signature-based NIDS (Figure 1.1b) compares traffic data with a predefined set of attacks. It can only detect known attacks. On the other hand, an anomaly-based NIDS (Figure 1.1a) creates a normal model of the behaviour of a network using machine learning, statistical-based or knowledge-based methods. Any significant deviation between the observed behaviour and the modeled behaviour is regarded as a potential intrusion. Although it can identify new and existing attacks, there are some challenges in applying it in the real network environments.

Limitations of Existing Studies And Approaches

Most of the existing security solutions use signature-based systems because of their efficiency in detecting known attacks. However, signature-based NIDSs are unable to recognize zero-day or new attacks [10]. In contrast, anomaly-based detection systems compare network traffic with normal traffic model and marks a significant deviation from the normal traffic behaviour as an anomaly. As a result, these systems can detect zero-day attacks. However, these are not commercially used due to low detection accuracy and high false alarm rates [10].

Machine learning-based approaches are generally utilized in anomaly-based NIDSs to improve detection accuracy and to achieve a low false alarm rate. These approaches include Decision Tree [11], K-nearest Neighbor [12], Naive Bayes Network [13], Self-organized Map [14], Support Vector Machine [15], and Artificial Neural Network [16]. However, these techniques demand extensive feature engineering because of having high-dimensional[1] network data. Here, another challenge is that more number of samples is required for the predictive modeling. It is usually called the curse of dimensionality.

Recently, deep learning-based approaches are being applied in implementing anomaly-based NIDSs. Deep learning-based systems have the capability to learn feature correlation automatically. Deep learning-based methods used for intrusion detection includes Deep Belief Network (DBN) [17, 18], Deep Neural Network (DNN) [19], AutoEncoder [20], and Recurrent Neural Network (RNN) [21]. However, there are some problems in applying deep learning-based methods in intrusion detection. Firstly, network data is not balanced. Attack samples are very low in comparison to normal samples. This causes biasedness in the classification. Secondly, most of the network data is unlabeled. To achieve good performance deep learning algorithms require a significant amount of labeled data.

Motivation behind This Study

To overcome the challenges in applying deep learning-based methods in intrusion detection, synthesizing intrusion data that resemble original data is very important. State-of-the-art algorithms in synthesizing data are based on SMOTE [22] and ADASYN [23]. Recently, Variational AutoEncoder (VAE) based approaches have been applied for intrusion data generation [24, 25]. Variational AutoEncoder is a probabilistic generative model that combines variational inference with deep learning [26]. However, VAE approximates posterior distribution by fully factorized Gaussian that has limited expressiveness [27].

To overcome the limitations, in this study, we present two probabilistic data generative models called Conditional Variational Laplace AutoEncoder (CVLAE) and Conditional Variational Laplace Attention AutoEncoder (CVLAAE), which are based on Variational Laplace AutoEncoder [27] used to generate intrusion data. Variational Laplace AutoEncoder approximates posterior distribution by fully-factorized Gaussian, whose covariance is determined by the local behaviour of the generative network. We modify the conventional VLAE architecture to include class labels along with features to

[1]High-dimensional data usually makes a predictive modeling task harder. This happens as the computational complexity increases exponentially with the increase in the number of features.

generate latent variable representation. Besides, we introduce attention mechanism [28] in CVLAAE to learn the latent features more effectively. Here, we use Deep Neural Network as a classifier of network intrusion trained on the combination of original and synthesized data.

Our Contributions

This study contributes to the field of anomaly-based network intrusion detection systems. Here, we primarily focus on improving the capability of intrusion detection systems in identifying new and zero-day attacks. However, as the problem in this regard lies in the availability of enough data pertinent to the new and zero-day attacks, our study aims to efficiently generate the data. To do so, we leverage deep learning-based data generative models to overcome the challenges as already mentioned above.

Based on our work, we make the following set of contributions in this study:

- We propose two new probabilistic data generative models named CVLAE and CVLAAE for intrusion data generation. CVLAE incorporates class labels in VLAE [27] model to learn the latent distribution of complex network traffic. Then, we extend CVLAE by adding attention mechanism [28] to learn feature representation more effectively. Using them, we generate attack samples according to class labels and balance the network traffic data.
- We explore a DNN to train on our generated and original data to classify the attack samples. To do so, We adopt two benchmark datasets namely the NSL-KDD[1] and KDD CUP 99 [2] datasets.
- We compare our proposed method against three state-of-the-art methods namely DNN, SAAE-DNN, and CVAE-DNN based on the two benchmark datasets. We achieve better performance in terms of precision, recall, specificity, and F1 score in comparison to the other three existing methods.

Organization of This Study

We organize the rest of this study in the following way. In Chapter 2, we will discuss background of our work. We will present research studies related to this work in Chapter 3. After that, we will discuss our proposed method in Chapter 4. Next, we will show the experimental results and compare

performance of our proposed method with other existing methods in Chapter 5. We will discuss limitations and applications of this study in Chapter 6. Finally, we will conclude the study, including some future work in Chapter 8.

Chapter 2

Background of Our Study

As this study aims at developing an anomaly-based NIDS that can detect both known and unknown network attacks, different notions such as NIDS, AE, VAE, VLAE, and attention mechanism are highly related to this study. Therefore, in this section, we present brief backgrounds on NIDS, AE, VAE, VLAE, and attention mechanism.

Network Intrusion Detection System (NIDS)

Intrusion Detection System or IDS monitors and analyzes the events in a computer system or network to detect possible attacks. IDS is an important security system to protect network resources against security threats, which are increasing significantly in their number and impacts. IDS can be used to monitor host and network-based environments. A host-based IDS (HIDS) monitors the events of hosts to detect suspicious activities [29]. A network-based IDS or NIDS monitors network traffic to identify remote attacks over a network connection [29]. An NIDS realizes an essential security mechanism, as it provides a solid line of defence against malicious activities in a network. In this study, we focus on the network-based intrusion detection system.

Figure 2.1 shows typical architecture of an NIDS. An NIDS is typically placed behind the firewall on the edge of a network. It can also be placed in various locations for different purposes. An NIDS between the Internet and firewall is useful for learning about malicious activities on the Internet. An NIDS in the DMZ will see attacks originating from the Internet that can get through the outer firewall to public servers. It sniffs the internal interface of the firewall and sends alerts to an NIDS management server. It analyzes all the traffic to detect malicious traffic. There are three major

categories of NIDSs namely Signature-based IDS(SIDS), Anomaly-based IDS(AIDS), and Stateful Protocol Analysis (SPA) [29].

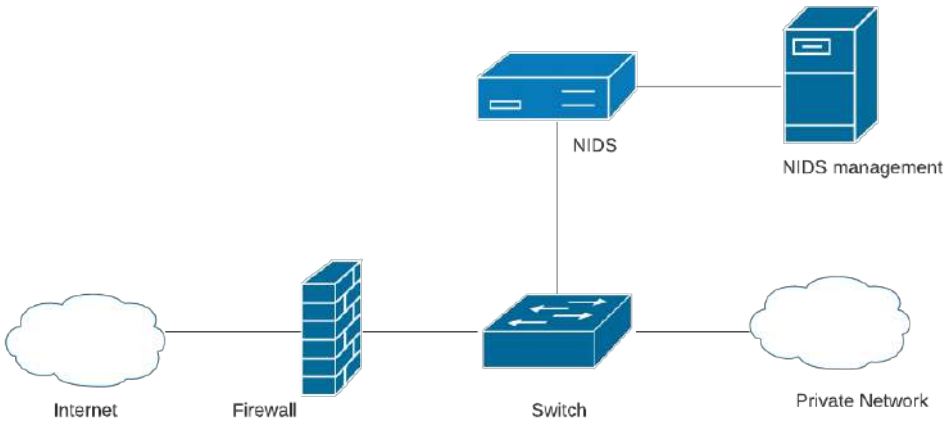


Figure 2.1: Architecture of an NIDS

- Signature-based IDS (SIDS): A signature is a pattern that represents a known attack or threat. An SIDS is based on pattern matching techniques to find a known attack. It compares captured events with signatures of known attacks to detect possible intrusions. Because of using the knowledge accumulated based on specific attacks and system vulnerabilities, SIDS is also known as knowledge-based detection system or misuse detection system.

- Anomaly-based IDS (AIDS): An AIDS creates a standard model of the behaviour of a network using machine learning, statistical-based, or knowledge-based methods. Any significant deviation between the model and observed behaviour is regarded as an intrusion in an AIDS. It can identify known and unknown attacks and require less effort to construct its profile of the normal behaviour of the network than an SIDS.

- Stateful Protocol Analysis (SPA): An SPA examines and traces protocol states, e.g., pairing requests with replies. Although an SPA is mostly similar to an AIDS, it relies on vendor-developed profiles of certain protocols. It requires information of the relevant network's protocol standard from international standard organizations, e.g., IETF [29]. An SPA focuses on known attacks or threats. SPA is also known as a specification-based detection system.

This thesis focuses on probabilistic generative models and how to use them to design an anomaly-

based IDS. We specifically focus on AIDS considering its generalizability compared to other alternatives. Here, we attempt to leverage the capability of an AIDS through exploiting the notion of variational autoencoder.

AutoEncoder (AE)

AutoEncoder (AE) refers to a neural network that learns to reconstruct the original input while compressing the data to create a more efficient and compressed representation. AE consists of an encoder and a decoder, as shown in Figure 2.2. The encoder compresses the data to a lower-dimensional latent representation, and the decoder decodes the latent representation to a very close representation to the original data. The decoder ensures that the latent space can capture most of the information from the dataset space by forcing the latent space to mostly output what was fed as input to the encoder.

To exemplify the tasks of encoder and decoder, let's say x is a 28×28-pixel photo of a handwritten digit. The encoder encodes the 784-dimensional (28×28) data into a latent (hidden) representation space z, which is much less than 784 dimensions. The decoder takes the latent representation z as input and outputs the reconstructed input data. Thus, the output of the decoder will be another 784-dimensional (28×28) photo of a handwritten digit.

AE is trained to minimize the reconstruction error between the input data x and the reconstructed data x', and thus, ensures that the latent space can capture important information from the input data. Here, the reconstruction error can be the mean squared error between the encoder input and the decoder output [30], which can be modeled as follows.

$$\mathrm{L}(x, x') = ||x - x'||^2 = ||x - d_\vartheta(z)||^2 = ||x - d_\vartheta(e_\phi(x))||^2 \tag{2.1}$$

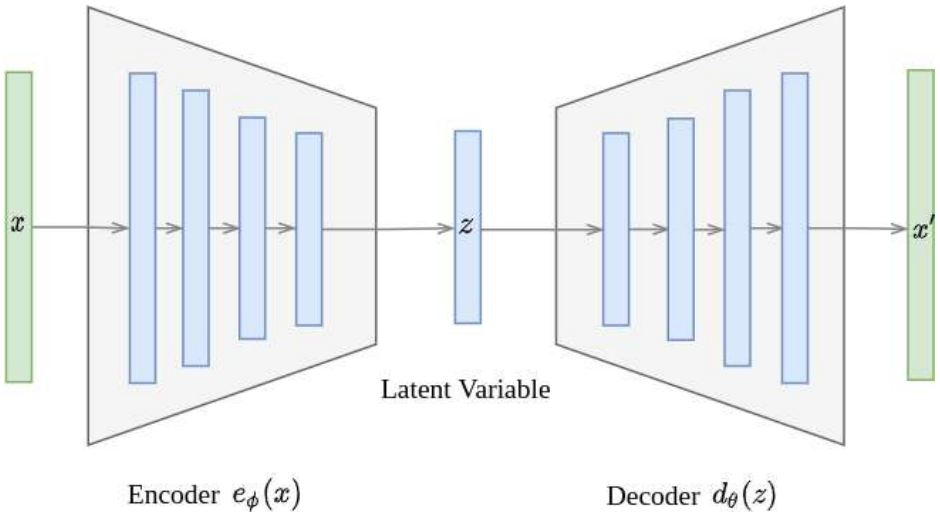


Figure 2.2: Architecture of AE

Parameters ϑ and ϕ are learned jointly by minimizing the reconstruction error using the notion of backpropagation. Here, the encoder maps the input x to the latent variable z. In this regard, there is no constraint on the distribution of the latent space.

Variational AutoEncoder (VAE)

Variational AutoEncoder [26] is a data generative model that has recently shown tremendous performance in producing highly realistic pieces of data, such as images [31], texts [32] and speeches [33]. Variational AutoEncoder [26] realizes a deep latent generative model $p_\vartheta(x, z) = p_\vartheta(x|z)p(z)$ consisting of an inference model $q_\phi(z|x)$ (encoder) and a generative model $p_\vartheta(x|z)$ (decoder). Here, the encoder compresses the data into a low-dimensional latent representation and the decoder reconstructs the data from the latent representation.

To exemplify the tasks of encoder and decoder here having a similarity with our previous example, let's again consider x as a 28×28-pixel photo of a handwritten digit. Here, contrasting to our earlier case, the encoder encodes the 784-dimensional (28×28) data into a latent (hidden) representation space z through providing parameters of the probability distribution of z. The parameters of the probability distribution are even much lesser than the 784 dimensions. Besides, again contrasting to our earlier case, here, the decoder takes the latent representation z as input and outputs the parameters of the probability distribution of the data. In the case of black and white picture of a

digit, the probability distribution of a single pixel can be a Bernoulli distribution.

The architecture of VAE is shown in Figure 2.3. VAE approximates the posterior $p_\vartheta(z|x)$ by an inference model [26] as follows.

$$p_\vartheta(z|x) \approx q_\phi(z|x) \tag{2.2}$$

VAE optimizes the evidence lower bound (ELBO) of the marginal log-likelihood of data [26] as follows.

$$\mathrm{L}_{\vartheta,\phi}(x) = \log p_\vartheta(x) - D_{KL}(q_\phi(z|x)||p_\vartheta(z|x)) \tag{2.3}$$

The first part of the ELBO is the log likelihood of $p_\vartheta(x)$, that is the probability to obtain the desired data x. The second part is the Kullback-Leibler (KL) divergence between the probability distribution $q_\phi(z|x)$ and the actual posterior distribution $p_\vartheta(z|x)$.

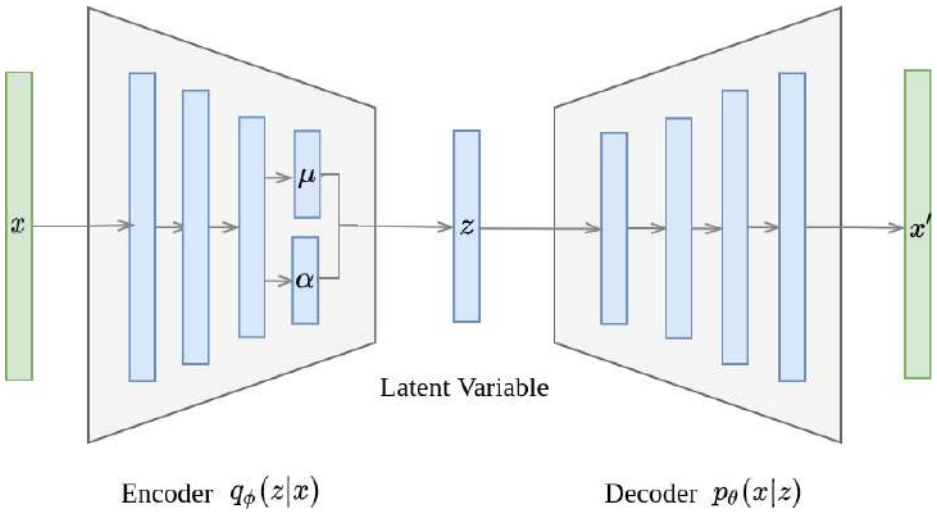


Figure 2.3: Architecture of VAE

Both inference and generative network of VAE are jointly trained to maximize the ELBO. VAE amortizes variational inference (VI) [34] by the encoder network. VAE uses fully-factorized Gaussian as the posterior distribution. However, fully-factorized Gaussian does not have enough expressive power and cannot properly capture complex posterior distribution. This causes approximation error [35]. Another problem of VAE is the amortization error that causes due to amortized inference of posterior distribution [36].

Variational Laplace AutoEncoder (VLAE)

Variational Laplace AutoEncoder [27] is a variant of VAE that uses full-covariance Gaussian as posterior distribution. VLAE enhances the expressive power of the posterior distribution and reduces amortization error using Laplace Approximation of the posterior distribution [27] as follows.

$$q(z|x) = \mathrm{N}(\mu, \Sigma), \ \textit{where}\ \Sigma^{-1} = -\nabla^2_z \log p_\vartheta(x, z)|_{z=\mu} \tag{2.4}$$

For Gaussian output ReLU network, local linearity is used to approximate the posterior. The architecture of VLAE is shown in Figure 2.4.

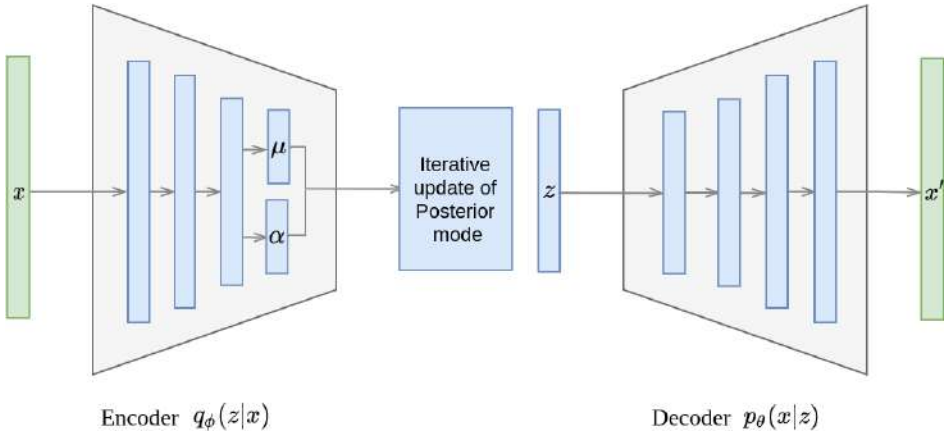


Figure 2.4: Architecture of VLAE

Posterior mode is iteratively searched for T steps under the linear assumption $g_\vartheta(\mu_t) \approx W_t\mu_t + b_t$ [27] as follows.

$$\mu_{t+1} = \sigma^{-1}(\sigma^{-1}W_t^T W_t + I)^{-1}W^T(x - b) \tag{2.5}$$

VLAE approximate the posterior distribution using $p_\vartheta(x|z) = \mathrm{N}(W_\mu z + b_\mu, \sigma^2 I)$ [27] as follows.

$$q(z|x) = \mathrm{N}(\mu, \sigma), \ \textit{where}\ \Sigma = (\sigma^{-2}W_\mu^T W_\mu + I)^{-1} \tag{2.6}$$

Attention Mechanism

Attention Mechanism allows neural networks to focus and give more "*Attention*" on the relevant parts of the input data as needed [28, 37, 38]. The notion of Attention is used to realize the fact that the encoder network can compress data, however, all of them are not equally important. Attention mechanism was originally introduced to improve the performance of Seq2Seq model for neural machine translation, and now, it is applied in different cases.

To exemplify, let's consider a case where we have to translate a English sentence "How was your day?" to the French version "Comment se passe ta journée?". What the Attention layer of a neural network will do for each word in the output sentence is map the important and relevant words from the input sentence and assign higher weights to these words, enhancing the accuracy of the output translation. Thus, the Attention layer ensures giving more attention to the more important parts.

A basic seq2seq approach consists of an encoder-decoder model, where the encoder analyzes the input data and compresses the information into a context vector of a fixed length (sentence embedding), and the decoder is computed with the context vector to emit the transformed output. Though, this architecture has shown its effectiveness in Seq2Seq models, it has one crucial drawback. The sentence embedding is generated in one vector. When the length of the input data increases, it becomes difficult for the model to capture the information in this vector. Thus, it has the inability to preserve longer input data as it tends to forget parts of it. The attention mechanism was introduced by Bahdanau [28] to solve the problem associated with fixed-length context vector in neural machine translation. As the encoder encodes every input sequence to the fixed-length context vector, the decoder does not have enough information for long or complex sentences. The attention mechanism solves this problem by creating shortcuts between the context vector and the entire input sequence. It permits the decoder to utilize the most relevant parts of the input sequence by a weighted combination of all of the encoded input vectors, with the most relevant vectors being given the highest weights.

Each input words is assigned a weight by the attention mechanism which is then used by the decoder to predict the next word in the sentence. The attention weights are computed by applying softmax function to the alignment scores of the input sequence. The alignment score, e_{ti}, that indicates how well the elements of the input sequence align with the current output at position, t, can be calculated by applying a feed forward network over the encoded hidden states, h_i, and the previous decoder output, s_{t-1}, shown in the following equation.

$$e_{ti} = tanh(W_1 h_i + W_2 s_{t-1}) \tag{2.7}$$

The attention weights a_{ti} are computed by applying softmax function to the alignment scores, e_{ti} as shown in the following equation.

$$a_{ti} = \frac{exp(e_{ti})}{\sum_{k=1}^{T} exp(e_{tk})} \tag{2.8}$$

The context vector, C_i, is computed by a weighted sum of all, T, encoder hidden states using the following equation.

$$c_i = \sum_{i=1}^{T} a_{ti} h_i \tag{2.9}$$

Chapter 3

Related Work

Various machine learning and deep learning-based approaches have been applied to implement NIDS. This section presents several related studies based on such NIDSs.

Machine Learning-based Approaches in NIDSs

Machine learning-based approaches are categorized in three categories namely supervised, unsupervised, and semi-supervised learning approaches [39]. Accordingly, NIDSs adopt three different categories of machine learning.

Supervised Learning Approaches

In supervised learning, the algorithms learn mapping functions from the input to the output using labelled training data [40]. Supervised learning approaches applied in NIDS include Support Vector Machine (SVM), Decision Tree (DT), K-Nearest Neighbour (KNN), Random Forest (RF), Artificial Neural Network (ANN), etc., [41, 40]. For example, Wagner et al., [42] leveraged one-class SVM to build a network anomaly detection system. Besides, the study in [15] proposed a detection model using the ant system for feature reduction and SVM for classification. Another study [43] built a least-square SVM based intrusion detection system using the features selected by own feature selection algorithm.

Kim et al., [11] proposed a hybrid intrusion detection method that incorporated a DT based misuse detection model and a one-class SVM based anomaly detection model through a decomposition structure. Here, the normal training data was decomposed into smaller subsets using DT and multiple one-class SVM models were built for the decomposed subsets. Another study [44] explored eight tree-

based classification algorithms for classifying network attacks. Here, DT was used for feature selection and RF was applied as a classifier.

Kuang et al., [45] proposed a Dependable Network Intrusion Detection System (DNIDS) based on the Combined Strangeness and Isolation Measure K-Nearest Neighbor (CSI-KNN) algorithm. Jadhav et al., [46] proposed distributed and parallel approaches using SVM, KNN, DT, and NB classifier to enhance the efficiency of detecting the intrusions. Besides, Manzoor et al., [47] proposed a detection system that used information gain and correlation-based ranking for feature reduction along with a classifier based on ANN. Amoordon et al., [48] proposed a method based on RF and KNN that can detect several attacks in wireless networks. Ouiazzane et al., [49] proposed a model based on multi-agent systems and DT model to detect DoS attacks in the UAV networks.

Unsupervised Learning Approaches

In unsupervised learning, the algorithms learn patterns and representations from unlabeled data []. Examples of unsupervised learning algorithms adopted in NIDS are K-means, Principal Component Analysis (PCA), Self-Organizing Map (SOM), etc. [41, 40]. In this regard, Bhuyan et al., [50] designed an outlier-based NIDS in which legitimate data were clustered using a K-means technique, and then, a reference point was computed for each cluster. With these points, samples are classified as attacks if they differ by a certain threshold value. Another study [51] adopted a Principal Component Analysis (PCA) algorithm for feature selection and a Support Vector Machine as a classifier to select the optimum feature subset. The study in [52] proposed a lightweight DDoS flooding attack detection solution, which used emulation to build a NOX based network in SDN using Self-Organized Map (SOM). Wang et al., [53] designed an IDS using the ensemble of AutoEncoders to learn the network data and an Isolation Forest algorithm to perform the anomaly detection.

Semi-supervised Learning Approaches

Semi-supervised learning approaches use a small amount of labeled data and a large amount of unlabeled data during their training. Semi-supervised learning algorithms are required where labeling data is challenging or expensive. In this regard, a semi-supervised Support Vector Machine was used to enhance the accuracy of NIDSs in [54]. In another study [55], two semi-supervised classification methods namely Spectral Graph Transducer and Gaussian Fields were used to detect unknown attacks, and a semi-supervised clustering method namely MPCK-means was used to improve the performance

of the detection systems.

Deep Learning-based Approaches

Deep learning algorithms present an advancement to artificial neural networks that use multiple layers of neurons to progressively extract higher-level features from the raw input data. Deep learning algorithms can learn the representation of data with various levels of generalization. Deep learning methods such as Deep Belief Network (DBN), Deep Neural Network (DNN), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Generative Adversarial Network (GAN), etc. have been applied successfully in computer vision, speech recognition, natural language processing, machine translation, information retrieval, and many other fields [56]. In recent years, this method has been widely used to identify network intrusion and achieved remarkable detection results.

Tang et al., [19] build a Deep Neural Network (DNN) model for flow-based anomaly detection systems and train the model with the NSL-KDD [1] dataset. In this work, they used six basic features and gained an accuracy of 75.75%. Besides, Ma et al., [57] proposed a hybrid method, named SCDNN, leveraging Spectral Clustering (SC) and Deep Neural Network (DNN). Here, SC divided the original training dataset into training subsets, and the subset was used to train multiple DNN classifiers. The method achieved 72.64% and 44.55% accuracy on the NSL-KDD (KDDTest+) [1] and NSL-KDD (KDDTest-21) [1] datasets respectively. Additionally, Niyaz et al., [20] proposed a Stacked AutoEncoder (SAE) based DL approach only for DDoS detection systems in Software Defined Network (SDN). The authors claimed that they have achieved a binary classification accuracy of 99.82% and an 8-class classification accuracy of 95.65%. Further, Javaid et al., [58] used Self-Taught Learning (STL), a deep learning-based technique, on the NSL-KDD dataset. They achieved an accuracy of 79.10% for the 5-class classification.

The study in [21] presented a Recurrent Neural Network (RNN) based model with a soft-max classifier. In this study, the model was evaluated on the NSL-KDD dataset. They achieved 81.29% accuracy for the test set KDDTest+ for five-class classification. In another study [59], the authors applied LSTM in flow-based network for intrusion detection and compared performance with various machine learning classifiers. They achieved 74.26% accuracy for five-class classification.

Khan et al., [60] proposed a deep learning model based on Stacked AutoEncoder with a soft-max classifier for network intrusion detection. The model comprised two decision stages. The first stage

was responsible for classifying network traffic as normal or abnormal using a probability score value. This classification is then used in the final decision stage as an additional feature for detecting the normal state and other classes of attacks. The proposed model was evaluated on the KDD 99 [2] and UNSW-NB15 [61] datasets, and achieved 99.996% and 89.134% accuracy on the datasets respectively. Besides, in [62], Li et al. used LSTM and Gated Recurrent Unit (GRU) with a variable number of hidden layers along with a Broad Learning System (BLS) to build network anomaly detection models. The BGP [63] and NSL-KDD datasets were used to evaluate the performance of the proposed models in terms of training time, accuracy, and F1 score. On the BGP dataset, the experimental results showed that RNN and BLS models can provide the best accuracy and F1 score over the range of 90%-95%. On the NSL-KDD dataset, the experimental results showed that the best performance can be obtained using LSTM4 and GRU3 along with the CFBLS (BLS with cascades of mapped features). Additionally, Vinayakumar et al., [64] employed distributed DNN models to develop a scalable and hybrid intrusion detection model called Scale-Hybrid-IDS-AlertNet (SHIA). The proposed SHIA could effectively monitor a large number of network-level and host-level events to automatically identify malicious attacks to provide network administrators with appropriate alerts. Experimental tests on various benchmark IDS datasets showed that the proposed model performed well compared to other traditional machine learning classifiers.

Mighan et al., [65] combined the advantages of deep network and machine learning methods, using an Stacked AutoEncoder (SAE) network for latent feature extraction, followed by several machine learning methods, such as Support Vector Machine, Random Forest, Decision Tree, and Naive Bayes for intrusion detection. In another study [66], Zhou et al., proposed a Variational Long Short-Term Memory (VLSTM) learning model that detects intrusion anomalies efficiently based on feature reconstruction. Here, an encoder–decoder neural network associated with a variational reparameterization scheme was designed to learn the low-dimensional feature representation from high-dimensional raw data. The proposed VLSTM resulted in an accuracy of 89.5% on UNSW-NB15 dataset. Further, Khan et al., [67] proposed a Convolutional Recurrent Neural Network (CRNN) based hybrid framework namely HCRNNIDS to predict malicious attacks in the network. In HCRNNIDS, a Convolutional Neural Network performs convolution to capture local features and a Recurrent Neural Network captures temporal features to improve the system's performance. HCRNNIDS attained an accuracy of up to 97.75% on CSE-CIC-IDS2018 [68] dataset with 10-fold cross-validation.

Saurabh et al., [69] developed models based on variants of LSTMs namely stacked LSTM and

bidirectional LSTM for intrusion detection systems for IoT networks. Alqahtani et al., [70] proposed a hybrid optimized LSTM approach for IoT networks. In this approach, CNN was used to extract the temporal and spatial correlated features of IoT networks, and the optimized LSTM was used to predict the different attacks in the networks.

Variational AutoEncoder-based Approaches

Variational AutoEncoder (VAE) is a deep learning based probabilistic generative model that has achieved significant improvements in different areas [71, 72, 73, 74, 75]. Some recent studies have incorporated VAE in intrusion detection. For example, Jinwon et al., [76] presented an anomaly detection system based on VAE using reconstruction probability. Besides, Kawachi et al., [77] employed VAE for supervised anomaly detection. Sun et al., [78] also used VAE to learn sparse representations for anomaly detection. Additionally, Lopez–Martin et al., [79] proposed a conditional VAE (CVAE) based intrusion detection system, called Intrusion Detection-Conditional Variational AutoEncoder (ID-CVAE). To classify a sample, ID-CVAE reconstructed samples associated with each class label. Subsequently, ID-CVAE used the Euclidean distance to measure the similarity between reconstructed samples and original samples. Yanq et al., [24] proposed a method that combined an improved conditional VAE (ICVAE) with a DNN classifier. ICVAE was used to learn sparse representations between network data features and classes. ICVAE achieved 85.97% and 89.08% accuracy on NSL-KDD and UNSW-NB15 dataset respectively. Xu et al., [80] proposed a method named Log-Cosh Conditional Variational Autoencoder (LCVAE) using conditional VAE and log hyperbolic cosine (log-cosh) loss function. Neuschmied et al., [81] proposed a two-stage approach combining a filtering method with VAE using reconstruction probability. In the first step, a fast anomaly detector filters out data that do not belong to any anomaly. In the second step, the remaining data are then evaluated by a more specific VAE based anomaly detector providing more accurate decision. Sabeel et al.,[82] proposed an adversarial incremental learning approach. The approach was based on a hybrid model consisting of a conditional VAE and a Generative Adversarial Network (GAN). Lova et al., [83] proposed a conditional VAE with an adaptive loss function. This study replaced the classical reconstruction loss function with a flexible loss function for the purpose of minimizing reconstruction error.

Attention-based Approaches

Attention is one of the most influential ideas in deep learning research that imitates cognitive attention. This mechanism has been extensively used in different fields such as natural language processing [84] and computer vision [85]. Some recent studies have applied attention mechanism in detecting network attacks too. For example, Yang et al., [86] used bidirectional RNN with attention mechanism to extract relevant features and give explainable results in identifying dominant attributes. Tang et al., [87] proposed a method based on Stacked AutoEncoder, DNN, and attention mechanism, named SAAE-DNN. Here, SAAE selected the needed features from the intrusion dataset and initialized weights of DNN to improve the intrusion detection accuracy. The proposed SAAE-DNN showed an accuracy of 87.74% on NSL-KDD dataset. Laghrissi et al., [88] proposed a detection model based on LSTM and attention mechanism. They also used four feature reduction algorithms namely Chi-Square, UMAP, PCA, and Mutual Information. The experimental results showed that using attention with all features and using PCA with three components exhibited the best performance, achieving an accuracy of 99.09% and 98.49% for binary and multi-class classification on NSL-KDD dataset respectively.

Limitations of the Existing Studies

As present in the previous sections, different types of approaches have been investigated in the literature. However, leveraging DNN using the exploitation of Variational Laplace AutoEncoder is yet to be explored in the literature to the best of our knowledge. Therefore, in this paper, we explore a new intrusion detection system based on VLAE and DNN. Here, considering the fact that VLAE has higher expressiveness and lower amortized error than VAE [27], we extend VLAE to incorporate class labels in the encoder of VLAE so that the latent representations of the feature data of different class labels are separated in latent space and we can generate attack samples based on class labels. We present out proposed methodology in the next section.

Chapter 4

Methodology of Our Study

In this chapter, we present the details of our proposed approach for network intrusion detection. Here, we first elaborate our proposed two models in Section 4.1 and 4.2 respectively. Then, we discuss four different phases of our approach namely data preprocessing, training CVLAE and CVLAAE, data augmentation, and attack classification in Section 4.3.

Proposed Conditional Variational Laplace AutoEncoder (CVLAE)

CVLAE is generally based on VLAE model that incorporates the class label as an input in the encoder. Incorporating the class label enables us to generate new data from separated latent space according to class labels. Here, the encoder $q_\phi(z|x, y)$ is conditioned on input features x and class labels y. Besides, the decoder $p_\vartheta(x|z, y)$ is conditioned on latent variables z and class labels y. The encoder learns the parameters of the intermediate probability distribution, $q_\phi(z|x, y)$ from the input features and class labels. Latent variable z is sampled from the intermediate distributions and passed as the input to the decoder. The decoder learns the parameters of the probability distribution $p_\vartheta(x|z, y)$ from the latent variables and the class labels. The output samples x' are generated from the learned distribution of the decoder. Figure 4.1 shows the structure of the proposed CVLAE model. Here, the evidence lower bound objective function of CVLAE is formulated as follows.

$$\mathrm{L}_{\vartheta,\phi}(x, y) = \log p_\vartheta(x|y) - D_{KL}(q_\phi(z|x, y)||p_\vartheta(z|x, y)) \tag{4.1}$$

The ELBO function consists of two parts: The first part is the log-likelihood of data and the second part is the KL divergence between the approximate distribution and the true posterior distribution.

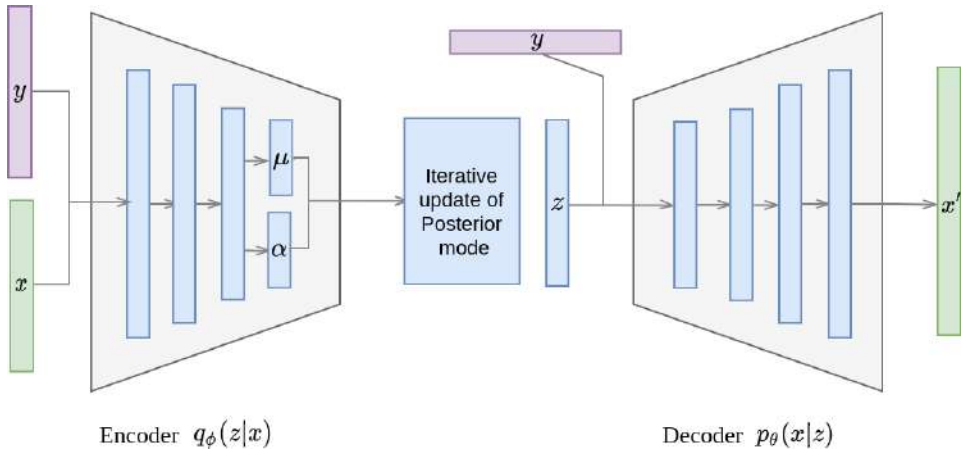


Figure 4.1: Architecture of CVLAE

The posterior mode is initialized by the inference ReLU network. Then the model iteratively searches for the posterior mode over T steps where density is concentrated and approximates a full-covariance Gaussian posterior at the mode. We use multivariate Gaussian as the distribution for $p_\vartheta(x|z, y)$ and use multivariate standard normal distribution N (**0**, **I**) as the prior $p_\vartheta(z)$. Here, the model is trained to optimize the ELBO using Adam [89] optimization algorithm.

Proposed Conditional Variational Laplace Attention Autoencoder (CVLAAE)

We extend the CVLAE model by adding an attention mechanism layer between the encoder and the latent layer. The original input data passes through the encoder, and the data get compressed. The layer calculates the attention vector of each simplified feature. The attention vector and the feature are multiplied to generate data input to the latent layer. When the attention vector finds that a specific feature does not contribute to the prediction, it sets the specific value in the vector to 0 causing the network to forget the feature. Figure 4.2 shows the structure of our proposed CVLAAE model.

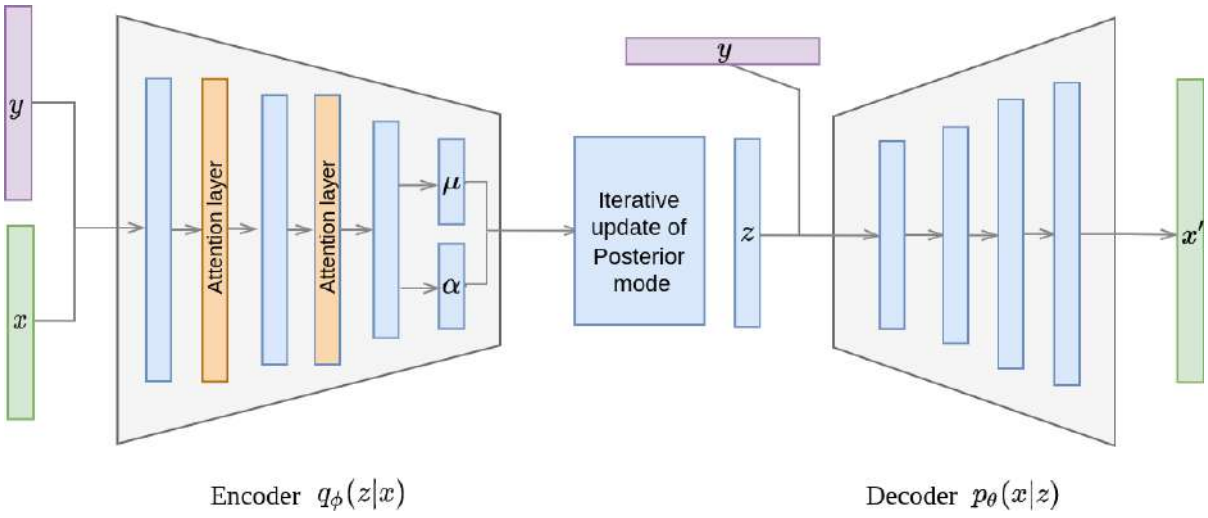


Figure 4.2: Architecture of CVLAAE

The attention mechanism layer contains a set of attention weights, and each weight denotes the importance value of the corresponding feature. Important features are selected from the input data through weighted summation according to the following equations.

$$M = tanh(W_a x' + b_a) \tag{4.2}$$

$$a_i = softmax(M_i) \tag{4.3}$$

$$v = \sum_{i=0}^{D} x'_i a_i \tag{4.4}$$

Here x' is the feature vector, W_a is the weight, b_a is the offset value, and a_i is the probability distribution of M_i normalized by softmax, and D is the number of samples. a_i is taken as the attention vector. It is multiplied by x' and summed up to obtain the more representative feature vector v. Finally, the probability distribution is taken as the weight and summarized with x' to obtain a more representative feature vector v, which eliminates unnecessary features. The attention layer focuses on important features and improves the performance of the network.

Four Phases of Our Proposed Approach

In this section, we discuss our proposed approach for network intrusion detection model. The proposed approach consists of four phases namely data preprocessing, training CVLAE and CVLAAE, data augmentation, and attack classification. Figure 4.3 shows the last three steps in sequence.

Phase-1: Data Preprocessing

Categorical features need to be converted to numerical values before being fed to deep learning models. For this purpose, we use one-hot encoding [90, 91, 92, 93, 94] and numeralize the categorical features. One-hot encoding maps each categorical value into a new categorical column and assigns a value of one or zero to those columns. We represent the categorical values as binary vectors. We set the column corresponding to the categorical feature as one and all other columns as zero. To exemplify, the NSL-KDD dataset contains three categorical features, protocol type (i.e., tcp, udp, and icmp), service (i.e., ftp, http, ssh, etc.) and flag (i.e., REJ, RSTO, RSTOS0, etc.). After one-hot encoding, they are mapped into 84-dimensional binary vectors.

The numerical features have different scales, and data normalization is required to improve the performance and training stability of ML models. We use the min-max [95, 96, 97, 98] normalization method to transform features to be on a similar scale. Then, we convert all numerical feature values to the range of 0 to 1 using the following equation.

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.5}$$

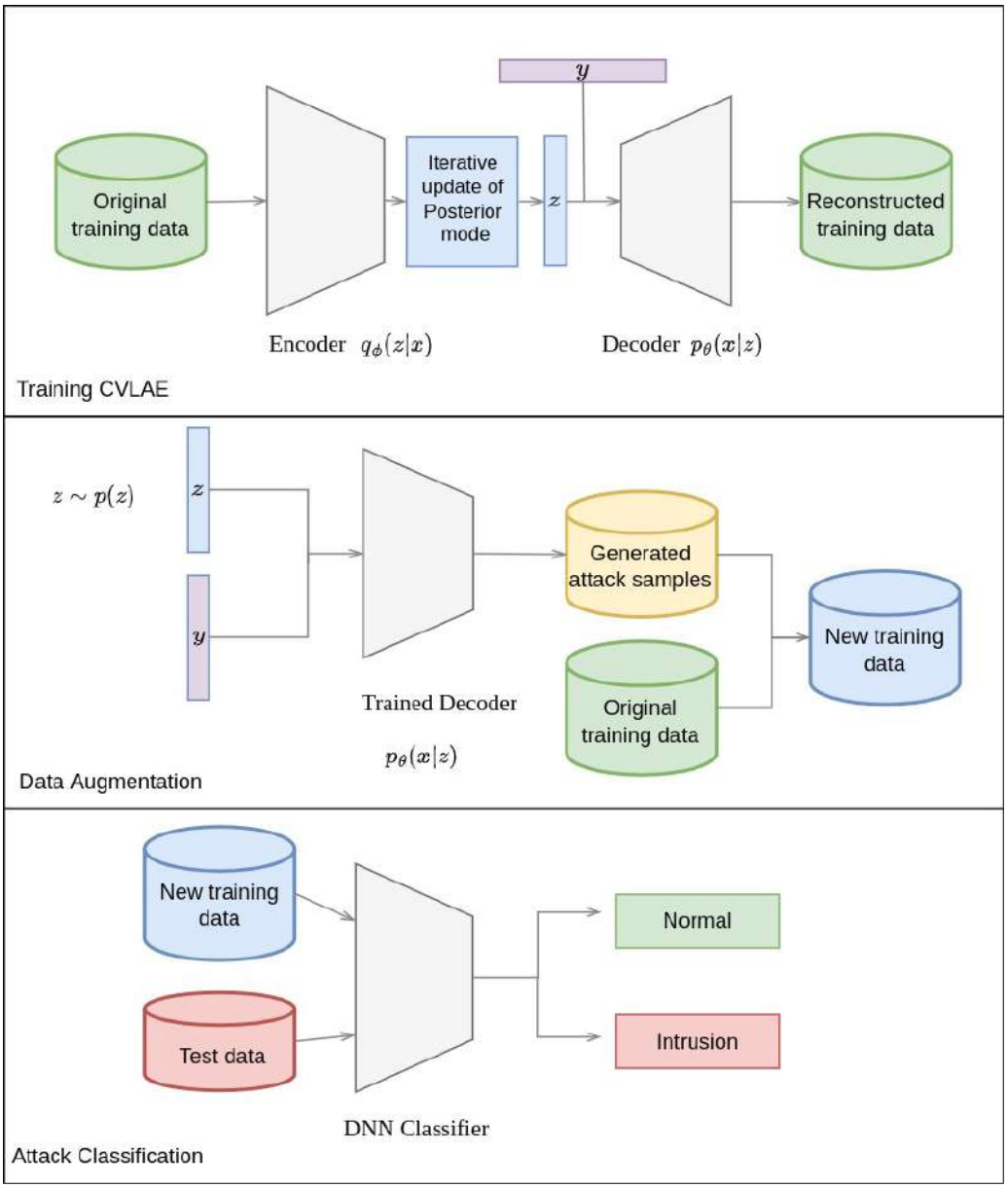


Figure 4.3: Development and training model, data augmentation, and attack classification in our proposed approach

Phase-2: Development and Training CVLAE and CVLAAE

We train our proposed CVLAE and CVLAAE models using benchmark datasets [1]. We train the models to minimize the difference between the reconstructed and original data by optimizing the ELBO function. The original input feature data and class labels pass through the encoder, and they

[1]we use NSL KDD train+ [1] and KDD CUP 10% [2] training datasets for this purpose.

get mapped to parameters (mode and variance) of the latent distribution $q(z|x)$. Then, the posterior mode is iteratively updated for T steps. Subsequently, the latent variable z is sampled from the distribution and passed through the decoder. The decoder maps the latent variable to the probability distribution $p(x|z)$, from which we get the reconstructed feature vector. We concatenate the feature vector with the one-hot encoded class labels and feed them into the models. Each encoder and decoder of the models have two hidden layers of 32 dimensions. The latent variable has 16 dimensions. Using such models, we update the posterior mode iteratively for ten steps. We use Adam optimizer [89] as the optimization algorithm to update network weights iteratively. Additionally, in our case, the activation function of all hidden layers is ReLU [99], and the activation function of output layer of the decoder is Sigmoid.

After training these models, we calculate the log-probability of each training sample (x_i, y_i) as follows.

$$\log p_\vartheta(x_i|z, y_i) = -n\log\sigma - \frac{n}{2}\log(2\pi) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_{i,j} - \mu)^2 \tag{4.6}$$

Where $x_{i,j}$ represents the j^{th} feature value of x_i and n represents the number of features. The parameters μ and σ are the mean and standard deviation of the posterior distribution respectively.

We calculate the minimum log probability of the k^{th} class as follows.

$$minP_k = \min\{\log p_\vartheta(x_i|z, y_i), \text{ for each } y_i \in \text{class } k\} \tag{4.7}$$

Phase-3: Data Augmentation

We use multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ as the prior distribution. Besides, for data augmentation, we sample the latent variable z from $\mathcal{N}(\mathbf{0}, \mathbf{I})$, concatenate it with specified class label y', and feed it into the trained decoder network. We feed the newly generated sample (x', y') into the model, and we calculate the log probability of the newly generated sample $\log p_\vartheta(x'|z, y')$ using Equation 4.6. If the log probability is greater than the minimum log probability $minP_k$ of the specified class k, then we merge the sample into the original training data. Otherwise, we discard the sample.

$$S = \begin{cases} S \cup \{x', y'\}, & \text{if } y' \text{ in class } k \text{ and } \log p_\vartheta(x'|z, y') \geq minP_k \\ S, & \text{Otherwise} \end{cases}$$

Attack Classification

We use a DNN as the intrusion classifier. The classifier is trained on the original and generated data. We use ReLU as the activation function of all hidden layers and Softmax [100] as the activation function of the output layer of the classifier. Besides, we optimize the classifier by Adam optimizer. Finally, we feed the test dataset into the trained DNN classifier to classify attacks.

We detail our proposed intrusion detection approach in Algorithm 1.

Algorithm 1 Proposed Intrusion Detection Approach

1: **Input:** Training dataset $S = (x, y)$, latent variable Z, and learning rate lr.

2: **Output:** The final classification results.

3: **Data preprocessing**: Categorical features are one-hot encoded, and numerical features are scaled to [0,1] using min-max normalization.

4: Train CVLAE on training dataset S with multivariate Gaussian distribution as prior $p(z)$ using learning rate lr and Adam optimization method.

5: Calculate the minimum log probability for each attack type in the training data set using Equation 4.7.

6: Sample z from $\mathrm{N}(\mathbf{0}, \mathbf{I})$, specify the attack class y, and feed them into the trained CVLAE decoder to generate a new attack sample x'. If the reconstruction loss of the newly generated sample is less than the maximum reconstruction loss, The newly generated sample (x', y') is merged into the training data set S.

7: Train the DNN classifier on the merged dataset.

8: Evaluate the DNN classifier on the test data set and return the result.

Chapter 5

Evaluation of Our Proposed Approach

In this chapter, we present an experimental evaluation of our proposed approach for network intrusion detection. Here, we compare results of our proposed approach against that of three recent state- of-the-art approaches. We perform our experimentation over two benchmark datasets pertinent to network intrusion. To elaborate our experimental outcomes, first, we describe our experimental setup, evaluation metrics, and the intrusion datasets used in our experimental evaluation. After that, we will present the performance comparison in detail.

Experimental Setup

For training and testing purposes, we use a Linux Pop! OS 20.04 LTS laptop having AMD Ryzen 7 CPU and 16 GB RAM. To speed up the training process, we use one NVIDIA GeForce RTX 2060 GPU having 8 GB RAM. As the training process requires huge time, we also train our models in Google Colaboratory. We use the Anaconda environment having Tensorflow 2.3 framework with Python 3.7 and CUDA 11.2.

Each encoder and decoder of CVLAE and CVLAAE has two hidden layers having 50 and 25 dimensions respectively. Besides, the latent variable has ten dimensions. With three dimensions, We update the posterior mode iteratively for ten steps. Here the activation function of hidden layers is ReLU, and the activation function of output layer of the decoder is Sigmoid. Our DNN classifier has five hidden layers. The activation function of hidden layers is ReLU, and the activation function of the output layer is Softmax in the classifier. We preprocess and normalize the data before feeding to the models. We train CVLAE and CVLAAE on NSL-KDD (KDDTrain+) and KDD-CUP (10%)

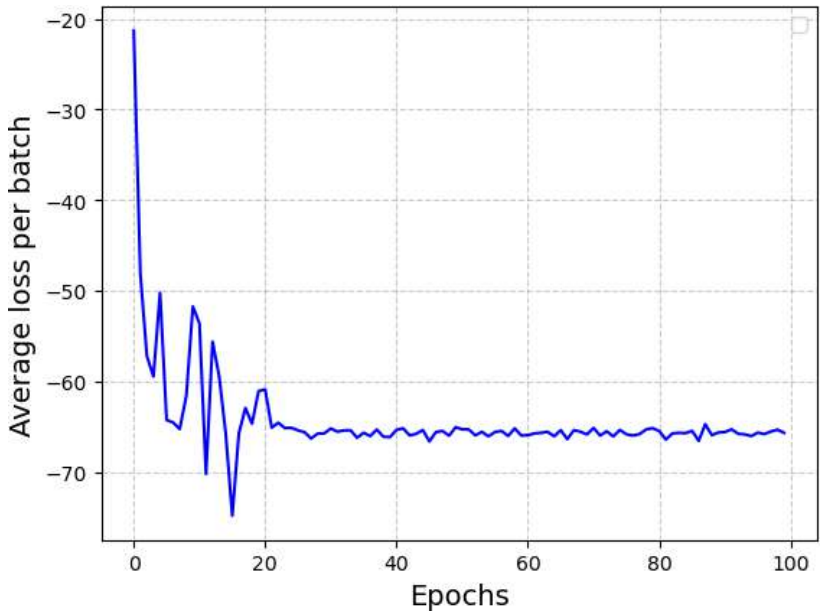


Figure 5.1: Training loss of CVLAAE

training set, and train the DNN classifier on the training and generated datasets. We use Adam [89] optimizer with a learning rate of 0.001 for all networks. We evaluate the DNN classifier on NSL-KDD (KDDTest+) and KDD-CUP(10%) testing set. Epochs of all the models are set to 100. Figure 5.1 and 5.2 show the training loss of CVLAAE and DNN respectively. Figure 5.3 shows the training accuracy of the DNN classifier.

Network Intrusion Datasets

We evaluate our proposed approach on two benchmark datasets namely KDD CUP 99 [2] and NSL-KDD [1]. We present a brief elaboration on each of these datasets below.

KDD CUP 99 Dataset

The KDD CUP 99 [2] intrusion detection dataset is based on the DARPA 98 [101] dataset, which was generated in a simulation in the US Air Force military network. DARPA 98 dataset was collected as binary tcpdump files from nine weeks of network traffic. The training data comprised about five million connection records, and the testing data comprised around two million records. In 1999, the DARPA 98 dataset was processed into 41 features and one label for each connection record by the BRO-IDS

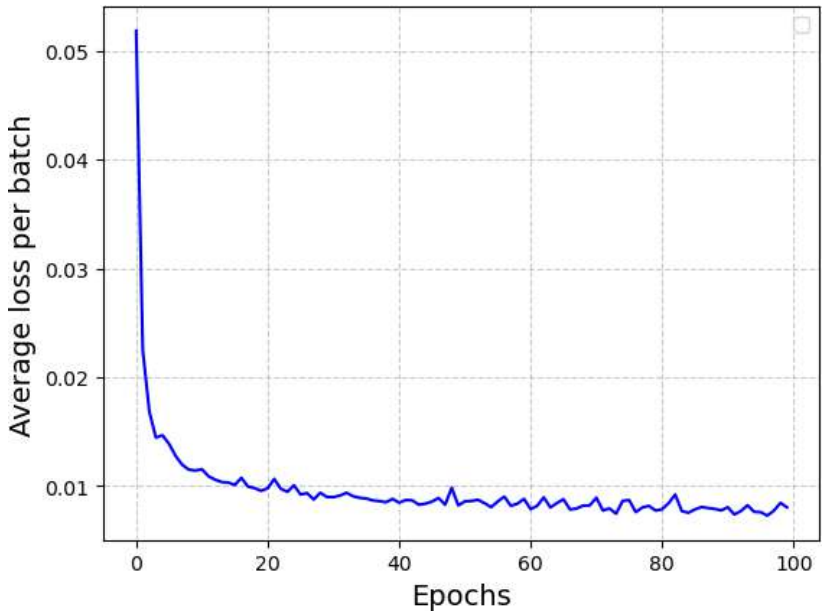


Figure 5.2: Training loss of DNN

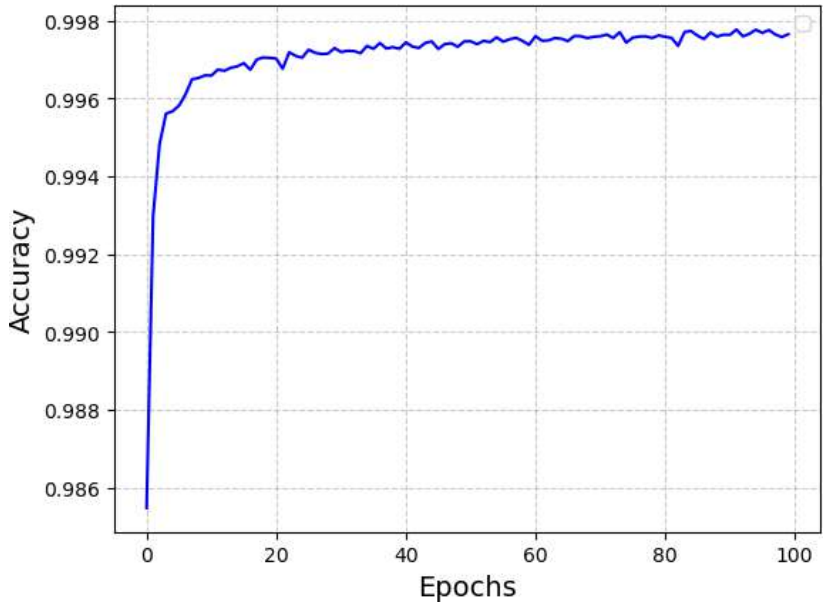


Figure 5.3: Training accuracy of DNN

tool and named KDD CUP 99 dataset. Among the 41 features, 34 features are continuous, and seven are discrete. The features are grouped into three categories namely basic, content and traffic. The basic features are derived from the headers of the network packets. The content features are captured from the payloads of the network packets. The traffic features are obtained from information about

Table 5.1: Features in the KDD CUP [2] dataset

No	Variable Name	Type	No	Variable Name	Type
1	Duration	Continuous	22	Is_guest_login	Discrete
2	Protocol_type	Discrete	23	Count	Continuous
3	Service	Discrete	24	Srv_count	Continuous
4	Flag	Discrete	25	Serror_rate	Continuous
5	Src_bytes	Continuous	26	Srv_serror_rate	Continuous
6	Dst_bytes	Continuous	27	Rerror_rate	Continuous
7	Land	Discrete	28	Srv_rerror_rate	Continuous
8	Wrong_fragment	Continuous	29	Same_srv_rate	Continuous
9	Urgent	Continuous	30	Diff_srv_rate	Continuous
10	Hot	Continuous	31	Srv_diff_host_rate	Continuous
11	Num_failed_logins	Continuous	32	Dst_host_count	Continuous
12	Logged_in	Discrete	33	Dst_host_srv_count	Continuous
13	Num_compromised	Continuous	34	Dst_host_same_srv_rate	Continuous
14	Root_shell	Continuous	35	Dst_host_diff_srv_rate	Continuous
15	Su_attempted	Continuous	36	Dst_host_same_src_port_rate	Continuous
16	Num_root	Continuous	37	Dst_host_srv_diff_host_rate	Continuous
17	Num_file_creations	Continuous	38	Dst_host_serror_rate	Continuous
18	Num_shells	Continuous	39	Dst_host_srv_serror_rate	Continuous
19	Num_access_files	Continuous	40	Dst_host_rerror_rate	Continuous
20	Num_outbound_cmds	Continuous	41	Dst_host_srv_rerror_rate	Continuous
21	Is_host_login	Discrete	42	Normal or Attack	Discrete

previous connections. Table 5.1 shows the feature list of KDD CUP 99. Here, the training set contains 22 attack types and the testing set contains 15 attack types. Attack types are grouped into four categories namely User to Root, Remote to Local, Denial of Service, and Probe. Table 5.2 presents definition of the attack types. The attack types and their counts in “10% KDD” training dataset are summarized in Table 5.3, which shows high imbalance in the dataset. Here, most records are of type normal, Denial of Service, or Probe. U2R and R2L attack types rarely appear in the dataset.

NSL-KDD Dataset

NSL-KDD [1] is an enhanced version of the KDD CUP 99 dataset. It is one of the most widely used datasets for evaluating intrusion detection systems. The total number of records in the training set (KDDTrain+) is 127,973, and in the testing set (KDDTest+) is 22,544. Each traffic record in the NSL-KDD dataset contains 41 features. According to feature characteristics, the attacks in the

Table 5.2: Attack types in the KDD CUP 99 [2] dataset

Attack Type	Definition of the Attack Type
Denial of Service (DoS)	Attacks, such as SYN flood, smurf, and teardrop exhaust the target system to obstruct legitimate users from using a service provided by system
Remote to Local (R2L)	Attacks are designed to get right of access to a machine without authorization, for example, the password-guessing attack
User to Root (U2R)	Attacks give super-user (root) access to the normal user, for instance, buffer overflow attacks
Probe	Attacks are designed to obtain information about the target client, for instance, port scanning and ping-sweep attacks

Table 5.3: Attacks in training and testing set of KDD CUP 99 10% [2] dataset

Category	**Training Set**		**Testing Set**	
	Count	**Attack type**	**Count**	**Attack type**
Normal	97277	No attack	60593	No attack
DoS	391458	back, pod, land, smurf, neptune, teardrop	229853	back, pod, mailbomb, land, smurf, neptune, teardrop, apache2, processtable, udpstorm
Probe	4107	ipsweep, nmap, portsweep, satan	4166	ipsweep, nmap, mscan, saint, portsweep, satan
R2L	52	guess-passwd, imap, multihop, warezclient, phf, warezmaster, spy, ftp-write	70	guess-passwd, multihop, warezmaster, snmpgetattack, xlock, sendmail, named, worm, imap, phf, ftp-write, snmpguess, xsnoop
U2R	1126	buffer-overflow, perl, loadmodule, rootkit	16347	buffer-overflow, perl, ps, loadmodule, rootkit, sqlattack, xterm, httptunnel

NSL-KDD dataset can be classified into four types: User to Root (U2R), Denial of Service (DOS), Root to Local (R2L), and Probing attacks (Probe). The number of attacks of type R2L and U2R is very low. Table 5.4 summarizes the types and their counts in the dataset. Several attacks exist in the testing set (KDDTest+) but not in the training set (KDDTrain+). The difference between training and testing sets provides a realistic theoretical basis for intrusion detection.

Table 5.4: Attacks in the training and testing set of NSL-KDD [1] dataset

Category	Training Set		Testing Set	
	Count	Attack type	Count	Attack type
Normal	67343	No attack	9711	No attack
DoS	45927	back, pod, land, smurf, neptune, teardrop	7458	back, pod, mailbomb, land, smurf, neptune, teardrop, apache2, processtable, udpstorm
Probe	11656	ipsweep, nmap, portsweep, satan	2421	ipsweep, nmap, mscan, saint, portsweep, satan
R2L	995	guess-passwd, imap, multihop, warezclient, phf, warezmaster, spy, ftp-write	2754	guess-passwd, multihop, warezmaster, snmpgetattack, xlock, sendmail, named, worm, phf, ftp-write, snmpguess, xsnoop
U2R	52	buffer-overflow, perl, loadmodule, rootkit	200	buffer-overflow, perl, loadmodule, rootkit, ps, sqlattack, xterm, httptunnel

Table 5.5: Confusion matrix for classification problems

	Actual Negative	*Actual Positive*
Predicted Negative	TN	FP
Predicted Positive	FN	TP

Metrics for Performance Evaluation

In order to effectively evaluate the performance, we use accuracy, precision, recall, specificity, and F1 score as evaluation metrics. These metrics are measured from the confusion matrix, which is built for a classification problem, as shown in Table 5.5. Here, TP (True Positive) is the number of correctly classified attack traffic records, and TN (True Negative) is the number of correctly classified normal traffic records. FP (False Positive) is the number of normal traffic records which are misclassified as attack traffic, and FN (False Negative) is the number of attack traffic records which are misclassified as normal traffic. Next, based on the measures of TP, TN, FP, and FN, we explain these evaluation metrics in detail.

Accuracy (AC): Defined as the percentage of correctly classified records over the total number of

records, as shown in Eq. 5.1.

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

Precision (P): Defined as the number of correctly predicted attacks divided by the total number of predicted attacks, as shown in Eq. 5.2.

$$P = \frac{TP}{TP + FP} \tag{5.2}$$

Recall (R): Defined as the number of correctly predicted attacks divided by the total number of actual attacks, as shown in Eq. 5.3.

$$R = \frac{TP}{TP + FN} \tag{5.3}$$

Specificity (S): Defined as the number of correctly predicted normal records divided by the total number of normal records, as shown in Eq. 5.4.

$$S = \frac{TN}{TN + FP} \tag{5.4}$$

F1 Score (F1): Defined as the harmonic mean of precision and recall, as shown in Eq. 5.5.

$$F1 = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{5.5}$$

In general, F1 score is considered as the most important metric for evaluating NIDS methods. F1 score is more useful for performance evaluation when dealing with unbalanced datasets.

Besides, In a multi-class classification problem, there are two possible ways to calculate results namely aggregated and One-vs.-Rest results [79]. In an aggregated result, a summary result over all classes is calculated. In One-vs.-Rest result, focus is given on a particular class and other classes are considered as a single class altogether. We have used One-vs.-Rest results, provided by scikit-learn [102] library, to calculate the precision, recall, specificity, and F1 score.

Evaluation Results and Performance Comparison

This section presents the detection performance of our proposed approach and shows a comparison with other methods. We compare our method with three state-of-the-art methods namely DNN [58],

Table 5.6: Number of samples in original and synthesized data from NSL-KDD [1] dataset

Category	Original data samples	Synthesized data samples	Total
Normal	67343	0	67343
DoS	45927	21416	67343
Probe	11656	55687	67343
R2L	995	66348	67343
U2R	52	67291	67343

SAAE-DNN [87], and CVAE-DNN [24] based on results obtained over the two benchmark datasets. Among these three state-of-the-art methods under comparison, SAAE-DNN [87] and CVAE-DNN [24] are well-known data-augmentation methods and more related to our method than the other methods in the literature. This happens as the SAAE-DNN and our method have incorporated attention mechanism in the encoder. Besides, as we condition VLAE on class labels, we compare it with CVAE-DNN that also used conditional VAE. Moreover, we compare with DNN [58] to evaluate our effectiveness of data augmentation compared to the baseline benchmark method.

The Detection Performance

NSL-KDD [1] presents an imbalanced dataset. Here, many attack types present in the testing dataset do not appear in the training dataset. Besides, the number of attacks under the categories of R2L and U2R is significantly low. However, most classification machine learning algorithms require a near-equal number of samples in every class. Accordingly, machine learning models trained on imbalanced dataset generally result in bias toward the majority classes and high false positive rate for minority classes. To solve the problem of imbalanced data, we use the data generation algorithm involving our proposed CVLAE and CVLAAE. Table 5.6 shows the number of data generated for each class using our proposed approach.

We present values of the performance metrics for CVLAE and CVLAAE in Table 5.7 - Table 5.8 and 5.9 - 5.10 respectively. We have achieved 77% and 96.50% accuracy for CVLAE-DNN on NSL-KDD and KDD CUP 99 datasets respectively. For CVLAAE-DNN, we have achieved 80% and 96.14% accuracy on NSL-KDD and KDD CUP 99 datasets respectively. On NSL-KDD dataset, CVLAE has gained precision of 83.00% and 78.00% in R2L and U2R attack types respectively, which are higher than CVLAAE has. However, CVLAAE has gained higher precision of 95.31%, 84.65%, and 72.96% in DoS, Probe, and normal attack types. CVLAAE has higher recall, F1 score, and specificity in all

Table 5.7: Performance results of our CVLAE-DNN for NSL-KDD [1] (KDDTest+) dataset

Category	Accuracy	Precision	Recall	Specificity	F1 score
DoS	77%	92.00%	84.00%	96.39%	% 88.00
Probe		80.00%	70.00%	97.89%	75.00%
R2L		83.00%	17.00%	99.52%	28.22%
U2R		78.00%	7.00%	99.98%	13.00%
Normal		69.00%	93.00%	68.38%	79.00%

Table 5.8: Performance results of our CVLAE-DNN for KDD CUP 99 [2] (KDD 10%) dataset

Category	Accuracy	Precision	Recall	Specificity	F1 score
DoS	96.50%	99.00%	99.00%	99.56%	99.00%
Probe		98.89%	97.27%	99.52%	98.07%
R2L		92.74%	94.57%	100.00%	93.64%
U2R		95.08%	94.80%	99.56%	94.94%
Normal		99.88%	99.96%	99.95%	99.92%

Table 5.9: Performance results of our CVLAAE-DNN for NSL-KDD [1] (KDDTest+) dataset

Category	Accuracy	Precision	Recall	Specificity	F1 score
DoS	80%	95.31%	85.60%	97.92%	90.19%
Probe		84.65%	73.84%	98.39%	78.88%
R2L		82.58%	18.18%	99.47%	29.80%
U2R		75.86%	22.00%	99.94%	34.11%
Normal		72.96%	96.78%	72.86%	83.20%

Table 5.10: Performance results of our CVLAAE-DNN for KDD CUP 99 [2] (KDD 10%) dataset

Category	Accuracy	Precision	Recall	Specificity	F1 score
DoS	96.14%	99.00%	99.44%	99.56%	99.22%
Probe		97.91%	99.03%	99.07%	98.46%
R2L		96.59%	93.04%	100.00%	94.78%
U2R		93.10%	81.92%	99.45%	87.16%
Normal		99.58%	99.87%	99.81%	99.72%

the attack types than CVLAE has. On KDD CUP dataset, CVLAE has higher precision of 98.89% and 95.08% in Probe and U2R attack types respectively, but CVLAAE has gained higher precision of 96.59% in R2L attack type. CVLAE has achieved higher recall in R2L and U2R attack types. In other attack types, both have achieved similar recall. Both models have gained almost similar results in terms of F1 score and specificity.

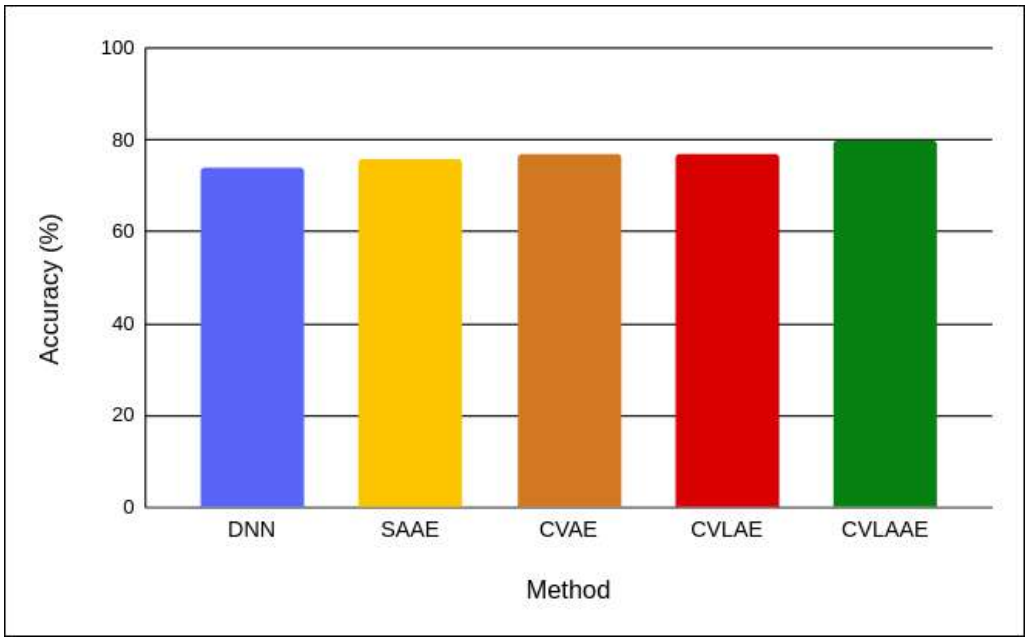


Figure 5.4: Comparison of accuracies of different methods on NSL-KDD test set [1]

Performance Comparison with Other Alternatives

We compare outcomes of our proposed approach with that of other alternatives namely DNN, SAAE-DNN, and CVAE-DNN in terms of accuracy, precision, recall, specificity, and F1 score. Figure 5.4 - 5.13 show values of the performance metrics for these three alternatives along with our proposed CVLAE and CVLAAE on NSL-KDD and KDD 99 datasets respectively. As per these figures, we can achieve 6%, 4%, and 3% higher accuracy than DNN, SAAE-DNN, and CVAE-DNN respectively on NSL KDD test set using CVLAAE-DNN. Besides, on KDD CUP 99 test set, the accuracy of CVLAAE-DNN is 17%, 16%, and 7% higher than DNN, SAAE-DNN, and CVAE-DNN, respectively. Besides, CVLAE-DNN gains 2% and 1% higher accuracy than DNN and SAAE-DNN respectively on NSL-KDD test set and 17%, 16%, and 7% higher accuracy than DNN, SAAE-DNN, and CVAE-DNN respectively on KDD CUP 99 test set. CVLAAE-DNN achieves 3% higher accuracy than CVLAE-DNN on NSL KDD and has similar on KDD CUP for the two methods.

Figure 5.6 shows that CVLAAE-DNN has achieved the highest precision for Dos, Probe and Normal traffic types as well as the second highest precision for R2L and U2R on NSL-KDD test set. CVLAE-DNN has slightly higher precision for R2L and U2R than CVLAAE-DNN. Besides, Figure 5.7, 5.8, and 5.9 show that CVLAAE-DNN achieves the highest recall, specificity, and F1 score values for all types of attacks respectively. Figure 5.10 presents that both CVLAAE-DNN and CVLAE-DNN

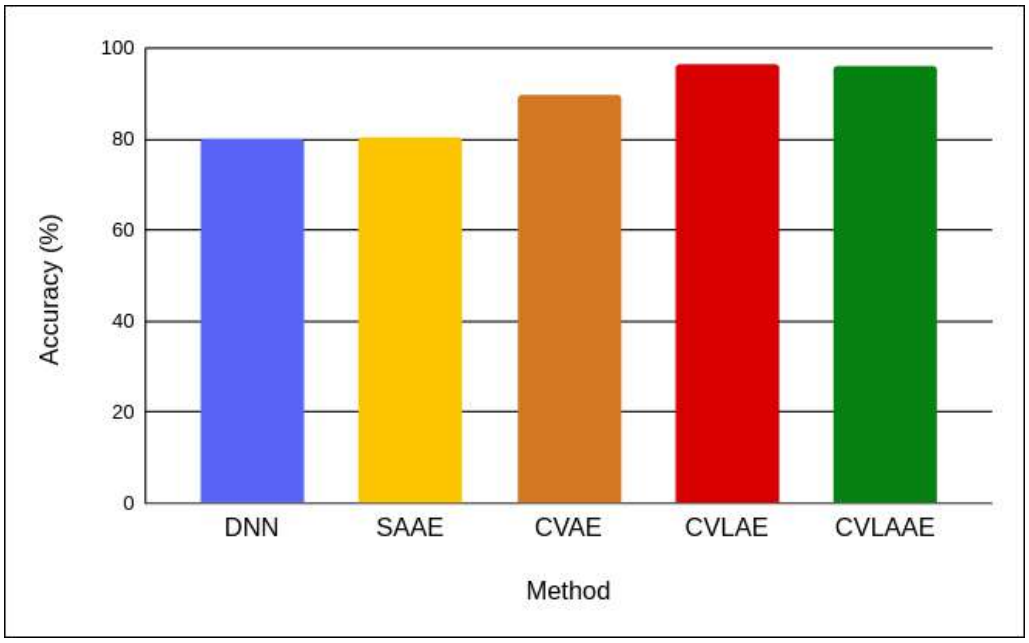


Figure 5.5: Comparison of accuracies of different methods on KDD CUP 99 test set [2]

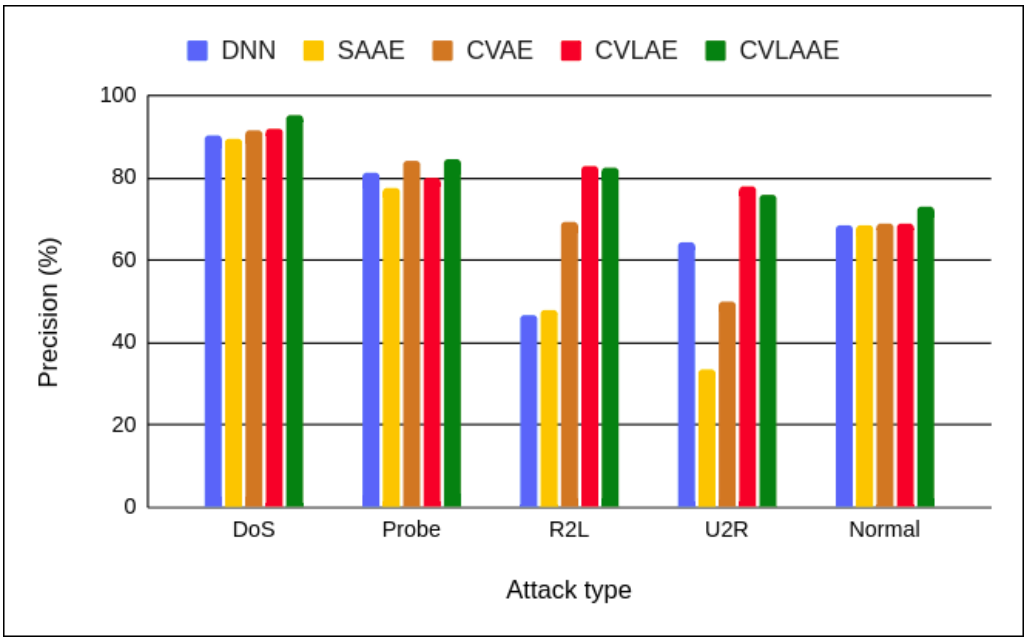


Figure 5.6: Comparison of precisions of different methods on NSL-KDD test set [1]

achieve much higher precision than the other methods on KDD CUP 99 test set. Figure 5.11, 5.12, and 5.13 depict similar results for recall, specificity, and F1 score on KDD CUP 99 test set.

Table 5.11 - 5.18 show the percentages of improvement achieved by CVLAAE-DNN over the other

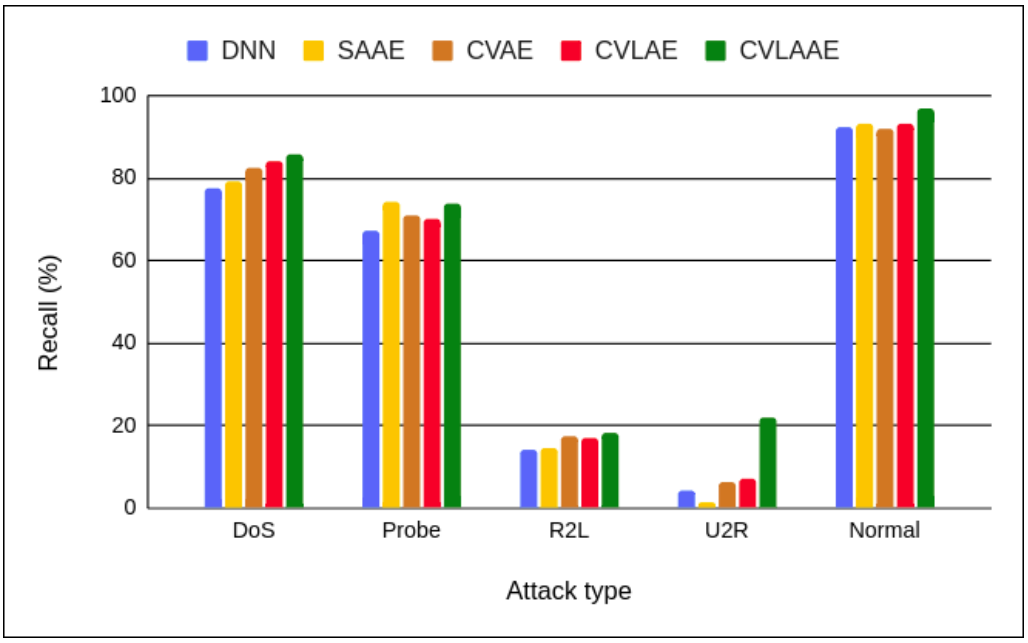


Figure 5.7: Comparison of recalls of different methods on NSL-KDD test set [1]

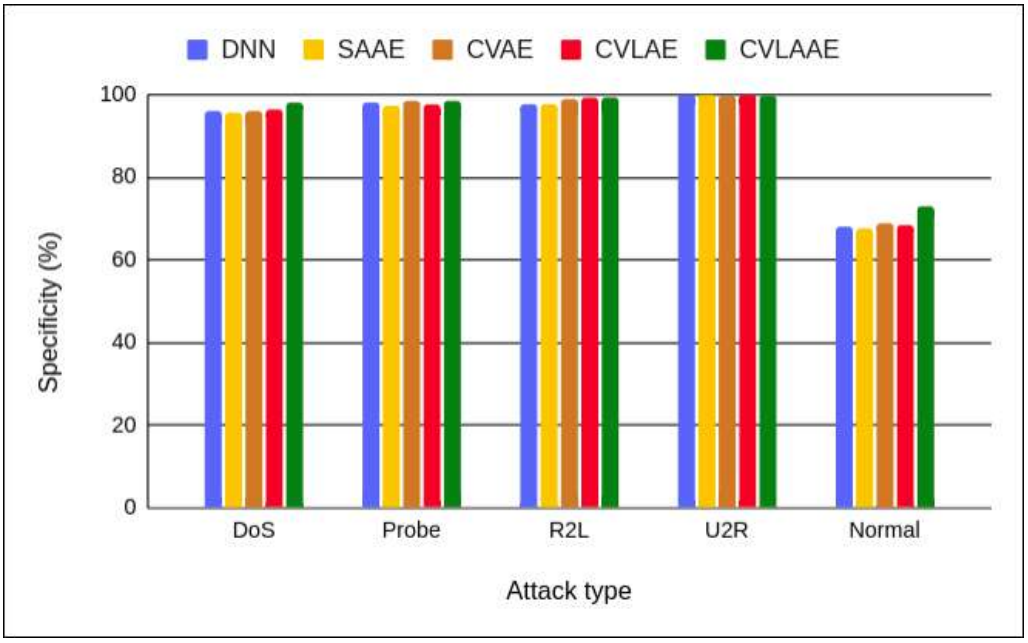


Figure 5.8: Comparison of specificities of different methods on NSL-KDD test set [1]

four methods. We present the improvement in terms of precision, recall, specificity, and F1 score both for NSL-KDD and KDD CUP 99 datasets. The extents of improvement confirm that our proposed approach achieves noteworthy increase in performance in most of the cases.

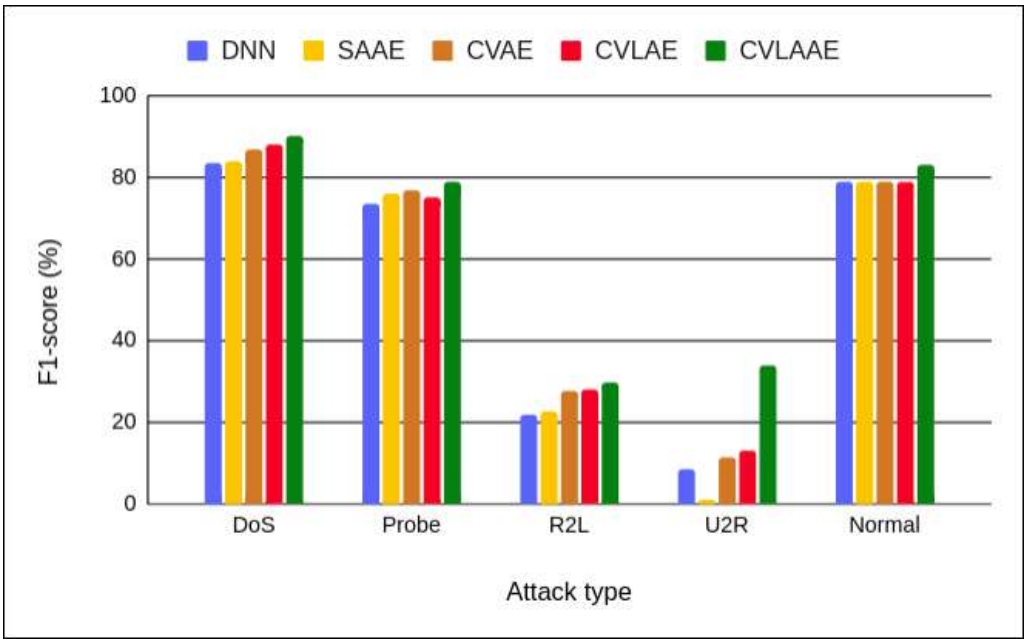


Figure 5.9: Comparison of F1 scores of different methods on NSL-KDD test set [1]

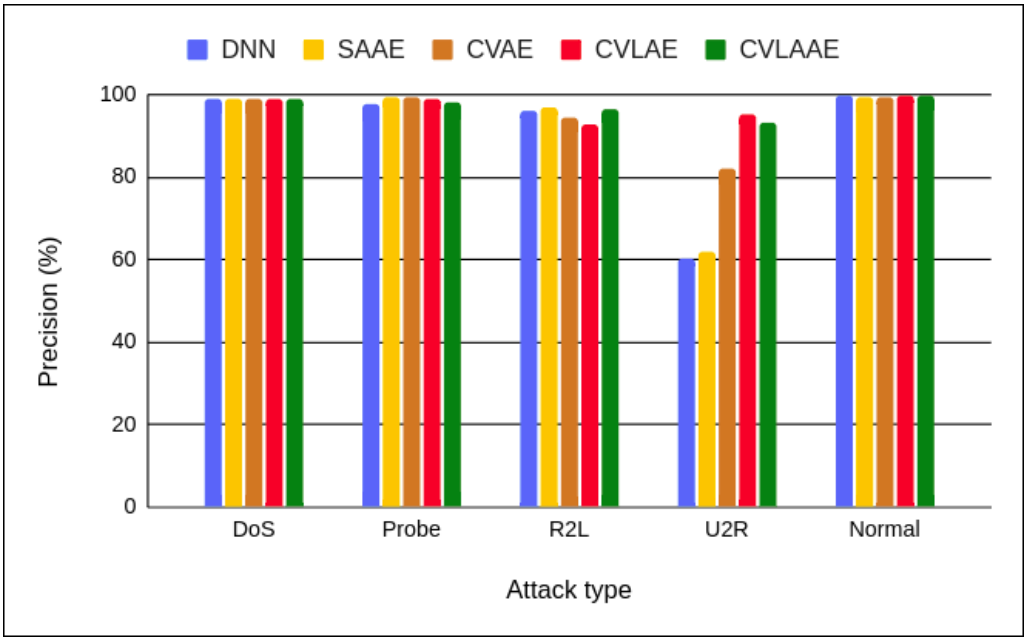


Figure 5.10: Comparison of precisions of different methods on KDD CUP 99 test set [2]

Nonetheless, to analyze whether our proposed approach introduce any substantial time penalty or not, we measure the time required for generating each data instance by each of the approaches in our experimental setup. To do so, we generate ∼20K instances by each of the classification approaches

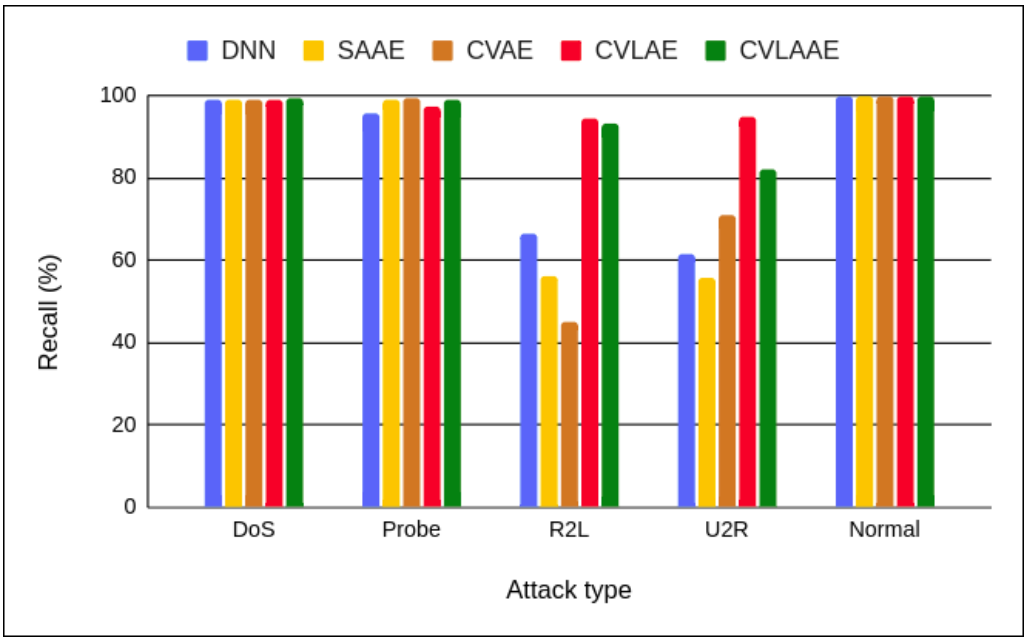


Figure 5.11: Comparison of recalls of different methods on KDD CUP 99 test set [2]

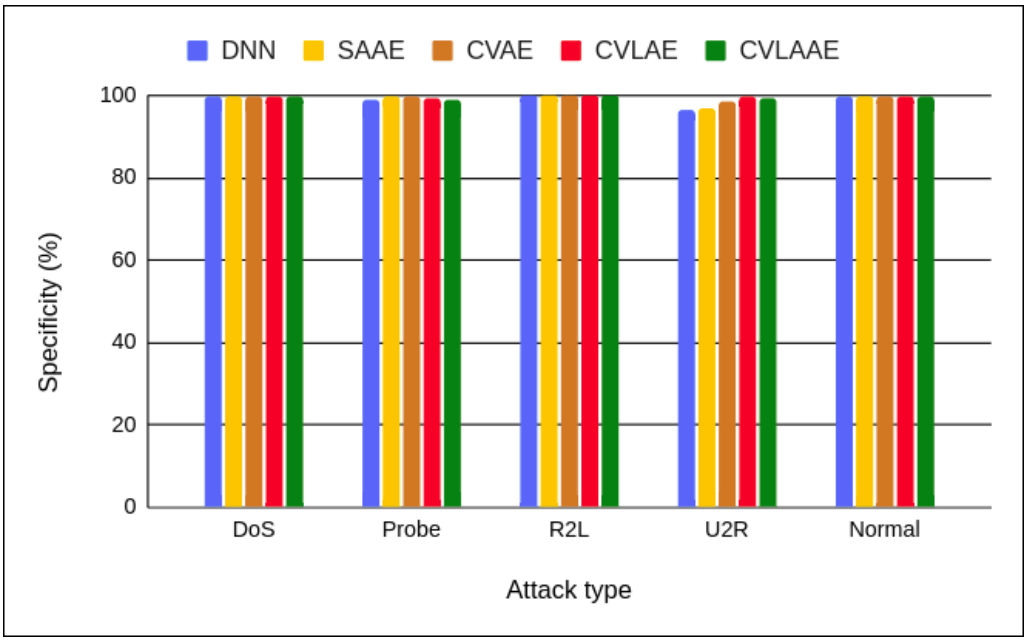


Figure 5.12: Comparison of specificities of different methods on KDD CUP 99 test set [2]

and take an average over the data generation time values to measure the time required for generating a single data instance. Figure 5.14 shows the comparison of average data generating time for SAAE, CVAE, CVLAE, and CVLAAE. The figure shows that the data generating time for each of the

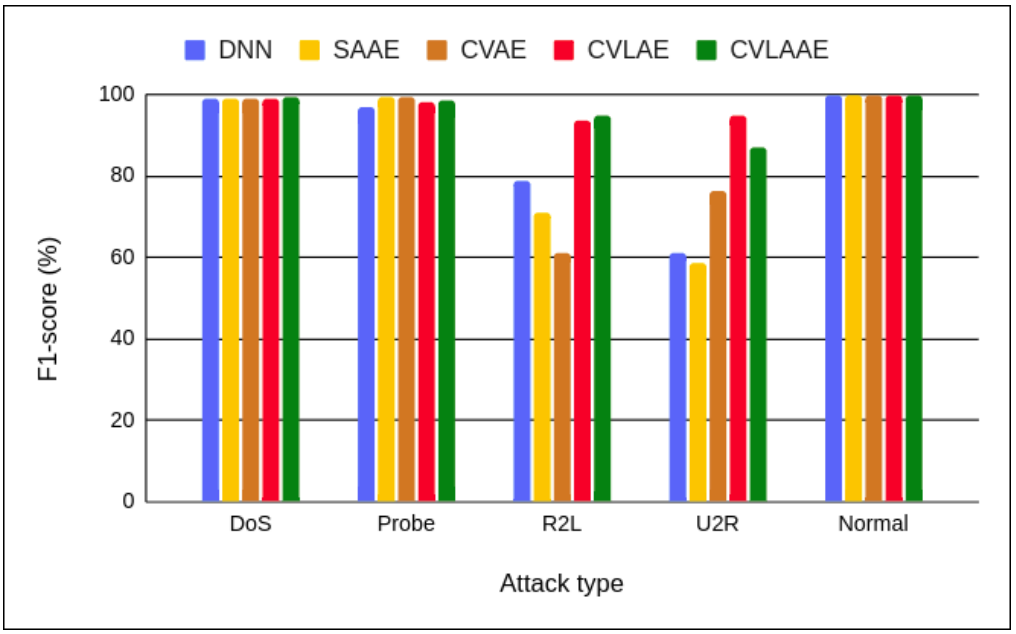


Figure 5.13: Comparison of F1 scores of different methods on KDD CUP 99 test set [2]

Table 5.11: Percentage of improvement achieved by CVLAAE-DNN in precision on NSL-KDD test set[1]

Method	DoS	Probe	R2L	U2R	Normal
DNN	5.04%	3.23%	35.84%	11.57%	4.38%
SAAE	5.73%	6.92%	34.73%	42.53%	4.38%
CVAE	3.95%	0.55%	13.11%	25.8%	3.90%
CVLAE	3.31%	4.65%	-0.42%	-2.14%	3.96%

Table 5.12: Percentage of improvement achieved by CVLAAE-DNN in recall on NSL-KDD test set [1]

Method	DoS	Probe	R2L	U2R	Normal
DNN	7.93%	6.65%	4.00%	17.50%	4.24%
SAAE	6.37%	-0.29%	3.47%	20.53%	3.73%
CVAE	3.09%	3.01%	0.85%	15.50%	4.64%
CVLAE	1.60%	3.84%	1.18%	15.00%	3.78%

approaches including our proposed ones is very small (in the scale of microseconds). Besides, due to integration of the iterative inference of posterior, CVLAE introduces a small time penalty resulting in a bit slower data generation compared to SAAE and CVAE (by 7% and 11% respectively). Moreover, due to the integration of attention mechanism in addition to the iterative inference of posterior in

Table 5.13: Percentage of improvement achieved by CVLAAE-DNN in specificity on NSL-KDD test set [1]

Method	DoS	Probe	R2L	U2R	Normal
DNN	2.06%	0.23%	1.71%	-0.04%	4.94%
SAAE	2.47%	0.94%	1.70%	-0.04%	5.12%
CVAE	1.78%	0.00%	0.53%	0.00%	4.10%
CVLAE	1.53%	0.49%	-0.05%	-0.04%	4.48%

Table 5.14: Percentage of improvement achieved by CVLAAE-DNN in F1 score on NSL-KDD test set [1]

Method	DoS	Probe	R2L	U2R	Normal
DNN	6.69%	5.26%	8.05%	25.70%	4.42%
SAAE	6.10%	2.99%	7.30%	33.12%	4.24%
CVAE	3.48%	1.98%	2.07%	22.61%	4.25%
CVLAE	2.19%	3.88%	1.58%	21.11%	4.20%

Table 5.15: Percentage of improvement achieved by CVLAAE-DNN in precision on KDD CUP 99 test set [2]

Method	DoS	Probe	R2L	U2R	Normal
DNN	0.00%	0.16%	0.38%	32.67%	-0.15%
SAAE	0.00%	-1.60%	-0.49%	31.40%	0.14%
CVAE	0.00%	-1.38%	2.18%	10.88%	0.25%
CVLAE	0.00%	-0.98%	3.85%	-1.98%	-0.30%

Table 5.16: Percentage of improvement achieved by CVLAAE-DNN in recall on KDD CUP 99 test set [2]

Method	DoS	Probe	R2L	U2R	Normal
DNN	0.44%	3.20%	26.45%	20.38%	-0.07%
SAAE	0.44%	-0.09%	37.00%	26.15%	-0.08%
CVAE	0.44%	-0.31%	48.01%	10.77%	-0.06%
CVLAE	0.44%	1.76%	-1.53%	-12.88%	-0.09%

CVLAAE, CVLAAE introduces a bit more time penalty compared to CVLAE. However, even in the case of CVLAAE, the introduced time penalty remains small resulting in only a bit slower data generation compared to SAAE and CVAE (by 10% and 14% respectively). Thus, we can conclude that none of the CVLAE and CVLAAE introduces any significant time penalty, and therefore, they are equally applicable for real-time data generation tasks.

Table 5.17: Percentage of improvement achieved by CVLAAE-DNN in specificity on KDD CUP 99 test set [2]

Method	DoS	Probe	R2L	U2R	Normal
DNN	0.22%	1.68%	16.08%	26.18%	-0.12%
SAAE	0.22%	-0.86%	23.72%	28.57%	0.03%
CVAE	0.22%	-0.86%	33.81%	10.87%	0.09%
CVLAE	0.22%	0.39%	1.14%	-7.78%	-0.20%

Table 5.18: Percentage of improvement achieved by CVLAAE-DNN in F1 score on KDD CUP 99 test set [2]

Method	DoS	Probe	R2L	U2R	Normal
DNN	0.00%	0.04%	0.00%	3.08%	-0.07%
SAAE	0.00%	-0.72%	0.00%	2.57%	0.06%
CVAE	0.00%	-0.62%	0.00%	0.84%	0.11%
CVLAE	0.00%	-0.45%	0.00%	-0.10%	-0.13%

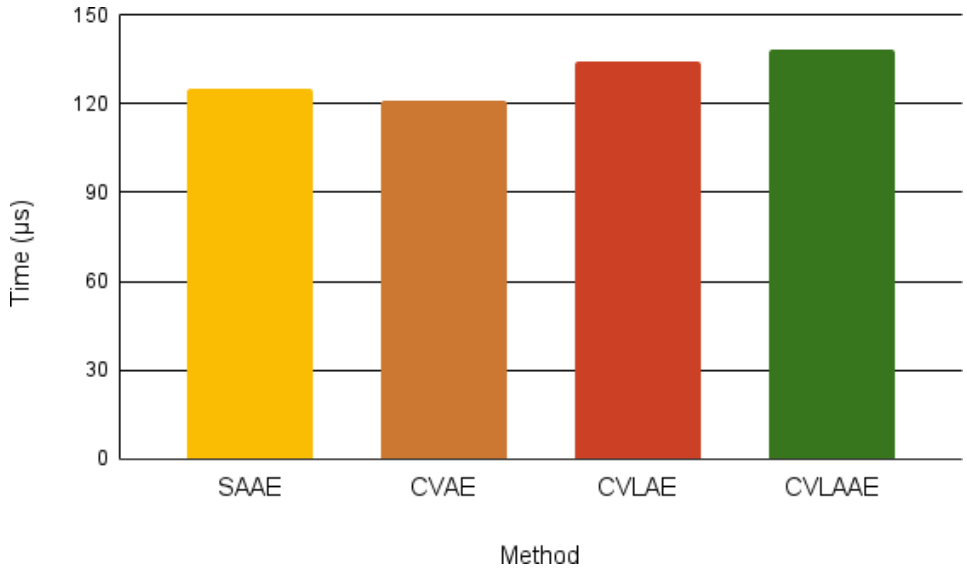


Figure 5.14: Comparison of average data generation time per instance

Chapter 6

Discussion

In this chapter, we will qualitatively compare our approach with other existing approaches available in the literature. Besides, We will also discuss applications and limitations of this study.

Comparison with Other Existing Approaches

In the previous chapter, we have already quantitatively compared our approach with two other data generative methods namely CVAE-DNN [24] and SAAE-DNN [87] along with one DNN [58] classifier. CVAE-DNN used an improved version of conditional VAE. It embedded class label y only in the decoder network so that the encoder network can be used to initialize parameters in the DNN classifier. Here, the decoder probability distribution is conditional on the latent variable z and class label y, and the encoder is only conditional on input feature data x. Besides, it used Multivariate Gaussian distribution for the encoder $Q(z|x)$ and multivariate Bernoulli distribution for the decoder $P(x|z, y)$. On the other hand, in our proposed CVLAE, we condition both encoder and decoder on class label y to better reconstruct features according to class labels. For both encoder and decoder, we use Multivariate Gaussian distribution. We do so as VLAE approximates posterior distribution with full-covariance Gaussian distribution; therefore, this distribution offers more expressive power than VAE. Accordingly, from Figures 5.6 - 5.13, we can find that CVLAE-DNN has higher precision, recall, specificity, and F1 score in minority attack types, where data balancing through new data generation impacts the most.

Additionally, SAAE-DNN [87] combined Stacked AautoEncoder (SAE), attention mechanism, and DNN. SAE is built by stacking autoencoder layer by layer, where the autoencoder is data specific. The

autoencoder can only encode data similar to the data presented in the training set. This happens as it cannot learn the probabilistic nature of data. There is no regularization on the values or distribution of the latent variables. The latent space may not be continuous, and therefore, the autoencoder does not have data generative capability. Accordingly, SAAE-DNN only used the weights of the trained SAAE to initialize the weights of the DNN classifier. As a result, experiments show that SAAE-DNN mostly attains a much lower F1 score in U2R attacks compared to both CVLAE-DNN and CVLAAE-DNN.

It is worth mentioning that, in a few cases of some of the performance metrics, CVLAAE-DNN exhibit slightly worse performance than the other methods. For example, in the case of NSL-KDD dataset, CVLAAE-DNN has a bit lower specificity in U2R attack type. This happens following the reality that a trade-off between different performance metrics such as precision, recall, specificity, etc., can often occur, as already reported in the literature [103, 104, 105]. Similarly, we get a trade-off between recall and specificity in our case. Here, as CVLAAE-DNN has achieved substantially higher recall in minority attacks, it has slightly lower specificity in some cases. Similar types of trade-off occur in a few other cases too.

Finally, between our two proposed approaches CVLAE-DNN and CVLAAE-DNN, CVLAAE-DNN mostly exhibits better performances in all the attack types. This happens as the addition of attention mechanism in CVLAE improves the detection performance.

Applications of This Study

Modern networks demand advanced security measures to ensure reliable and trusted service. As network attacks are becoming more sophisticated, adaptive protection technologies are now becoming more important to mitigate the threats. In this regard, NIDSs take places at various locations in a network to monitor traffic to and from all devices on the network. When placed at a strategic point or points within a network to monitor traffic to and from all devices on the network, an NIDS will perform an analysis of passing traffic and match the traffic that is passed on the subnets to the library of known attacks. Once an attack is identified or abnormal behaviour is sensed, the alert can be sent to the administrator. Here, the introduction of adaptive technologies can enable the NIDSs to filter out attacks that are little known. In this case, this study can be applied and leverage the filtration of little known attacks in the NIDSs.

NIDSs can be applied in a network having a distributed or centralized architecture. The dis-

tributed architecture may comprise of multiple instances of NIDSs placed at different positions of a network [106, 107, 108, 109]. In a resource constrained environment, e.g., an IoT system, NIDSs are usually placed in a distributed way across the network. Due to the resource constrained nature of these systems, the NIDSs must be lightweight and work independently. Conversely, the centralized architecture consists of a single NIDS conventionally deployed at a single position in the network that analyzes all the traffic of the network [110, 111, 112, 113]. Centralized networks, e.g., a Software Defined Network (SDN), can follow the centralized architecture. Such an architecture [114] is particularly important to consider due to their potential security attacks [115, 116]. Here, our study presented in this work will facilitate identifying new or zero-day attacks.

Limitations of This Study

Our proposed methods have been trained and tested on two benchmark datasets. Though the benchmark datasets are widely used for evaluating different intrusion detection systems, they do not contain different recent network attack types. As the network attacks are evolving rapidly and new attack types are generating now and then, our method requires testing in more recent real network settings.

Our proposed method presents a standalone anomaly-based method. It can be combined with signature-based methods to further improve its detection performance. Hierarchical anomaly-based methods can also be used to provide even better detection performance. Moreover, there is no feedback approach in our method. Feedback approach can be incorporated to strengthen the system.

Chapter 7

Future Work

Our future work will focus on improving the performance more for minority attacks and new attacks. To do so, we will explore other variants of VAE such as β-VAE [117, 118, 119] and VQ-VAE [120, 121] for generating new attack samples. To reconstruct samples more effectively, we will try to improve latent variable representation. Here, we plan to use adversarial learning methods to reconstruct attack samples better. The rationale behind this plan is that, using adversarial learning methods, attacks can be synthesized more for diversifying attack samples. This will increase our classification accuracy.

We also plan to build relevant methods to extract important flows and features from network traffic. The methods will choose important network flows from many packets, and then, extract significant features related to intrusive behaviour.

Besides, we plan to incorporate our proposed method with signature-based and other anomaly-based methods in a hierarchical manner. We will also try to apply our method in large-scale applications such as cloud computing, Internet of Things, and software-defined networks.

Chapter 8

Conclusion

As our dependency on network communication is increasing rapidly and the networks are also evolving continuously, ensuring its security has become a crucial issue. Accordingly, different security measures should be applied to different layers of the network. Besides signature-based systems, anomaly-based systems should be employed to detect unknown attacks more effectively.

Therefore, in this study, we propose a network intrusion detection method based on Conditional Variational Laplace AutoEncoder (CVLAE) and Deep Neural Network (DNN). We condition VLAE on class labels so that the latent representation of samples of different class labels get separated in the latent space. We can also generate attack samples based on class labels. VLAE uses full-covariance Gaussian as posterior distribution, so it has higher expressive power than VAE. We name the enhanced model Conditional Variational Laplace AutoEncoder (CVLAE). Further, we also extend the model by adding attention mechanism to learn the feature representation more effectively and term the new method Conditional Variational Laplace Attention AutoEncoder (CVLAE). We use the generative models to balance the network datasets by increasing minority attack samples. We train a DNN classifier as an intrusion classifier on the balanced dataset and the DNN classifier performs better on minority attacks after being trained on the balanced dataset. We evaluate our proposed methods on NSL-KDD and KDD CUP 99 datasets and compare them with three conventional data augmentation intrusion detection methods. Experimental results confirm that CVLAE-DNN mostly exhibits substantially better performance for the minority attacks, and CVLAAE-DNN mostly exhibits overall better performance on all the types of attacks.

References

[1] UNB, "Nsl kdd dataset." https://www.unb.ca/cic/datasets/nsl.html, 2009. Accessed: August 5, 2022.

[2] S. J. Stolfo, "Kdd cup 99 dataset." http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999. Accessed: August 5, 2022.

[3] D. C. Le and N. Zincir-Heywood, "A frontier: Dependable, reliable and secure machine learning for network/system management," *Journal of Network and Systems Management*, vol. 28, no. 4, pp. 827–849, 2020.

[4] A. K. Nayak, A. Reimers, N. Feamster, and R. Clark, "Resonance: Dynamic access control for enterprise networks," in *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, pp. 11–18, ACM, 2009.

[5] "The capital one hack." https://edition.cnn.com/2019/07/29/business/capital-one-data-breach/index.html, 2019. Accessed: August 1, 2022.

[6] "Iot under fire: Kaspersky detects more than 100 million attacks on smart devices in h1 2019." https://www.kaspersky.com/about/press-releases/2019_iot-under-fire-kaspersky-detects-more-than-100-million-attacks-on-smart-devices-in-h1-2019, 2019. Accessed: August 1, 2022.

[7] "Solarwinds hack explained:everything you need to know." https://whatis.techtarget.com/feature/SolarWinds-hack-explained-Everything-you-need-to-know, 2021. Accessed: August 1, 2022.

[8] "Massive ddos attack disrupts belgium parliament." https://threatpost.com/ddos-disrupts-belgium/165911/, 2021. Accessed: August 1, 2022.

[9] U. government, "Cyber security breaches survey 2021." https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2021, 2019. Accessed: August 1, 2022.

[10] S. Kumar, "Survey of current network intrusion detection techniques," *Washington Univ. in St. Louis*, pp. 1–18, 2007.

[11] G. Kim, S. Lee, and S. Kim, "A novel hybrid intrusion detection method integrating anomaly detection with misuse detection," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1690–1700, 2014.

[12] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," *Computers & security*, vol. 21, no. 5, pp. 439–448, 2002.

[13] L. Koc, T. A. Mazzuchi, and S. Sarkani, "A network intrusion detection system based on a hidden na¨ıve bayes multiclass classifier," *Expert Systems with Applications*, vol. 39, no. 18, pp. 13492–13500, 2012.

[14] K. Choksi, B. Shah, and O. Kale, "Intrusion detection system using self organizing map: a surevey," *International Journal of Engineering Research and Applications*, vol. 4, no. 12, pp. 11–16, 2014.

[15] T. Mehmood and H. B. M. Rais, "Svm for network anomaly detection using aco feature subset," in *Proceedings of the International symposium on mathematical sciences and computing research*, pp. 121–126, IEEE, 2015.

[16] R. Sen, M. Chattopadhyay, and N. Sen, "An efficient approach to develop an intrusion detection system based on multi layer backpropagation neural network algorithm: Ids using bpnn algorithm," in *Proceedings of the 2015 ACM SIGMIS Conference on Computers and People Research*, pp. 105–108, ACM, 2015.

[17] S. Huda, S. Miah, J. Yearwood, S. Alyahya, H. Al-Dossari, and R. Doss, "A malicious threat detection model for cloud assisted internet of things (cot) based industrial control system (ics) networks using deep belief network," *Journal of Parallel and Distributed Computing*, vol. 120, pp. 23–31, 2018.

[18] T. Aldwairi, D. Perera, and M. A. Novotny, "An evaluation of the performance of restricted boltzmann machines as a model for anomaly network intrusion detection," *Computer Networks*, vol. 144, pp. 111–119, 2018.

[19] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Proceedings of the 2016 International Conference on Wireless Networks and Mobile Communications*, pp. 258–263, IEEE, 2016.

[20] Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based ddos detection system in software-defined networking (sdn)," *arXiv preprint arXiv:1611.07400*, 2016.

[21] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[22] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[23] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *Proceedings of the International Joint Conference on Neural Networks*, pp. 1322–1328, IEEE, 2008.

[24] Y. Yang, K. Zheng, C. Wu, and Y. Yang, "Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network," *Sensors*, vol. 19, no. 11, p. 2528, 2019.

[25] L. Vu, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang, E. Dutkiewicz, *et al.*, "Learning latent distribution for distinguishing network traffic in intrusion detection system," in *Proceedings of the International Conference on Communications*, pp. 1–6, IEEE, 2019.

[26] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013.

[27] Y. Park, C. Kim, and G. Kim, "Variational laplace autoencoders," in *Proceedings of the International Conference on Machine Learning*, pp. 5032–5041, PMLR, 2019.

[28] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[29] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

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

[31] K. Gregor, I. Danihelka, A. Graves, D. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," in *Proceedings of the International Conference on Machine Learning*, pp. 1462–1471, PMLR, 2015.

[32] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing, "Toward controlled generation of text," in *Proceedings of the International Conference on Machine Learning*, pp. 1587–1596, PMLR, 2017.

[33] M. Blaauw and J. Bonada, "Modeling and transforming speech using variational autoencoders," in *Proceedings of the 2016 Interspeech*, pp. 1770–1774, ISCA, 2016.

[34] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Machine learning*, vol. 37, no. 2, pp. 183–233, 1999.

[35] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improved variational inference with inverse autoregressive flow," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[36] C. Cremer, X. Li, and D. Duvenaud, "Inference suboptimality in variational autoencoders," in *Proceedings of the 35th International Conference on Machine Learning*, pp. 1078–1086, PMLR, 2018.

[37] J. D. Bodapati, N. S. Shaik, and V. Naralasetti, "Composite deep neural network with gated-attention mechanism for diabetic retinopathy severity classification," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 10, pp. 9825–9839, 2021.

[38] T. R. Toha, N. A. Al-Nabhan, S. I. Salim, M. Rahaman, U. Kamal, and A. A. Al Islam, "Lc-net: Localized counting network for extremely dense crowds," *Applied Soft Computing*, vol. 123, p. 108930, 2022.

[39] A. A. Aburomman and M. Bin Ibne Reaz, "Survey of learning methods in intrusion detection systems," in *Proceedings of the 2016 International Conference on Advances in Electrical, Electronic and Systems Engineering (ICAEES)*, pp. 362–365, IEEE, 2016.

[40] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR)*, vol. 9, pp. 381–386, 2020.

[41] 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, no. 2, pp. 493–501, 2019.

[42] C. Wagner, J. François, T. Engel, *et al.*, "Machine learning approach for ip-flow record anomaly detection," in *Proceedings of the International Conference on Research in Networking*, pp. 28–39, Springer, 2011.

[43] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE transactions on computers*, vol. 65, no. 10, pp. 2986–2998, 2016.

[44] S. Thaseen and C. A. Kumar, "An analysis of supervised tree based classifiers for intrusion detection system," in *Proceedings of the International Conference on Pattern Recognition, Informatics and Medical Engineering*, pp. 294–299, IEEE, 2013.

[45] L. Kuang and M. Zulkernine, "Dnids: a dependable network intrusion detection system using the csi-knn algorithm," Master's thesis, 2007.

[46] A. D. Jadhav and V. Pellakuri, "Intrusion detection system using machine learning techniques for increasing accuracy and distributed & parallel approach for increasing efficiency," in *Proceedings of the 5th International Conference On Computing, Communication, Control And Automation*, pp. 1–4, IEEE, 2019.

[47] I. Manzoor, N. Kumar, *et al.*, "A feature reduced intrusion detection system using ann classifier," *Expert Systems with Applications*, vol. 88, pp. 249–257, 2017.

[48] A. Amoordon, V. Deniau, A. Fleury, and C. Gransart, "A single supervised learning model to detect fake access points, frequency sweeping jamming and deauthentication attacks in ieee 802.11 networks," *Machine Learning with Applications*, vol. 10, p. 100389, 2022.

[49] S. Ouiazzane, M. Addou, and F. Barramou, "A multiagent and machine learning based denial of service intrusion detection system for drone networks," in *Geospatial Intelligence*, pp. 51–65, Springer, 2022.

[50] M. H. Bhuyan, D. Bhattacharyya, and J. K. Kalita, "An effective unsupervised network anomaly detection method," in *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, pp. 533–539, IEEE, 2012.

[51] F. E. Heba, A. Darwish, A. E. Hassanien, and A. Abraham, "Principle components analysis and support vector machine based intrusion detection system," in *Proceedings of the 10th International Conference on Intelligent Systems Design and Applications*, pp. 363–367, IEEE, 2010.

[52] R. Braga, E. de Souza Mota, and A. Passito, "Lightweight ddos flooding attack detection using nox/openflow," in *Proceedings of the IEEE Local Computer Network Conference*, vol. 10, pp. 408–415, IEEE, 2010.

[53] Y. Wang, G. Sun, X. Cao, and J. Yang, "An intrusion detection system for the internet of things based on the ensemble of unsupervised techniques," *Wireless Communications and Mobile Computing*, vol. 2022, 2022.

[54] J. Haweliya and B. Nigam, "Network intrusion detection using semi supervised support vector machine," *International Journal of Computer Applications*, vol. 85, no. 9, 2014.

[55] C. Chen, Y. Gong, and Y. Tian, "Semi-supervised learning methods for network intrusion detection," in *Proceedings of the International Conference on Systems, Man and Cybernetics*, pp. 2603–2608, IEEE, 2008.

[56] L. Deng, D. Yu, *et al.*, "Deep learning: methods and applications," *Foundations and Trends® in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[57] T. Ma, F. Wang, J. Cheng, Y. Yu, and X. Chen, "A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks," *Sensors*, vol. 16, no. 10, p. 1701, 2016.

[58] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pp. 21–26, ICST, 2016.

[59] L. Nicholas, S. Y. Ooi, Y. H. Pang, S. O. Hwang, and S.-Y. Tan, "Study of long short-term memory in flow-based network intrusion detection system," *Journal of Intelligent & Fuzzy Systems*, vol. 35, no. 6, pp. 5947–5957, 2018.

[60] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, "A novel two-stage deep learning model for efficient network intrusion detection," *IEEE Access*, vol. 7, pp. 30373–30385, 2019.

[61] ACCS, "Unsw nb15 dataset." https://www.unsw.adfa.edu.au/australian-centre-for-cyber-security/cybersecurity/ADFA-NB15-Datasets/, 2015. Accessed: August 5, 2022.

[62] Z. Li, A. L. G. Rios, G. Xu, and L. Trajković, "Machine learning techniques for classifying network anomalies and intrusions," in *Proceedings of the international symposium on circuits and systems*, pp. 1–5, IEEE, 2019.

[63] I. Dataport, "Bgp dataset." https://dx.doi.org/10.21227/98aa-sh66, 2020. Accessed: August 5, 2022.

[64] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[65] S. N. Mighan and M. Kahani, "A novel scalable intrusion detection system based on deep learning," *International Journal of Information Security*, vol. 20, no. 3, pp. 387–403, 2021.

[66] X. Zhou, Y. Hu, W. Liang, J. Ma, and Q. Jin, "Variational lstm enhanced anomaly detection for industrial big data," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3469–3477, 2020.

[67] M. A. Khan, "Hcrnnids: hybrid convolutional recurrent neural network-based network intrusion detection system," *Processes*, vol. 9, no. 5, p. 834, 2021.

[68] UNB, "Cse-cic-ids2018 dataset." https://www.unb.ca/cic/datasets/ids-2018.html, 2018. Accessed: August 5, 2022.

[69] K. Saurabh, S. Sood, P. A. Kumar, U. Singh, R. Vyas, O. Vyas, and R. Khondoker, "Lbdmids: Lstm based deep learning model for intrusion detection systems for iot networks," in *Proceedings of the 2022 IEEE World AI IoT Congress (AIIoT)*, pp. 753–759, IEEE, 2022.

[70] A. S. Alqahtani, "Fso-lstm ids: hybrid optimized and ensembled deep-learning network-based intrusion detection system for smart networks," *The Journal of Supercomputing*, vol. 78, no. 7, pp. 9438–9455, 2022.

[71] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin, "Variational autoencoder for deep learning of images, labels and captions," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[72] W. Xu, H. Sun, C. Deng, and Y. Tan, "Variational autoencoder for semi-supervised text classification," in *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, p. 3358–3364, AAAI, 2017.

[73] X. Li and J. She, "Collaborative variational autoencoder for recommender systems," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 305–314, 2017.

[74] J. Pereira and M. Silveira, "Unsupervised anomaly detection in energy time series data using variational recurrent autoencoders with attention," in *Proceedings of the 17th IEEE International Conference on Machine learning and Applications*, pp. 1275–1282, IEEE, 2018.

[75] V. Røsjø, "Variational autoencoders with mixture density networks for sequence prediction in algorithmic composition-a musical world model," Master's thesis, 2018.

[76] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, pp. 1–18, 2015.

[77] Y. Kawachi, Y. Koizumi, and N. Harada, "Complementary set variational autoencoder for supervised anomaly detection," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2366–2370, IEEE, 2018.

[78] J. Sun, X. Wang, N. Xiong, and J. Shao, "Learning sparse representation with variational auto-encoder for anomaly detection," *IEEE Access*, vol. 6, pp. 33353–33361, 2018.

[79] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot," *Sensors*, vol. 17, no. 9, p. 1967, 2017.

[80] X. Xu, J. Li, Y. Yang, and F. Shen, "Toward effective intrusion detection using log-cosh conditional variational autoencoder," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6187–6196, 2020.

[81] H. Neuschmied, M. Winter, K. Hofer-Schmitz, B. Stojanovic, and U. Kleb, "Two stage anomaly detection for network intrusion detection," in *Proceedings of the International Conference on Information Systems Security and Privacy*, pp. 450–457, 2021.

[82] U. Sabeel, S. S. Heydari, K. Elgazzar, and K. El-Khatib, "Cvae-an: Atypical attack flow detection using incremental adversarial learning," in *Proceedings of the 2021 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2021.

[83] R. F. Lova, R. M. Fifaliana, and W. P. De Silva, "Intrusion detection toward feature reconstruction using huber conditional variational autoencoder," in *Proceedings of the 2022 International Conference on Information Networking (ICOIN)*, pp. 13–17, IEEE, 2022.

[84] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," *arXiv preprint arXiv:1901.02860*, 2019.

[85] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 2921–2929, IEEE, 2016.

[86] T. Yang, Y. Hu, Y. Li, W. Hu, and Q. Pan, "A standardized ics network data processing flow with generative model in anomaly detection," *IEEE Access*, vol. 8, pp. 4255–4264, 2019.

[87] C. Tang, N. Luktarhan, and Y. Zhao, "Saae-dnn: Deep learning method on intrusion detection," *Symmetry*, vol. 12, no. 10, p. 1695, 2020.

[88] F. Laghrissi, S. Douzi, K. Douzi, and B. Hssina, "Ids-attention: an efficient algorithm for intrusion detection systems using attention mechanism," *Journal of Big Data*, vol. 8, no. 1, pp. 1–21, 2021.

[89] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[90] "One-hot encoding." https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/, 2019. Accessed: August 30, 2022.

[91] T. Al-Shehari and R. A. Alsowail, "An insider data leakage detection using one-hot encoding, synthetic minority oversampling and machine learning techniques," *Entropy*, vol. 23, no. 10, p. 1258, 2021.

[92] L. Yu, R. Zhou, R. Chen, and K. K. Lai, "Missing data preprocessing in credit classification: One-hot encoding or imputation?," *Emerging Markets Finance and Trade*, vol. 58, no. 2, pp. 472–482, 2022.

[93] B. Gu and Y. Sung, "Enhanced reinforcement learning method combining one-hot encoding-based vectors for cnn-based alternative high-level decisions," *Applied Sciences*, vol. 11, no. 3, p. 1291, 2021.

[94] A. Y. Hussein, P. Falcarin, and A. T. Sadiq, "Enhancement performance of random forest algorithm via one hot encoding for iot ids," *Periodicals of Engineering and Natural Sciences (PEN)*, vol. 9, no. 3, pp. 579–591, 2021.

[95] L. A. A. Shalabi, Z. Shaaban, and B. Kasasbeh, "Data mining: A preprocessing engine," *Journal of Computer Science*, vol. 2, pp. 735–739, 2006.

[96] S. A. D. Prasetyowati, M. Ismail, E. N. Budisusila, M. H. Purnomo, *et al.*, "Dataset feasibility analysis method based on enhanced adaptive lms method with min-max normalization and fuzzy intuitive sets," *International Journal on Electrical Engineering and Informatics*, vol. 14, no. 1, pp. 55–75, 2022.

[97] H.-J. Kim, J.-W. Baek, and K. Chung, "Associative knowledge graph using fuzzy clustering and min-max normalization in video contents," *IEEE Access*, vol. 9, pp. 74802–74816, 2021.

[98] H. Henderi, T. Wahyuningsih, and E. Rahwanto, "Comparison of min-max normalization and z-score normalization in the k-nearest neighbor (knn) algorithm to test the accuracy of types of breast cancer," *International Journal of Informatics and Information Systems*, vol. 4, no. 1, pp. 13–20, 2021.

[99] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[100] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *International Journal of Engineering Applied Sciences and Technology*, vol. 4, no. 12, pp. 310–316, 2017.

[101] MIT, "Darpe 98 dataset." https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset, 1998. Accessed: August 5, 2022.

[102] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[103] S. Zhang, Y. Yuan, Z. Yao, X. Wang, and Z. Lei, "Improvement of the performance of models for predicting coronary artery disease based on xgboost algorithm and feature processing technology," *Electronics*, vol. 11, no. 3, p. 315, 2022.

[104] J. Bopaiah and R. Kavuluru, "Precision/recall trade-off analysis in abnormal/normal heart sound classification," in *International Conference on big data analytics*, pp. 179–194, Springer, 2017.

[105] S. A. Alvarez, "An exact analytical relation among recall, precision, and classification accuracy in information retrieval," *Boston College, Boston, Technical Report BCCS-02-01*, pp. 1–22, 2002.

[106] A. Javadpour, P. Pinto, F. Ja'fari, and W. Zhang, "Dmaidps: a distributed multi-agent intrusion detection and prevention system for cloud iot environments," *Cluster Computing*, pp. 1–18, 2022.

[107] Z. Hu, H. Hasegawa, Y. Yamaguchi, and H. Shimada, "High-performance distributed nids cluster based on hybrid detection platform," in *Proceedings of the IEICE*, The Institute of Electronics, Information and Communication Engineers, 2021.

[108] Z. Ahmad, A. Shahid Khan, K. Nisar, I. Haider, R. Hassan, M. R. Haque, S. Tarmizi, and J. J. Rodrigues, "Anomaly detection using deep neural network for iot architecture," *Applied Sciences*, vol. 11, no. 15, p. 7050, 2021.

[109] H. Kim, S. Ahn, W. R. Ha, H. Kang, D. S. Kim, H. K. Kim, and Y. Paek, "Panop: Mimicry-resistant ann-based distributed nids for iot networks," *IEEE Access*, vol. 9, pp. 111853–111864, 2021.

[110] M. S. Habeeb and T. R. Babu, "Network intrusion detection system: A survey on artificial intelligence-based techniques," *Expert Systems*, p. e13066, 2022.

[111] J. Verma, A. Bhandari, and G. Singh, "inids: Swot analysis and tows inferences of state-of-the-art nids solutions for the development of intelligent network intrusion detection system," *Computer Communications*, 2022.

[112] N. M. Yungaicela-Naula, C. Vargas-Rosales, and J. A. Perez-Diaz, "Sdn-based architecture for transport and application layer ddos attack detection by using machine and deep learning," *IEEE Access*, vol. 9, pp. 108495–108512, 2021.

[113] A. O. Alzahrani and M. J. Alenazi, "Designing a network intrusion detection system based on machine learning for software defined networks," *Future Internet*, vol. 13, no. 5, p. 111, 2021.

[114] A. Khurshid, W. Zhou, M. Caesar, and P. B. Godfrey, "Veriflow: Verifying network-wide invariants in real time," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, pp. 49–54, Association for Computing Machinery, 2012.

[115] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2317–2346, 2015.

[116] V. Patil, C. Patil, and R. Awale, "Security challenges in software defined network and their solutions," in *Proceedings of the 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–5, IEEE, 2017.

[117] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "Beta-VAE: Learning basic visual concepts with a constrained variational framework," in *Proceedings of the International Conference on Learning Representations*, ICLR, 2017.

[118] A. Khan and A. Storkey, "Adversarial robustness of $\beta-$ vae through the lens of local geometry," *arXiv preprint arXiv:2208.03923*, 2022.

[119] Y. Xiang, J. L. Højvang, M. H. Rasmussen, and M. G. Christensen, "A deep representation learning speech enhancement method using β-vae," *arXiv preprint arXiv:2205.05581*, 2022.

[120] C. Yuan, M. Su, C. Ni, X. Liu, Y. Xu, and X. Cui, "Horizon auto-picking with quantitative uncertainty evaluation by using a modified vq-vae framework," *Journal of Geophysics and Engineering*, vol. 19, no. 4, pp. 788–806, 2022.

[121] T. Srikotr and K. Mano, "Vector quantization of speech spectrum based on the vq-vae embedding space learning by gan technique," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 105, no. 4, pp. 647–654, 2022.