International Journal of Safety and Security Engineering
Vol. 15, No. 6, June, 2025, pp. 1265-1278
Journal homepage: http://iieta.org/journals/ijsse

# Multi-Class Anomaly Detection in Network Intrusion Detection Using Variational Autoencoder

Azhar F. Al-zubidi[1,2*], Alaa Kadhim Farhan[2]

[1] Computer Science Department, College of Sciences, AL Nahrain University, Baghdad 10072, Iraq
[2] Computer Sciences Department, University of Technology, Baghdad 10066, Iraq

Corresponding Author Email: cs.21.07@grad.uotechnology.edu.iq

https://doi.org/10.18280/ijsse.150616

**Received:** 5 March 2025
**Revised:** 20 April 2025
**Accepted:** 25 May 2025
**Available online:** 30 June 2025

***Keywords:***
*network intrusion detection, semi-supervised learning, variational autoencoder, class-specific detectors, multi-class anomaly detection*

## ABSTRACT

The advances in digital communication and network systems mandate the vital role of efficient network intrusion detection in identifying cyber threats. The inability to differentiate a potentially threatening action from a normal one becomes more complex due to the massive and highly diverse amount of network traffic. Intending to improve detection in multi-class anomaly detection tasks, this paper introduces a novel enhanced Multi Class Semi-Supervised Variational Autoencoder (MCSS-VAE) complemented with class-specific detectors derived by the Negative Selection Algorithm (NSA) and further optimized by the Clonal Selection Algorithm (CSA). The scoring mechanism consists of reconstruction loss, Kullback-Leibler (KL), with two other loss function to manage the multi class loss of attack, the divergence loss, and anomaly detection loss, which have been proven to improve anomaly detection performance. Tests were carried out on the NSL-KDD dataset. Results showed 99.43% accuracy with recall always higher than the baseline model, especially in semi-supervised settings that work with limited labeled data. The model significantly enhances the labeled and unlabeled detection accuracy, while reducing the false positives and false negatives. The developed model can be scaled up for real-world intrusion detection systems.

## 1. INTRODUCTION

System have emerged as significant challenge that organizations face today [1]. Compromises in modern networks are threatened by calculated assaults such as denial of service (DoS) attack, intrusion, and data leakage [2]. This can lead to a disruption of service, loss of data and ultimately lead to a monetary loss. Thus, identification of these attacks and measures against them are of prime importance for the protection of network systems. Conventional measures used in networks include firewalls and IDS that chiefly use signature and rule that define suspicious activities [3]. Despite these methods' efficiency against recognizable threats, they fail to identify new or different types of attacks, known as zero-day attacks [4]. Also, there is a massive and highly diverse amount of network traffic, which creates conditions for producing a highly heterogeneous data space, often resulting in an inability to differentiate the genuine activity from potentially threatening actions [5]. This complexity proves the usefulness of the new recent techniques comprising of adaptive mechanism for responding to new kinds of attacks and the ability of detecting anomalies.

Introduction of anomaly detection in network traffic rEq.uires many approaches due to several factors; First, the data of network traffic is always skewed, when normal traffic comprises the large majority of the dataset while the attacks are sparse [6]. This lack of balance hinders standard machine learning model from being able to learn significant patterns regarding anomaly detection. Second, most of the anomaly detection systems center on the binary of normal and anomalous behavior, and the fact that network attacks come in multiple forms is not taken into account [7]. Nevertheless, it is significant to identify different kinds of attacks to improve measures that are taken during an incident. Also, one of the main challenges of anomaly detection is the unavailability of labeled data. The process of labeling network traffic especially in the detection of new or upcoming threats is quite tedious bearing in mind the strains of actually analyzing and dissecting the traffic itself [8]. Therefore, a significant number of datasets have a high volume of data that have not been labeled, and, therefore, cannot be effectively processed with fully supervised learning model [9, 10]. This has resulted in more attention being paid to semi-supervised learning schemes in which a model is trained using a combination of labeled and unlabeled data; whereby the abundance of unlabeled data can be used to enhance the detection of tumor tissues [11].

There are many limitations in the above solution like class imbalance, multi-class problem, and the labeled data is a scarce commodity [12]. Previous approaches for computing anomaly detection have been focus on binary anomaly detection but do not distinguish between various types of attacks (e.g., Dos, probe, U2R, R2L) (have good generalization when the context rEq.uires multi-class classification in order to distinguish between the different types of attacks in incidents. Moreover, the situation with the normal and anomalous traffic is not balanced, which only aggravates the question [13, 14].

In this context, this paper introduces anomaly detection framework to address these challenges that combining semi-supervised Variational Autoencoders (VAE) with class-specific detectors derived from the NSA and optimized by the CSA. Our work employs the VAE ability to map data into a latent space and estimate the probability density function of the network traffic while using class-specific detectors to enhance attack type detection and differentiation. The effectiveness of our approach is tested on the NSL-KDD dataset, through extensive experiments, it is demonstrated that the proposed model in this work significantly outperforms existing methods in both anomaly detection and multi-class classification, with limited labeled data. Section 2 provides a related work review of related anomaly detection using machine learning techniques. Section 3 talk about the methods used, section 4 display the proposed semi-supervised VAE model in detail, including its architecture and training procedure. Section 5 the experimental setup and results is presented, Section 6 introduce the discussion to the work. And finally, Section 7 contains the outlines directions, conclusion and future research.

## 2. RELATED WORK

Many machine learning models have been applied in anomaly-based [13]. Initial studies in this area, started some 20 years back [15]. Feature engineering was a crucial component of these methodologies, and different techniques were proposed for achieving top 10 ranked features from NSL-KDD dataset through information gain [16]. In addition, multivariate correlation analysis supported by dissimilarity measures was applied to enhance accuracy [17, 18]. It is widely accepted that no single model can effectively address all types of problems. Even if multiple models perform well for a particular issue, identifying the optimal model for varying data distributions or statistical mixtures remains challenging [19].

Ensemble techniques, which aggregate the predictive of several models to improve the predictive ability, have therefore been investigated in the context of NIDS [20, 21]. Recent progress in deep learning has driven efforts to develop NIDS that do not rely on manual feature engineering. Techniques, such as spectral clustering combined with deep neural networks, were applied to NIDS for sensor networks [22]. Additionally, recurrent neural networks (RNNs) [23], including those based on LSTM [24], were proposed, alongside applications of convolutional neural networks (CNNs) [25]. Some approaches integrate LSTM and CNN to capture spatiotemporal patterns as feature extraction [26], while others utilize Generative Adversarial Networks (GANs) to derive statistical features [27]. Attention mechanisms were employed to enhance feature learning by emphasizing key inputs in sEq.uential network flow data within bidirectional LSTM models [28]. Autoencoders (AEs) are particularly suitable for anomaly-based NIDS due to their ability to detect deviations from previously learned normal states based on reconstruction errors exceeding a certain threshold [29]. To address challenges, such as noise reduction [30], a robust AE was introduced [31]. In addition, a max correntropy AE was suggested to deliver outlier and noise-induced representations [32], where reconstruction errors of AE were utilized to train classifiers. A stacked AE has also been suggested to conduct feature learning from n-gram hypertext transmission protocol log data [33]. Other authors have suggested NIDS architectures consisting of AEs and deep belief networks for feature learning, followed by the use of classifiers. These include NIDS based on asymmetric AEs [26], stacked AEs [34], and sparse AEs [35]. Stochastic denoising AEs [36] were applied to various NIDS [37], while stacked contractive AEs [38, 39] were utilized in other systems. Additionally, deep belief networks were employed [40], incorporating classifiers in a second stage that utilize shallow learning algorithms such as random forests [41], SVMs [35, 42] and Softmax classifiers [35]. Self-taught learning-based NIDS were also suggested [43].

Whereas several studies have employed machine learning techniques in NIDS, none of them have taken into consideration the problem of restricted labeled training samples. A straightforward problem while training models with limited data is overfitting. One-shot and few-shot learning, in so far as human beings can learn from a few examples, have been demonstrated to be outstanding when it comes to model training on insufficient data and high-level performance in image classification [44]. These techniques have also been recently extended to network security, namely malware detection [45, 46], wherein image data is utilized for malware mapping [47]. One-shot learning employing Siamese networks has also recently been extended for NIDS [48] to learn pair similarities and not class-specific features. Challenges still exist, for instance, ensuring there are good training pairs which are well-balanced for every class combination when constructing the training set.

Semi-supervised learning is also a significant solution to overfitting due to the few labeled training examples. Variational Autoencoder-based models (VAE) were investigated for that reason [49]. An autoencoder-enhanced one-class SVM, a hybrid model, which was trained only on normal class instances was also proposed. In addition, semi-supervised deep learning models using stacked sparse autoencoders (SSAE) [48], as well as their extension into federated learning (FL) [49], were developed. These approaches merge unsupervised feature extraction with supervised classification algorithms, utilizing both labeled and unlabeled data to improve performance.

Despite significant advancements, the field of anomaly detection-based network intrusion detection, particularly in multi-class scenarios, continues to face considerable challenges [50]. Existing machine learning methods struggle with the complexity of diverse network behaviors, especially when confronted with highly imbalanced datasets. These imbalances often lead to biased models that prioritize normal traffic, overlooking minority classes and failing to accurately detect diverse attack types. Moreover, many models fail to properly employ the latent space, which serves for maximizing the data reconstruction and the classification of anomalies with regard to multiple categories [51, 52]. This limits a clear and distinct categorization of different types of attacks, which complicates the interpretation and precision of the models.

The last important issue is a fact that network data are often characterized by the high dimensionality. The detection accuracy of machine learning models is inclined to decline as the number of features rises. Therefore, it becomes critical to devise sound feature reduction approaches [53]. If these challenges are not handled, then their impact will affect the enhancement of multi-class anomaly detection in NID systems. To address these challenges, a novel hybrid framework is proposed, that integrates a semi-supervised learning with class-specific detectors optimized through the bio-inspired algorithms NSA and CSA to enhance detection accuracy, reduce false positives and false negative, and improve generalization with limited labeled data. By using variational autoencoders for unsupervised feature learning and a hybrid scoring mechanism (reconstruction loss, KL divergence, divergence loss, and anomaly detection loss), the aim of this model is to robustly differentiate between normal and diverse cyber threats in large-scale network traffic. The proposed approach is validated on the NSL-KDD dataset, with a focus on scalability and adaptability to real-word intrusion detection systems.

## 3. METHODS

With the rapid increase in complexity and sophistication of cyber-attacks, traditional IDS systems become somewhat irrelevant in the domain of cybersecurity. One of the critical methods becomes anomaly detection; it utilizes machine learning and statistics to detect deviations from normal behavior that may indicate possible attacks.

### 3.1 Variational autoencoders

VAE are a strong family of deep generative models that occupy the middle ground between graphical probabilistic models and plain vanilla autoencoders. While vanilla autoencoders mostly try to learn compact data compressions and reconstructions, VAEs specifically try to learn the probability distribution of the input data. It does so by marrying the architectural advantages of neural networks with those of variational inference, enabling VAEs not only to reconstruct inputs but to sample novel, reasonable data points that are distributionally close to the training set. This unique skill is the product of their design, which puts a disciplined and ongoing latent space, and therefore they are highly well-suited for application in other tasks than just simple dimensionality reduction, such as data generation, data imputation, and, significantly for our interest, anomaly detection [54].

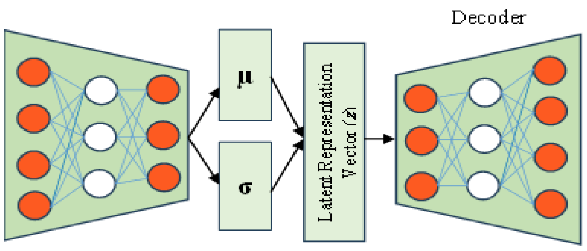


**Figure 1.** Variational autoencoder architecture

The architecture of a VAE, as presented as in Figure 1, is largely based on the optimal choice of its parts and their related parameters. The architecture is essentially motivated by achieving an excellent balance between compressing and reconstructing the data effectively [49].

The VAE adds a probabilistic twist to the traditional autoencoder as follows Eq. (1):

$$z = \mu + \sigma \odot \epsilon \tag{1}$$

where, μ and σ latent distribution mean and standard deviation, respectively; $\epsilon$ is a realization of a random noise vector, 0 and I are the mean and identity matrix, respectively.

The encoder maps the input data x to the parameters of the latent distribution ($\mu$ and $\sigma$). It's usually made of a few fully connected layers with non-linear activation functions (for example, ReLU). The encoder reduces the dimensionality of input data bit by bit until it becomes squeezed into lower-dimensional latent space.

The decoder has the symmetric structure with the encoder but works in reverse. The input to it is a latent variable z as input and reconstructs the original input data $\hat{x}$. The decoder has multilayers with increasing dimensionality step by step till it outputs the same dimensions as the input data. The last layer takes an appropriate activation function (for example, sigmoid for binary data) to ensure that the reconstructed data $\hat{x}$ is within the proper range.

The VAE is trained by optimizing two key objectives:

a) Reconstruction Loss:

This is a measure of how good the decoder is at reconstructing the input data from the latent representation. It trains the model to learn to produce good reconstructions.

In case of binary data, reconstruction loss is typically calculated with binary cross-entropy (BCE) Eq. (2):

$$\mathcal{L}_{\text{rec}} = -\sum_{i=1}^{n} [x_i \log(\hat{x}_i) + (1 - x_i)\log(1 - \hat{x}_i)] \tag{2}$$

For continuous data, MSE is commonly used Eq. (3):

$$\mathcal{L}_{\text{rec}} = \|x - \hat{x}\|^2 \tag{3}$$

b) KL Divergence Loss:

This regularizes the latent space by encouraging the learned distribution q(z|x) to match a prior distribution p(z), usually a standard normal distribution N (0, I). The KL divergence loss is calculated as Eq. (4):

$$\text{KL Divergence Loss} = -\frac{1}{2}\sum_{i=1}^{d} \left(1 + \log(\sigma_i^2) - \mu_i^2 - \sigma_i^2\right) \tag{4}$$

where, $d$ is the dimensionality of the latent space.

The total loss for the VAE is a sum of the reconstruction loss and the KL divergence loss Eq. (5):

$$\text{VAE Loss=Reconstruction Loss + KL Divergence Loss} \tag{5}$$

The VAE architecture combines the strengths of autoencoders and probabilistic modeling.

### 3.2 Negative selection algorithm (NSA)

Is inspiration-based computational model. works on producing binary strings that are usable to the matching of foreign strings but never arrive at matching up to self-strings [18]. If the binary string produced is compatible with a self-string, then the alignment is rejected in a never-ending process that simulates how the antibodies of the adaptive system interact with only harmful antigens and how T lymphocytes produced recognize only foreign cells [53]. NSA generates a set of detectors that are designed to match anomalous patterns while avoiding normal data. as in Algorithm (1) ,This makes

NSA a powerful tool for identifying outliers in complex datasets. However, the effectiveness of NSA depends heavily on the quality and diversity of the detectors, which still can be optimized using evolutionary algorithms. Algorithm steps would thus include training phase with self-data and detection phase.

```
NSA General Algorithm (1)
1. Initialization:
    • Define the self set S
    • Specify matching rule (e.g., Hamming distance threshold θ)
2. Detector Generation:
    • Randomly generate candidate detectors
    • Remove detectors which match any $s \in S$
    • Retain only valid detectors D
3. Anomaly Detection:
    • For a test sample xx check if matches any detector $d \in D$
    • If matched then classify x as anomaly.
```

### 3.3 Clonal selection algorithm (CSA)

CSA follows the lead of the immune system's. it chooses the detectors that perform best, copies them, and has them mutate so that they can better detect anomalies. The algorithm of clonal selection follows the lead of the biological immune response that is responsible for the stimulation of proliferation of the antibodies to detect a certain antigen. When the B lymphocytes become activated to be antibody-specific, they are cloned and the antibodies then undergo heavy genetic mutation so that they can be compatible with the antigen. Similarly, in the algorithm, it finds the best-fit binary strings and clones them to mutate so that the mutated strings are more compatible [55]. It is used as an adjunct algorithm to the NSA, as in Algorithm (2).

```
CSA General Algorithm (2)
1. Initialization:
    • Generate rando population A
2. Affinity Evaluation:
    • Calculate affinity for each antibody.
3. Cloning:
    • Select the best k antibodies according to affinity
    • Proportional cloning of selected antibody
4. Mutation:
    • Mutate cloned antibodies, so that mutation rate is inversely
      proportional to its affinity.
5. Selection:
    • Replace worst affinity antibodies from A with random new
      candidates.
6. Repeat the process until convergence or a predefined number
   of generations reached.
```

## 4. PROPOSAL OF MULTI-CLASS ANOMALY DETECTION USING SEMI-SUPERVISED

Based on the challenges outlined in the former section, this paper proposes a new paradigm for multi-class anomaly detection in NID. Using a VAE as an unsupervised encoder, we add a method utilizing the NSA and Clonal Selection Algorithm (CSA) in a semi-supervised learning setup. The proposed method, referred to as Multi-Class Semi-Supervised Variational Autoencoder (MCSS-VAE), is a novel approach that aims for higher efficiency. The NSL-KDD dataset is used as a benchmark for network intrusion detection. The dataset originally contains five different classes.

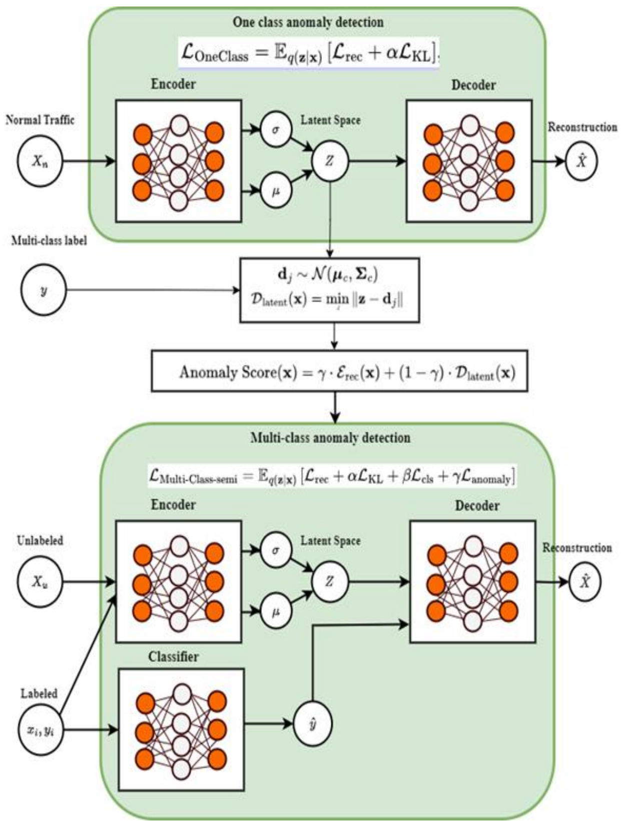

One class anomaly detection
$\mathcal{L}_{\text{OneClass}} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\mathcal{L}_{\text{rec}} + \alpha\mathcal{L}_{\text{KL}}\right]$
Encoder
Decoder
Normal Traffic
$X_n$
$\sigma$
$\mu$
Latent Space
$Z$
Reconstruction
$\hat{X}$
Multi-class label
$y$
$\mathbf{d}_j \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$
$\mathcal{D}_{\text{latent}}(\mathbf{x}) = \min_j \|\mathbf{z} - \mathbf{d}_j\|$
Anomaly Score$(\mathbf{x}) = \gamma \cdot \mathcal{E}_{\text{rec}}(\mathbf{x}) + (1-\gamma) \cdot \mathcal{D}_{\text{latent}}(\mathbf{x})$
Multi-class anomaly detection
$\mathcal{L}_{\text{Multi-Class-semi}} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\mathcal{L}_{\text{rec}} + \alpha\mathcal{L}_{\text{KL}} + \beta\mathcal{L}_{\text{cls}} + \gamma\mathcal{L}_{\text{anomaly}}\right]$
Encoder
Decoder
Unlabeled
$X_u$
Latent Space
$Z$
Reconstruction
$\hat{X}$
Classifier
Labeled
$x_i, y_i$
$\hat{y}$

**Figure 2.** The proposed approach

Our approach makes an attempt to improve the above abnormality detection and we show how NSA and CSA can help improve the modeling of normal behavior and keep improving the identification of different subtypes of anomalies. Meanwhile, MCSS-VAE tries to learn the low-dimensional space with semi-supervised learning setting. The Multi-Class Anomaly Detection methodology employs a VAE improved with class-specific detectors trained with the aid of NSA and optimized utilizing the CSA, Algorithm 1 represent the pseudocode for MCSS-VAE model. This methodology will be used to solve real problems in multi-class anomaly detection for a semi-supervised setting in which only a subset of the data samples is labeled.

This proposed approach integrates reconstruction-based anomaly detection with class-specific latent space modeling to provide robust and accurate detection across multiple network traffic classes. Figure 2 shows the structure of the proposed approach.

### 4.1 One-class anomaly detection VAE (M1)

One-Class Anomaly Detection VAE (M1) is learned with a small-sized normal network traffic dataset and can recognize anomalies as reconstruction mistakes. It consists of two main parts:

a) Encoder: Encoder maps input $x$ to a latent space representation z and produces outputs for mean $\mu(x)$ and log variance $\log(\sigma^2(x))$, themselves Gaussian modeled. The reparameterization trick Eq. (6) samples the latent variable $z$ as in Eq. (6):

$$z = \mu(x) + \sigma(x) \odot \epsilon,\ \epsilon \sim N(0, I) \qquad (6)$$

where, $\odot$ represents element-wise multiplication, and $\epsilon$ is a noise vector.

b) Decoder: recentralize the input $x$ from the latent variable $z$, learning to model the data distribution by minimizing the reconstruction error Eq. (7):

$$\hat{x} = g(z) \tag{7}$$

where, the decoder network is $g(\cdot)$. The reconstruction loss ($\mathcal{L}_{\text{rec}}$) and Kullback-Leibler ($\mathcal{L}_{\text{KL}}$) divergence are optimized as loss function combining the Eq. (8):

$$L = \mathbb{E}_{q(z|x)}[L_{\text{rec}}(x, \hat{x}) + \alpha L_{\text{KL}}(q(z|x) \parallel p(z))] \tag{8}$$

### 4.2 Class-specific detector generation

Our model introduces class-specific detectors, which enhance anomaly detection in multi-class scenarios. Each class has a set of detectors, positioned in the latent space to represent the typical distribution of normal data for that class. These detectors are generated and refined through NSA and CSA. For each class c, a set of detectors Dc={d1, d2…, dm} is generated Eq. (9). These detectors are sampled from a distribution centered around the mean of the latent representations for class c:

$$d_j \sim N(\mu_c, \Sigma_c),\ j = 1,2,\dots,m \tag{9}$$

where, $\mu_c$ is the mean of the latent representations for class $c$, and $\Sigma_c$ is the covariance matrix. Detectors are refined through NSA, which eliminates detectors close to anomalies, and CSA, which adjusts detectors to align with the normal data distribution Eq. (10):

$$d_j' = d_j - \eta \nabla_{d_j} L_{\text{refine}}(d_j) \tag{10}$$

### 4.3 Hybrid anomaly detection mechanism

We further improve detection accuracy with a hybrid anomaly detection mechanism, combining reconstruction errors and latent space distances to class-specific detectors:

a) Reconstruction Error: The reconstruction error, calculated using binary cross-entropy, where the difference between the input and its reconstruction was measured as in Eq. (11):

$$E_{\text{rec}}(x) = -\sum_{i=1}^{n} [x_i \log(\hat{x}_i) + (1 - x_i)\log(1 - \hat{x}_i)] \tag{11}$$

b) Latent Space Distance: This is the minimum distance between the latent representation z of the input and the nearest detector for the predicted class Eq. (12):

$$D_{\text{latent}}(x) = \min_j \|z - d_j\| \tag{12}$$

The final anomaly score combines these two components Eq. (13):

$$\text{Anomaly Score}(x) = \gamma E_{\text{rec}}(x) + (1 - \gamma) D_{\text{latent}}(x) \tag{13}$$

### 4.4 multi-class anomaly detection with semi-supervised VAE (M2)

The VAE forms the core of our proposed approach. It is a generative model designed to learn the underlying distribution of the data while simultaneously performing classification in a semi-supervised setting. By incorporating labeled and unlabeled data, the VAE learns a rich latent space that is both informative for reconstruction and useful for classification, making it a powerful tool for anomaly detection in complex multi-class scenarios. The structure of this model is similar to model M1 with a classification model with semi-supervised setting, as shown in Figure 2. This model consists of three primary components: the encoder, the decoder, and the classifier. The classifier is a neural network that predicts the class label y for the input $x$. It takes the input x and outputs a probability distribution over the class labels Eq. (14):

$$\mathbf{y}_{\text{pred}} = h(\mathbf{x}) \tag{14}$$

where, $h(\cdot)$ is the classifier network and $y_{pred}$ is the predicted class distribution. In the final stage, the M2 is re-trained in a semi-supervised setting using data from all classes. The class-specific detectors generated in Stage 2 are integrated into the training process to improve anomaly detection performance. The labeled and unlabeled data were combined, including both normal and different classes of attack traffic. Labeled data are used for classification, while both labeled and unlabeled data are used for reconstruction and latent space learning. M2 is initialized with the encoder, decoder, and a classifier. The previously trained encoder and decoder weights were fine-tuned, while the classifier is trained from scratch. M2 was trained using a hybrid loss function that includes:

a) Reconstruction Loss: The reconstruction loss is what gives the decoder the ability to reconstruct input data from the latent representation. For binary data, the reconstruction loss is usually the binary cross-entropy between the reconstruction x̂ and input $x$. Eq. (15):

$$\mathcal{L}_{\text{rec}} = -\sum_{i=1}^{n} [\mathrm{x}_i \log(\hat{\mathrm{x}}_i) + (1 - \mathrm{x}_i)\log(1 - \hat{\mathrm{x}}_i)] \tag{15}$$

where, $n$ is the number of features in $x$.

b) KL Divergence Loss: The KL divergence regularizes the latent space by encouraging the posterior distribution $q(z|x)$ to be close to a prior distribution p(z), usually a standard normal distribution. The KL divergence is defined as Eq. (16):

$$\mathcal{L}_{\text{KL}} = \frac{1}{2}\sum_{j=1}^{d} \left(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2\right) \tag{16}$$

where, $d$ is dimensionality of latent space, $\mu_j$ and $\sigma_j^2$ are mean and variance of the j-th latent dimension.

**c)** Classification Loss: The classification loss guides the learning of the classifier by minimizing the discrepancy between the true labels y and the predicted labels $y_{pred}$. For multi-class classification, this is typically the categorical cross-entropy Eq. (17):

$$\mathcal{L}_{\text{cls}} = -\sum_{k=1}^{K} y_k \log(y_{\text{pred},k}) \tag{17}$$

where, $K$=classes number, $y_k$=class $k$ true label, and $y_{pred,k}$ =class $k$ predicted probability.

c) Anomaly Detection Loss: Incorporates the distance to class-specific detectors in the latent space Eq. (18):

$$\mathcal{L}_{\text{anomaly}} = \gamma \cdot \mathcal{E}_{\text{rec}}(\mathbf{x}) + (1-\gamma) \cdot \mathcal{D}_{\text{latent}}(\mathbf{x}), \quad (18)$$

d) The Final Loss Function: The overall loss function during this phase is a hybrid of unsupervised and supervised components Eq. (19):

$$\mathcal{L}_{\text{Multi-classSemi}} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\big[\mathcal{L}_{\text{rec}} + \alpha\mathcal{L}_{\text{KL}} + \beta\mathcal{L}_{\text{cls}} + \gamma\mathcal{L}_{\text{anomaly}}\big], \quad (19)$$

where, $\mathcal{L}_{\text{cls}}$ is the classification loss, ensuring the correct prediction of class labels for labeled data; $\mathcal{L}_{\text{anomaly}}$ is the anomaly detection loss, which penalizes data points that deviate from the class-specific detectors in the latent space; $\beta$ and $\gamma$ are weighting parameters.

## 4.5 Training procedure

The training consists of two stages. In the first stage, the model (M1) is trained in an unsupervised manner using only normal traffic data. The objective is to obtain how a concise latent representation can capture the normal traffic pattern. The first part of the network called the encoder, transforms the input into the so-called latent space, and the second part, the decoder, reconstructs the same input from the latent space representation. As mentioned, after the training of the model, the encoder produces the representation of all available data in the latent space from which class-specific detectors are derived. In the second stage, M2 is trained with the labeled and the unlabeled data set with the help of class specific detectors refined through NSA and CSA. The detectors assist in improving the model's performance of detecting anomalies between classes. The labeled data are used for the classification tasks, the labeled and unlabeled data for the learning latent space representations. M2 is pre-trained using the weights of the encoder and decoder parts of M1 as the starting point for the weights update. The classification head is trained from scratch. In training, since a hybrid loss function is used, M2 is trained to be able to reconstruct data and classify anomalies into their appropriate classes. They are trained in mini-batches and the parameter are updated through the Adam optimizer with 0.001 learning rate. Use of early stopping as a means of preventing overfitting, or else training would be stopped when there was no improvement of the validation loss over the past 10 epochs, as in Algorithm (3).

**Algorithm 3: Multi-Class Semi-Supervised Variational Autoencoder for Anomaly Detection (MCSS-VAE)**

**Input:** Network traffic data (labeled and unlabeled), NSL-KDD dataset.
**Output:** Anomaly detection predictions and multi-class classification.

- **For data preprocessing**
- Clean missing values, and standardize features.
- Normalize the data to a fixed range (e.g., [0, 1]).
- Split the dataset into training (80%) and testing (20%) sets.
- **Construct input matrices**
- For each class, construct a matrix where rows represent samples and columns represent features.
- Include both normal and anomalous samples for labeled data.
- **Initialize MCSS-VAE model**
- Initialize the encoder, decoder, and classifier networks with random weights.
- Set hyperparameters: latent space dimension, learning rate, batch size, and loss weights (α, β, γ).
- **For unsupervised training of VAE (M1)**

a) Train VAE on normal data
- Train the VAE (encoder and decoder) using only normal traffic data.
- Optimize the loss function:

$$L_{\text{M1}} = \mathbb{E}_{q(z|x)}[L_{\text{rec}}(x,\hat{x}) + \alpha L_{\text{KL}}(q(z|x) \parallel p(z))]$$

Where:
$L_{\text{rec}}$: Reconstruction loss (e.g., binary cross-entropy).
$L_{\text{KL}}$ : KL divergence between the latent distribution and a standard normal prior.

b) Generate Latent Representations
- Use the trained encoder to map all data (normal and anomalous) into the latent space.
- Obtain latent vectors z for each sample.

c) Generate Class-Specific Detectors
- For each class c, generate a set of detectors $D_c = \{d_1, d_2, \dots, d_m\}$ in the latent space:

$$d_j \sim N(\mu_c, \Sigma_c),\ j = 1,2,\dots,m$$

where $\mu_c$ and $\Sigma_c$ are the mean and covariance of latent vectors for class
- Refine detectors using NSA and CSA.
- **Semi-Supervised Training of MCSS-VAE (M2)**
- Train MCSS-VAE with Labeled and Unlabeled Data
- Train the full MCSS-VAE model (encoder, decoder, and classifier) using both labeled and unlabeled data.
- Optimize the hybrid loss function:

$$\mathcal{L}_{\text{M2}} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\big[\mathcal{L}_{\text{rec}}(x,\hat{x}) + \alpha\mathcal{L}_{\text{KL}}(q(z|x) \parallel p(z)) + \beta\mathcal{L}_{\text{cls}}(y,\hat{y}) + \gamma\mathcal{L}_{\text{anomaly}}(x)\big]$$

Where:
$\mathcal{L}_{\text{rec}}$ : Classification loss (e.g., categorical cross-entropy).
$\mathcal{L}_{\text{anomaly}}$ : Anomaly detection loss combining reconstruction error and latent space distance:

$$\mathcal{L}_{\text{anomaly}} = \gamma \cdot \mathcal{E}_{\text{rec}}(x) + (1-\gamma) \cdot \mathcal{D}_{\text{latent}}(x),$$

$\mathcal{E}_{\text{rec}}(x)$: Reconstruction error.
$\mathcal{D}_{\text{latent}}(x)$: Minimum distance between z and the nearest class-specific detector.
- **Fine-tune classifier**
- Fine-tune the classifier using labeled data to improve multi-class classification accuracy.
- **Detect Anomalies**
- For each test sample, compute the anomaly score: Anomaly Score$(x) = \gamma \cdot \mathcal{E}_{\text{rec}}(x) + (1-\gamma) \cdot \mathcal{D}_{\text{latent}}(x)$,
- Classify samples as normal or anomalous based on a predefined threshold.
- **Evaluate Model Performance**
- Test the model by measures metrics.

## 5. EXPERIMENTAL RESULTS

This section contributes a critical assessment of the proposed MCSS-VAE, aimed at network intrusion detection. Here, the roles of class-specific detectors and the hybrid abnormality scoring system will be discussed. This paper sought to use experiments which were done in the simulator using the NSL-KDD data set, which is considered to be among the best for experiments in the network intrusion detection systems [56].

## 5.1 Dataset

The experiments are run on the commonly used NSL-KDD dataset that contains five various classes: a normal traffic class and four attack types that are DoS, U2R, R2L, and Probe. To study the performance of the above suggested MCSS-VAE model under various situations. We tested for different percentages of labeled data of 20 %, 40%, 60%, 80% and 100%. Before the model training process, the dataset was normalized where the features were scaled using the Min-Max scaler in order to range from 0 and 1. No further data pre-processing was done in order to reduce class imbalance because the distribution of the samples was maintained as is. Such decision enables better assessment of how the model performs when learning from imbalanced data; a problem typical in NID tasks ("Network Security, Information Security, Cyber Security," n.d.).

5.1.1 Data preprocessing

In this section, we describe our proposed model: MSS-SAE based on multi-class Semi-Supervised of Sparse Autoencoder anomaly detection

a) Data Preprocessing

In this part we delve into preprocessing steps that are crucial for enhancing the performance and stability of the AE model when dealing with the NSL-KDD datasets, including (Split the Dataset, feature Encoding, One-Hot Encoding, Handling Test Set Differences, Column Reordering, and Final Encoding for Test Set)

b) Split the Dataset:

Is the process of splitting the data into training and validation sets. As 70% and 30% for training and validation. The training set will be used to train the model, and the validation set will test its performance.

c) Feature Encoding:

Convert categorical features into numerical representations. For example, imagine if the 'protocol_type' feature has categories like 'tcp', 'udp', and 'icmp'. We show these as numerical values (e.g., 0, 1, 2).

d) One-Hot Encoding:

Apply one-hot encoding to categorical features. Like for the 'service' feature, we create separate binary columns for each service type ('http', 'ftp', etc.). A connection with 'http' service will have '1' in the 'http' column and '0' in others.

e) Handling Test Set Differences:

Ensure consistency between training and validation sets. If the test set has new services not seen in the training set, handle them appropriately (e.g., map them to a common category or create a column).

f) Final Encoding for Test Set:

Apply the same one-hot encoding to the test set, to consistency ensures fair evaluation of the model performance. Such as when evaluating the model on unseen data (test set), use the same encoding scheme as in training and validation.

## 5.2 Experimental setup

The experiments were conducted using Python and TensorFlow on the Google Colab platform, utilizing an NVIDIA Tesla T4 GPU to accelerate training and inference. Both the One-Class VAE (M1) and the Semi-Supervised VAE (M2) share a common architecture for the encoder and decoder, with the primary distinction being the inclusion of a classification head in M2, enabling it to handle multi-class classification in a semi-supervised framework.

The encoder for both models processed a 122-dimensional input feature vector, compressing it into a 15-dimensional latent space. This encoder consisted of three dense layers with ReLU activations, where the first layer contained 256 units, followed by layers with 128 and 64 units, respectively. With two output layers mean and log variance. z was then sampled using the reparameterization trick. The decoder mirrored the encoder's structure, reconstructing the 15-dimensional latent vector z back into the original 122-dimensional input space. 64, 128, and 256 units dense layers was employed, respectively, each activated using ReLU. The output layer utilized a sigmoid activation function to generate the final reconstruction of the input feature vector.

The Semi-Supervised VAE (M2) incorporated an additional classification model designed to leverage both labeled and unlabeled data. This classifier enabled the model's multi-class classification capabilities within the semi-supervised framework. The classifier was composed of three hidden layers with ReLU activations, containing 256, 128, and 64 units, respectively.

A Softmax output layer with five units was used to classify the input data into the following categories: Normal, DoS, U2R, R2L, and Probe. Both models used Adam optimizer and 0.001 for learning rate. Training was performed for a maximum of 100 epochs, and early stopping was used in case of no validation loss improvement over the past 10 epochs. The batch size was held constant at 128 for all experiments. To retain the best-performing model, model checkpointing was used, keeping the version with the minimum validation loss.

To evaluate our proposed MCSS-VAE, in semi-supervised scenarios, different ratios of labeled data (10%, 20%, and 30% up to 100%) were employed. The rest of the data were regarded as unlabeled ones, meaning both labeled and unlabeled data were used for training. This setup was useful to perform a final integration and analyze the model's potential and capability to generalize in cases wherein the availability of labeled data is limited. The purpose of these experiments was to assess the performance of developed multi-class anomaly detection model as well as to investigate how the proposed model could be feasibly implemented in large scale practical applications. Accuracy, precision, recall, F1 measure, FPR and FNR were used to assess the performance of model. These metrics gave a complete evaluation of the level that the model can accurately diagnose the anomalies and the type of network traffic that is associated with the anomalies. Particular attention was paid to assessing the model in terms of how it fares when using (as an input) the class-specific detectors and the hybrid scoring scheme

## 5.3 Results

In this section, the experimental results of our proposed MCSS-VAE for multi-class anomaly detection are presented.

In Figure 3 the numerical latent space of the Semi-Supervised VAE shows that the normal class is tightly clustered, which indicated that the model has learned and represented normal network behavior. Each type of anomaly occupies distinct regions within the latent space, showing the model's ability in accurate identification and separation of different attack types. This would indeed make such class-specific detectors ensure that refinement in the latent space by anomalies will have clearer distinctions from the normal traffic. In such a way, a strong detection mechanism ensures that even small overlaps between normal and anomalous clusters do not affect the overall performance between various classes of attacks like Probe, U2L, and U2r.

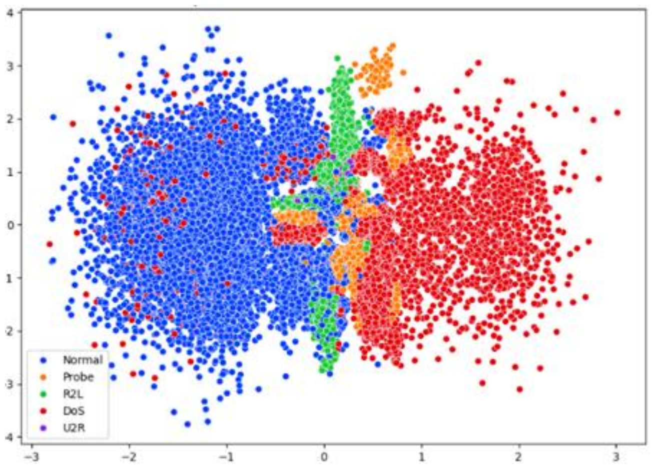
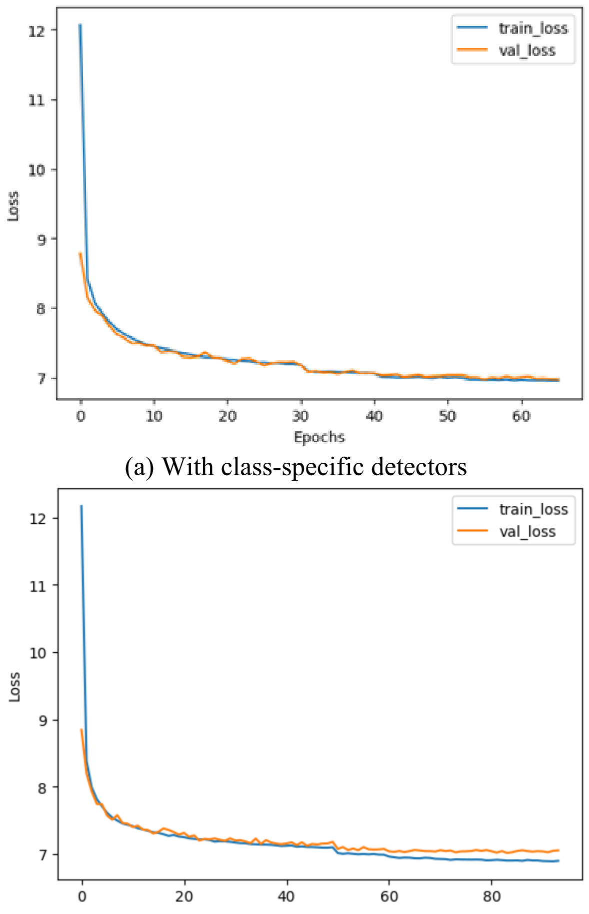


**Figure 3.** Latent space visualization with 5 class anomaly detection

The loss curves for training and validation are shows that models with class-specific detectors have a constant decrease without overfitting as in Figure 4 thus, there is good generalization over unseen data. On the other hand, in models without detectors, after 45 epochs, the training loss keeps on decreasing but the validation loss increases, indicating overfitting.



(a) With class-specific detectors

(b) Without class-specific detectors

**Figure 4.** Training vs. validation MSE loss

5.3.1 Statistical validation for model

The accuracy of the model achieves 99.43%, the importance of statistical rates is acknowledged to ensure the reliability of these results. To describe this, a comprehensive statistical analysis represents in Figure 5 by plotting distribution of anomaly scores for all classes (normal vs. attack types). That represent the separation between normal and attack types.

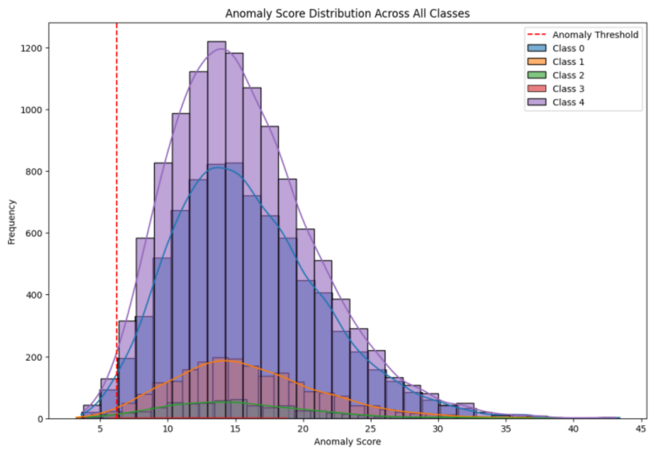


**Figure 5.** Anomaly score across all classes

In addition to the Figure 6 represented the reconstruction error for normal data distribution and derived on optimal anomaly threshold Eq.ual to 0.0323. this ensures robustness against random sampling variability.

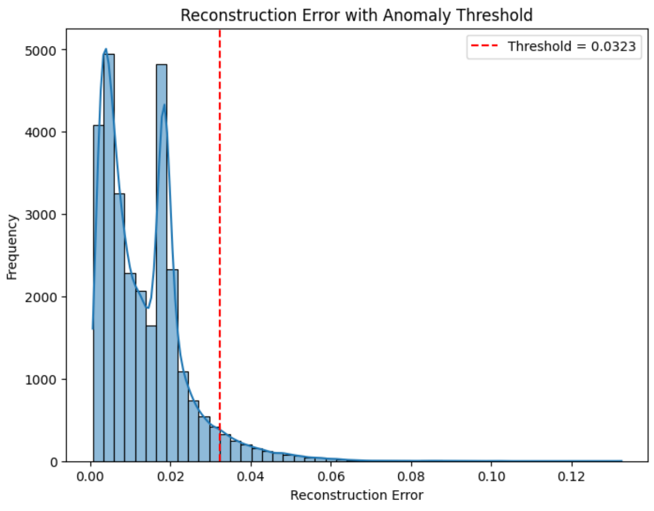
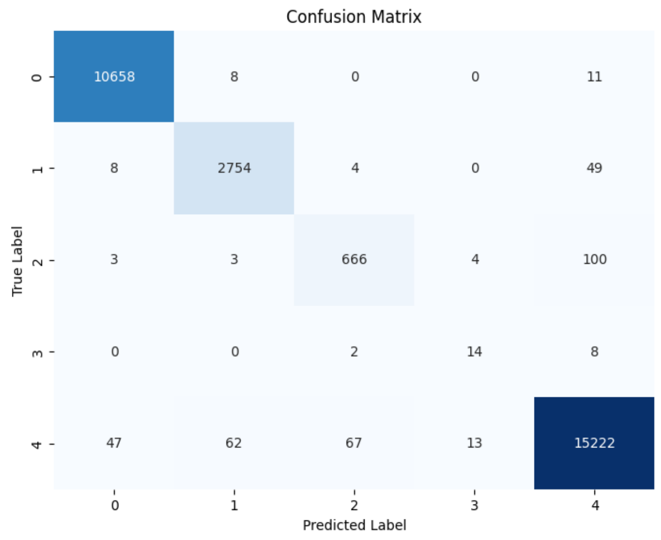


**Figure 6.** Reconstruction error with anomaly threshold

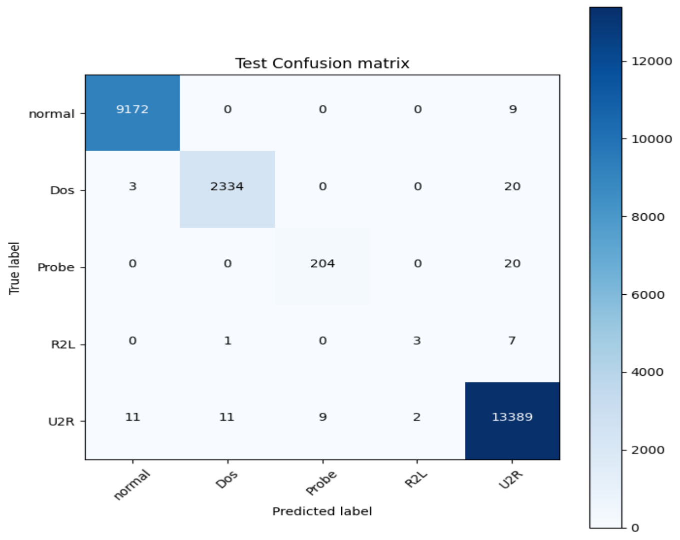
In Figure 7 the confusion matrix for train and test set is highlighted per-class with minimal difference between attack types.

In Figure 8 the AUC use to score and distinguish between attack types in the test set with greater than 0.99 for class 0,1, and 4 and lower performance for class 3 (AUC=0.636). The model was able to correctly discriminate between the majority high-risk classes (Classes 0 [normal], 1 [DoS], and 4 [Probe]) with AUC scores above 0.99, demonstrating perfect scoring efficiency. The NSL-KDD dataset and its behavioral patterns likely contributed to this result, such as the Probe class for port-scanning signatures boosted by NSA/CSA controllers. However, Class 3 (U2R or R2L attacks) struggles with significantly lower AUC scores of $0.636 \pm 0.082$, 95% CI. The challenge of detecting low, stealthy attacks like these that mimic normal traffic and severely imbalanced class representations rests somewhere beneath 0.1% is their downfall. This gap further demonstrates overclass score distributions alongside Class 3 and normal traffic in Figure 8 portending latent ambiguous space. Although bootstrapping confirms classism ranging around 0.999 ± 0.001, this results wider confidence intervals of Class 3 claiming data paucity uncertainty. Further work can look into solving this issue through domain specific heuristics, or hierarchical frameworks to better identify defining rare class attacks.

Table 1 represents the results of performance without class-specific detectors and Table 2 represents the results of performance with class-specific detectors and hybrid scoring.



(a) Train data confusion matrix



(b) Test data confusion matrix

**Figure 7.** Train and test confusion matrix

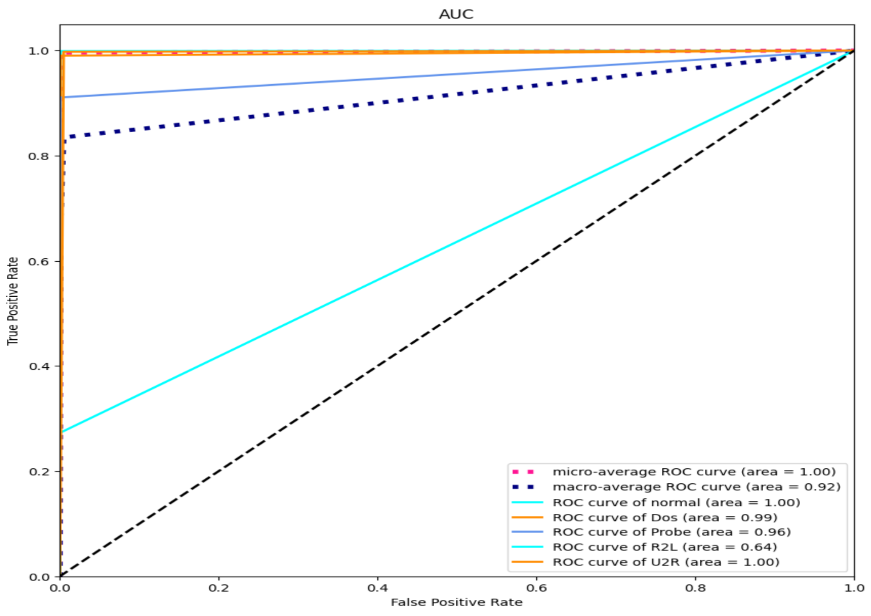


**Figure 8.** Attack types test set

**Table 1.** Performance without class-specific detectors and hybrid scoring

| Label Rate | Accuracy | Precision | Recall | F1 Score | FPR | FNR |
|---|---|---|---|---|---|---|
| 0.1 | 0.9887 | 0.9756 | 0.8222 | 0.8579 | 0.0037 | 0.1778 |
| 0.2 | 0.9906 | 0.9094 | 0.8854 | 0.8964 | 0.0029 | 0.1146 |
| 0.3 | 0.9912 | 0.8830 | 0.8481 | 0.8604 | 0.0026 | 0.1519 |
| 0.4 | 0.9921 | 0.9112 | 0.8695 | 0.8859 | 0.0024 | 0.1305 |
| 0.5 | 0.9926 | 0.9218 | 0.9124 | 0.9170 | 0.0024 | 0.0876 |
| 0.6 | 0.9927 | 0.9354 | 0.8762 | 0.9007 | 0.0024 | 0.1238 |
| 0.7 | 0.9935 | 0.9257 | 0.8767 | 0.8980 | 0.0022 | 0.1233 |
| 0.8 | 0.9936 | 0.9366 | 0.8944 | 0.9121 | 0.0020 | 0.1056 |
| 0.9 | 0.9939 | 0.9263 | 0.8941 | 0.9082 | 0.0019 | 0.1059 |
| 1.0 | 0.9936 | 0.9285 | 0.9064 | 0.9163 | 0.0020 | 0.0936 |

**Table 2.** Performance with class-specific detectors and hybrid scoring

| Label Rate | Accuracy | Precision | Recall | F1 Score | FPR | FNR |
|---|---|---|---|---|---|---|
| 0.1 | 0.9862 | 0.8454 | 0.8957 | 0.8677 | 0.0037 | 0.1043 |
| 0.2 | 0.9898 | 0.8707 | 0.9209 | 0.8934 | 0.0027 | 0.0791 |
| 0.3 | 0.9912 | 0.9022 | 0.9205 | 0.9108 | 0.0025 | 0.0795 |
| 0.4 | 0.9921 | 0.9062 | 0.9115 | 0.9087 | 0.0025 | 0.0885 |
| 0.5 | 0.9927 | 0.9081 | 0.9153 | 0.9116 | 0.0023 | 0.0847 |
| 0.6 | 0.9933 | 0.9108 | 0.9056 | 0.9081 | 0.0021 | 0.0944 |
| 0.7 | 0.9936 | 0.9263 | 0.9087 | 0.9163 | 0.0019 | 0.0913 |
| 0.8 | 0.9935 | 0.9264 | 0.9080 | 0.9160 | 0.0020 | 0.0920 |
| 0.9 | 0.9937 | 0.9336 | 0.8962 | 0.9114 | 0.0019 | 0.1038 |
| 1.0 | 0.9943 | 0.9540 | 0.9066 | 0.9265 | 0.0018 | 0.0934 |

Label Rate was from 0.1 to 1.0 with increments of 0.1. "Accuracy" is the proportion of the number of true results (true positives and true negatives) to the number of total cases tested Eq. (20) [40]:

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative} \quad (20)$$

Precision is computed as Eq. (21) [41]:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (21)$$

For Recall as Eq. (22):

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (22)$$

F1 Measure is as Eq. (23) [35]:

$$F1\ score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (23)$$

Lastly, there is both the FPR and the FNR; FPR depicts the ratio of the false positive cases over all potential positive samples for each individual classifier using Eq. (24). The FNR is the proportion of assumed positive instances that were falsely beaten as negative Eq. (25) [44].

$$FPR = \frac{FP}{(FP + TN)} \quad (24)$$

$$FNR = \frac{FN}{(TP + FN)} \quad (25)$$

Results when class-specific detectors have not been used show performance metrics at different label rates. At label rate 0.1, the model exhibited the following performance metrics: accuracy 0.9887, precision 0.9756, recall 0.8222, and F1 score of 0.8579. The false positive rate (FPR) and the false negative rate (FNR) are also measured at 0.0037 and 0.1778 respectively. These values change as the label rate increases to 1.0, the accuracy increases slightly to 0.9936, precision increases to 0.9285, recall increases to 0.9064, and F1 score changes to 0.9163. Label rates of 1.0 show both measures of FPR and FNR at 0.0020 and 0.0936 respectively.

The incorporation of detectors specific to the respective classes considerably raises the efficacy of the model Table 2. With a 0.1 label rate, the model score rises to 0.9862 while its precision lowers to 0.8454. The model recall is set at 0.8957 while the F1 score is raised to 0.8677. FPR and FNR comes at 0.0037 and 0.1043 respectively. At a label rate of 1.0, the graph depicts that the model is able to reach a peak score of 0.9943 at label rate precision of 0.9540, recall of 0.9066, and F1 score of 0.9265. It is denoted that both FPR and FNRs are still lowered within the model. After a model label rate of 1.0, the FNR level is then elevated to 0.0934.

In Figure 9 depicts enhanced performance using the model depicts enhanced performance on the basis of precision, recall, and F1-Score than the model without using it. The precision for label 1.0 is improved (0.9411 vs. 0.9340), the recall is improved (0.9063 vs. 0.8835), and the F1-Score is improved (0.9215 vs. 0.9038). This demonstrates that the inclusion of the detector autoencoder enhances the model's accuracy and reliability in classification tasks.

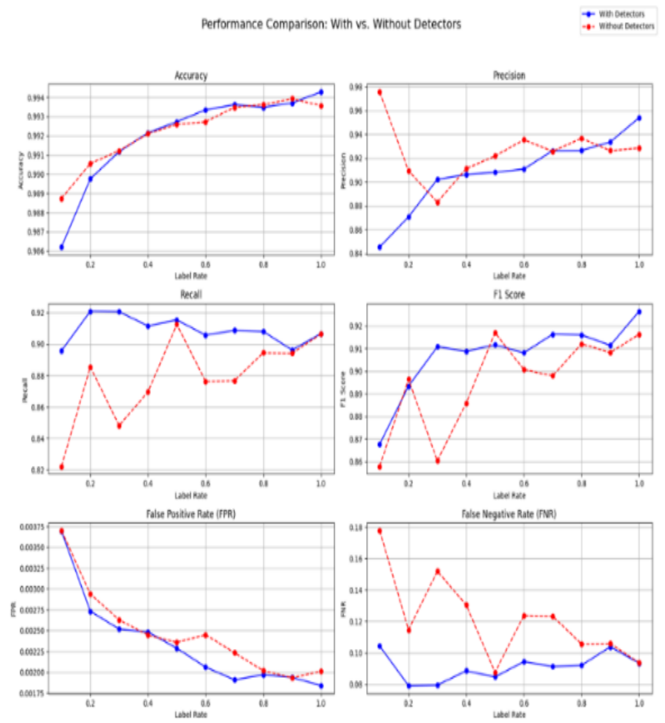


**Figure 9.** Key performance metric of Semi-Supervised Variational Autoencoder (VAE) model with and without class-specific detectors and hybrid scoring

## 6. DISCUSSION

First, a discussion for the results of each table is presented in the former Results section individually, next we do some comparative analysis between it. The system's monitoring of emerging attack trends (such as zero-day threats) is still dormant. This challenge stems from the fact that the NSL-KDD dataset predominantly uses known attack signatures. While our model attempts to semi-supervise VAE by using unlabeled data to construct normal traffic patterns, it suffers from using specific detectors designed for class labels, typically bounded by the scope of set attack types like DoS and Probe.

### 6.1 Performance without class-specific detectors

Considering Table 1, it can be seen that with the rise in the label rate, the proposed model achieves a steadily enhanced accuracy up to the maximum label rate of 100%, with the accuracy achieved being 99.36%. Overall, precision is present at an appreciable level for all label rates, although higher label rates tend to yield a lower score; this is most probable because at low rates, such as 0.1, the model is able to capture many actual positives. Nevertheless, the recall of this model at this low label rate decreases to only 82.22%, which suggests that our model does not have a high capability of identifying a large number of true anomalies in case the number of labelled data is small. FPR and FNR both decrease more with an increase in label rate, nevertheless, the FNR improvement is considerably more significant: when exposed to more significant labeled data amounts, the model demonstrates better capacity to isolate true anomalies.

### 6.2 Impact of class-specific detectors and hybrid scoring mechanism

Table 2 outlines the results of experiment based on the Semi-Supervised VAE model trained with the class-specific detectors as well as the proposed hybrid scoring system. On full labeling and with no detectors involved the model has an accuracy of 99.43% outperforming the earlier tested version. The precision gradually escalates as the label rate and becomes more precise at a 100% label rate of 95.40%. The results reveal that recall is always higher than the baseline model, showing that the class-specific detectors hold true anomalies even if there is a lack of labelled data. Further, FPR and FNR statistics are reduced over all labelled rates compared to the baseline model, which points to the fact that the hybrid scoring mechanism has the potential of improving the overall detection accuracy.

### 6.3 Comparative analysis

The combination of class-specific detectors and hybrid scoring mechanism improves the benchmark evaluation for all label rates in terms of precision and recall. This enhancement validates our hypothesis of the confinement of the detectors to the individual class features thus handling false alarms and improving the detection of true anomalies. Semi-Supervised VAE's numerical latent space is depicted in Figure 3. The results obtained clearly indicate clustering of normal and anomalous traffic patterns and clear separation of different attacks. This separation indicates that the model was able to segment different classes of network behavior quite well. The Normal class forms a tight, well-defined cluster, while each attack type occupies its own distinct region in the latent space. The class-specific detectors contributed to this well-structured separation by refining the latent space, ensuring that anomalies were more easily distinguished from normal traffic. Although there is a slight overlap between the Normal and DoS clusters, the overall clarity of separation between attack types, such as Probe, U2L, and U2R reflects the model's robustness in anomaly detection.

These validation and training loss curves further reflect the performance of the given method. In models trained with class-specific detectors, both training and validation losses showed a consistent decline, with no indications of overfitting, as shown in Figure 4 (a). The losses converged after 65 epochs, and the early stopping mechanism ensured that the model was not overtrained. In contrast, the model trained without detectors showed signs of overfitting, particularly after 45 epochs, where training loss continued to decrease, while validation loss rose, as shown in Figure 4 (b). This divergence highlights the importance of class-specific detectors and hybrid scoring in ensuring of performance in predicting unseen data.

The model having detectors trained for all classes demonstrate a less variability at lower label rates of 0.1-0.3 demonstrating the independence of the functionality from the labeled data. This makes the approach particularly suitable for the semi-supervised scenarios, where quantity of the labelled data is low. In addition, FPR and FNR are lower in case of class-specific detectors for all label rates, with a higher benefit at higher label rates as in Figure 5. These results show the optimality of employing hybrid scoring mechanism to assert the continual improvement of the approach's anomaly detection by minimizing misclassifications.

The model incorporating the detector autoencoder's performance is significantly better than the one not using it as shown in Figure 5. The model in question always achieves higher scores in precision, recall, and F1-Score, which suggests that there may be less false positive results and more true positive results. With label 1.0 affixed, the precision is 0.9411, and recall is at 0.9063, which is an improvement from 0.9340 and 0.8835 when the detector is not used. Furthermore, the detector autoencoder provides better F1-Scores and preserves the accuracy of the classifier, which indicates more balanced performance across different labels. To sum up, the model with such robustness does outperform the rest, which is now and no doubt much better for the task in question.

In Table 1 and Table 2 Precision and recall results show promising enhancements where class-specific detectors are applied as the findings of this study demonstrate. At lower label rates, where other methods generally fail to recognize anomalies, our approach yielded fairly high recall. This implies that the detectors incorporated the features of each class, and the model was able to flag true anomalous examples without much supervision [47]. The hybrid scoring mechanism also helped to further decrease the values of) FPR as well as FNR thus improving the accuracy in normal traffic detection and elimination of hypothetical alarms.

As seen from the results, our model has a high accuracy and stability when the label rate reduces to as low as 10% and this makes it suitable in cases where it is extremely expensive or time-consuming to label data. In that sense, by the efficient use of unlabeled data, the Semi-Supervised VAE with class-specific detectors proves the efficacy of an approach that is more scalable and less dependent on labeled data than fully-supervised models that rEq.uire large samples of labeled data to reach comparable performance levels [48].

The enhancement of anomaly detection performance has meaningful consEq.uences for present network security systems, especially for the areas where new sorts of cyber threats regularly encountered. Conventional IDS methods which use predominantly rule-based or supervised approaches lack the ability to detect emerging threats since their operation depends on labeled data. One way that our proposed model helps in overcoming this limitation is by using a semi-supervised learning method that can effectively learn and adapt to new data, it is an effective method for real-time network monitoring [49]. Even the high recall we get with so little labeled data indicates that this model may be handy in real-world scenarios where labeling the network traffic is not possible immediately. Reducing on heavy labeled datasets, the model enables organizations to detect and respond to cyber threats faster and more effectively.

The class-specific detectors used in the proposed model are highly resilient to different types of attacks and aim at being applicable to various network conditions [51]. However, the outcomes of this study bear some limitations. First, the proposed framework was tested on the NSL-KDD dataset which is one of the most frEq.uently used datasets in the literature of network intrusion detection. Despite the account of many attack types in this dataset, it might not sufficiently reflect the nature of the actual network traffic. Slight developmental differences of the model and comparison of its performance with more diverse and dynamic data have not been elaborated yet which is crucial for analyzing the model performance in conditions of changing threats [52]. Furthermore, the fact that the DoS and normal traffic clusters are partially merged in the latent space diagram also reveals a disadvantage of the approach - it may be problematic to distinguish between individual attack types. While the model have good accuracy in overall model, it could be seen a lot of overlap between the classes, which means that there may still be some fine tuning needed on the part of the machine learning algorithms in order to deliver increased levels of differentiation between different forms of attack, particularly where they overlap in the way that DoS does with DDoS [55].

One potential path of development is making the hybrid scoring mechanism gather further contextual data from the patterns in network traffic. Temporality or utilizing sEq.uence-based models including LSTMs could enable the model to analyze how the pattern dynamics in the network occur over time hence enhance its ability to identify long-term traffic patterns as anomalous. Last, yet importantly, there are efforts to minimize the computational cost of the algorithm but without much loss of efficiency in terms of implementation. Further research focusing on variants of the model which would allow for processing, in real-time, networks that contain intruding elements, with low resource consumption would be an interesting follow-up.

## 7. CONCLUSIONS

In this paper, the proposed multi-class anomaly detection using semi-supervised VAE (MCSS-VAE) presents a remarkably significant advance in the attempt to solve the challenges of multi-class anomaly detection faced by network intrusion systems. This research tries to solve the multi-class anomaly detection challenge within network intrusion detection systems featuring imbalanced classes, low labels, and high precision rEq.uirements for distinguishing attacks. Our main contribution is the merging of bio-inspired algorithms and semi-supervised deep learning, which offers new insight into tackling cyber security problems while also expanding the field's theoretical boundaries and practical implications. Our approach achieves increased efficiency and accuracy in the network by utilizing a variational autoencoder as an unsupervised encoder and introducing class-specific detectors with a hybrid scoring mechanism. This approach will leverage both NSA and CSA to fine-tune the class-specific latent spaces for robust and accurate anomaly detection. The MCSS-VAE is hence capable of bringing improvement in precision, recall, and overall detection performance; hence, it shows great promise in real-world multi-class anomaly detection tasks. These results validate the effectiveness of our method and show potential to achieve improved anomaly detection compared with previous semi-supervised models. Although experiments were conducted on the NSL-KDD dataset, further research is needed to analyze its effectiveness on other similar traffic and in real-time scenarios. The proposed model can be applied to large-scale networks while ensuring further enhanced efficiency and is therefore the potential answer to many contemporary network security mechanisms.

## REFERENCES

[1] Saeed, S., Altamimi, S.A., Alkayyal, N.A., Alshehri, E., Alabbad, D.A. (2023). Digital transformation and cybersecurity challenges for businesses resilience: Issues and recommendations. Sensors, 23(15): 6666. https://doi.org/10.3390/s23156666

[2] Arogundade, O.R. (2023). Network security concepts, dangers, and defense best practical. Computer Engineering and Intelligent Systems, 14(2): 25-38.

[3] Š vihrová, R., Lettner, C. (2020). A semi-supervised approach for network intrusion detection. In Proceedings of the 15th International Conference on Availability, Reliability and Security, pp. 1-6. https://doi.org/10.1145/3407023.3407073

[4] Bhati, N.S., Khari, M., García-Díaz, V., Verdú, E. (2020). A review on intrusion detection systems and techniques. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 28(Supp02): 65-91. https://doi.org/10.1142/S0218488520400140

[5] Alzaabi, F.R., Mehmood, A. (2024). A review of recent advances, challenges, and opportunities in malicious insider threat detection using machine learning methods. IEEE Access, 12: 30907-30927. https://doi.org/10.1109/ACCESS.2024.3369906

[6] Dong, S., Xia, Y., Peng, T. (2021). Network abnormal traffic detection model based on semi-supervised deep reinforcement learning. IEEE Transactions on Network and Service Management, 18(4): 4197-4212. https://doi.org/10.1109/TNSM.2021.3120804

[7] Xu, W., Jang-Jaccard, J., Singh, A., Wei, Y., Sabrina, F. (2021). Improving performance of autoencoder-based network anomaly detection on nsl-kdd dataset. IEEE Access, 9: 140136-140146. https://doi.org/10.1109/ACCESS.2021.3116612

[8] Li, Y., Xu, Y., Cao, Y., Hou, J., Wang, C., Guo, W., Li, X., Xin, Y., Liu, Z., and Cui, L. (2022). One-class LSTM network for anomalous network traffic detection. Applied Sciences, 12(10): 5051. https://doi.org/10.3390/app12105051

[9] Nguyen, V.Q., Ngo, L.T., Nguyen, V.H., Shone, N. (2024). Deep clustering hierarchical latent representation for anomaly-based cyber-attack detection. Knowledge-Based Systems, 301: 112366. https://doi.org/10.1016/j.knosys.2024.112366

[10] Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A. (2020). Application of deep reinforcement learning to

intrusion detection for supervised problems. Expert Systems with Applications, 141: 112963. https://doi.org/10.1016/j.eswa.2019.112963

[11] Mbona, I., Eloff, J.H. (2022). Detecting zero-day intrusion attacks using semi-supervised machine learning approaches. IEEE Access, 10: 69822-69838. https://doi.org/10.1109/ACCESS.2022.3187116

[12] Zhang, Y., Niu, J., He, G., Zhu, L., Guo, D. (2021). Network intrusion detection based on active semi-supervised learning. In 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), pp. 129-135. https://doi.org/10.1109/DSN-W52860.2021.00031

[13] Mishra, P., Varadharajan, V., Tupakula, U., Pilli, E.S. (2018). A detailed investigation and analysis of using machine learning techniques for intrusion detection. IEEE Communications Surveys & Tutorials, 21(1): 686-728. https://doi.org/10.1109/COMST.2018.2847722

[14] Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A. (2009). A detailed analysis of the KDD CUP 99 data set. In 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1-6. https://doi.org/10.1109/CISDA.2009.5356528

[15] Najafabadi, M.M., Khoshgoftaar, T.M., Kemp, C., Seliya, N., Zuech, R. (2014). Machine learning for detecting brute force attacks at the network level. In 2014 IEEE International Conference on Bioinformatics and Bioengineering, pp. 379-385. https://doi.org/10.1109/BIBE.2014.73

[16] Chugh, N., Tomar, G.S., Bhadoria, R.S., Saxena, N. (2021). A novel anomaly behavior detection scheme for mobile ad hoc networks. Electronics, 10(14): 1635. https://doi.org/10.3390/electronics10141635

[17] Pervez, M.S., Farid, D.M. (2014). Feature Selection and Intrusion Classification in NSL-KDD Cup 99 Dataset Employing SVMs. In the 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014), pp. 1-6. https://doi.org/10.1109/SKIMA.2014.7083539

[18] Shahadat, N., Hossain, I., Rohman, A., Matin, N. (2017). Experimental analysis of data mining application for intrusion detection with feature reduction. In 2017 International Conference on Electrical, Computer and Communication Engineering (ECCE), pp. 209-216. https://doi.org/10.1109/ECACE.2017.7912907

[19] Tan, Z., Jamdagni, A., He, X., Nanda, P., Liu, R.P., Hu, J. (2014). Detection of denial-of-service attacks based on computer vision techniques. IEEE Transactions on Computers, 64(9): 2519-2533. https://doi.org/10.1109/TC.2014.2375218

[20] Hemmer, A., Abderrahim, M., Badonnel, R., Chrisment, I. (2021). An ensemble learning-based architecture for security detection in IoT infrastructures. In 2021 17th International Conference on Network and Service Management (CNSM), pp. 180-186. https://doi.org/10.23919/CNSM52442.2021.9615588

[21] Vanerio, J., Casas, P. (2017). Ensemble-learning approaches for network security and anomaly detection. In Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks, pp. 1-6. https://doi.org/10.1145/3098593.3098594

[22] Al-zubidi, A.F., Farhan, A.K., Towfek, S.M. (2024). Predicting DoS and DDoS attacks in network security scenarios using a hybrid deep learning model. Journal of Intelligent Systems, 33(1): 20230195. https://doi.org/10.1515/jisys-2023-0195

[23] Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R., Ghogho, M. (2016). Deep learning approach for network intrusion detection in software defined networking. In 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), pp. 258-263. https://doi.org/10.1109/WINCOM.2016.7777224

[24] Loukas, G., Vuong, T., Heartfield, R., Sakellari, G., Yoon, Y., Gan, D. (2017). Cloud-based cyber-physical intrusion detection for vehicles using deep learning. IEEE Access, 6: 3491-3508. https://doi.org/10.1109/ACCESS.2017.2782159

[25] Li, Z., Qin, Z., Huang, K., Yang, X., Ye, S. (2017). Intrusion detection using convolutional neural networks for representation learning. In International Conference on Neural Information Processing, pp. 858-866. https://doi.org/10.1007/978-3-319-70139-4_87

[26] Wang, W., Sheng, Y., Wang, J., Zeng, X., Ye, X., Huang, Y., Zhu, M. (2017). HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. IEEE Access, 6: 1792-1806. https://doi.org/10.1109/ACCESS.2017.2780250

[27] Truong-Huu, T., Dheenadhayalan, N., Pratim Kundu, P., Ramnath, V., Liao, J., Teo, S.G., Praveen Kadiyala, S., (2020). An empirical study on unsupervised network anomaly detection using generative adversarial networks. In Proceedings of the 1st ACM Workshop on Security and Privacy on Artificial Intelligence, pp. 20-29. https://doi.org/10.1145/3385003.3410924

[28] Su, T., Sun, H., Zhu, J., Wang, S., Li, Y. (2020). BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset. IEEE Access, 8: 29575-29585. https://doi.org/10.1109/ACCESS.2020.2972627

[29] Madani, P., Vlajic, N. (2018). Robustness of deep autoencoder in intrusion detection under adversarial contamination. In Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security, pp. 1-8. https://doi.org/10.1145/3190619.3190637

[30] Al-zubidi, A.F., Farhan, A.K., El-Kenawy, E.S.M. (2024). Surveying machine learning in cyberattack datasets: A comprehensive analysis. Journal of Soft Computing and Computer Applications, 1(1): 1. https://doi.org/10.70403/3008-1084.1000

[31] Zhou, C., Paffenroth, R.C. (2017). Anomaly detection with robust deep autoencoders. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 665-674. https://doi.org/10.1145/3097983.3098052

[32] Qi, Y., Wang, Y., Zheng, X., Wu, Z. (2014). Robust feature learning by stacked autoencoder with maximum correntropy criterion. In 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6716-6720. https://doi.org/10.1109/ICASSP.2014.6854900

[33] Alsaedi, E.M., Farhan, A.K., Falah, M.W., Oleiwi, B.K. (2022). Classification of encrypted data using deep learning and legendre polynomials. In the International Conference on Innovations in Computing Research, pp. 331-345. https://doi.org/10.1007/978-3-031-14054-9_31

[34] Khan, F.A., Gumaei, A., Derhab, A., Hussain, A. (2019). A novel two-stage deep learning model for efficient

network intrusion detection. IEEE Access, 7: 30373-30385. https://doi.org/10.1109/ACCESS.2019.2899721

[35] Al-Qatf, M., Lasheng, Y., Al-Habib, M., Al-Sabahi, K. (2018). Deep learning approach combining sparse autoencoder with SVM for network intrusion detection. IEEE Access, 6: 52843-52856. https://doi.org/10.1109/ACCESS.2018.2869577

[36] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A., Bottou, L. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research, 11(12): 1-38. https://doi.org/10.5555/1756006.1953039

[37] Oleiwi, B.K., Abood, L.H., Farhan, A.K. (2022). Integrated different fingerprint identification and classification systems based deep learning. In 2022 International Conference on Computer Science and Software Engineering (CSASE), pp. 188-193. https://doi.org/10.1109/CSASE51777.2022.9759632

[38] Alhussan, A.A., Farhan, A.K., Abdelhamid, A.A., El-Kenawy, E.S.M., Ibrahim, A., Khafaga, D.S. (2023). Optimized ensemble model for wind power forecasting using hybrid whale and dipper-throated optimization algorithms. Frontiers in Energy Research, 11: 1174910. https://doi.org/10.3389/fenrg.2023.1174910

[39] Shone, N., Ngoc, T.N., Phai, V.D., Shi, Q. (2018). A deep learning approach to network intrusion detection. IEEE Transactions on Emerging Topics in Computational Intelligence, 2(1): 41-50. https://doi.org/10.1109/TETCI.2017.2772792

[40] Lake, B., Salakhutdinov, R., Gross, J., Tenenbaum, J. (2011). One Shot learning of simple visual concepts. In Proceedings of the Annual Meeting of the Cognitive Science Society, 33. https://escholarship.org/uc/item/4ht821jx

[41] Wang, W., Du, X., Shan, D., Qin, R., Wang, N. (2020). Cloud intrusion detection method based on stacked contractive auto-encoder and support vector machine. IEEE Transactions on Cloud Computing, 10(3): 1634-1646. https://doi.org/10.1109/TCC.2020.3001017

[42] Saadi, Z.M., Sadiq, A.T., Akif, O.Z., Farhan, A.K. (2024). A Survey: Security vulnerabilities and protective strategies for graphical passwords. Electronics, 13(15): 3042. https://doi.org/10.3390/electronics13153042

[43] Mohammed, A.A., Al-Ghrairi, A.H.T., Al-zubidi, A.F., Saeed, H.M. (2023). Unsupervised classification and analysis of istanbul-turkey satellite image utilizing the remote sensing. In AIP Conference Proceedings. https://doi.org/10.1063/5.0118339

[44] Tang, Z., Wang, P., Wang, J. (2020). ConvProtoNet: Deep prototype induction towards better class representation for few-shot malware classification. Applied Sciences, 10(8): 2847. https://doi.org/10.3390/app10082847

[45] Hindy, H., Tachtatzis, C., Atkinson, R., Bayne, E., Bellekens, X. (2021). Developing a siamese network for intrusion detection systems. In Proceedings of the 1st Workshop on Machine Learning and Systems, pp. 120-126. https://doi.org/10.1145/3437984.3458842

[46] Saadi, Z.M., Sadiq, A.T., Akif, O.Z., El-kenawy, E.S.M. (2025). Enhancing convolutional neural network for image retrieval. Journal of Intelligent Systems and Internet of Things, 2: 40-140. https://doi.org/10.54216/JISIoT.140212

[47] Aouedi, O., Piamrat, K., Bagadthey, D. (2020). A semi-supervised stacked autoencoder approach for network traffic classification. In 2020 IEEE 28th International Conference on Network Protocols (ICNP), pp. 1-6. https://doi.org/10.1109/ICNP49622.2020.9259390

[48] Aouedi, O., Piamrat, K., Muller, G., Singh, K. (2022). FLUIDS: Federated learning with semi-supervised approach for intrusion detection system. In 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC), pp. 523-524. https://doi.org/10.1109/CCNC49033.2022.9700632

[49] Abdelkhalek, A., Mashaly, M. (2023). Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning. The Journal of Supercomputing, 79(10): 10611-10644. https://doi.org/10.1007/s11227-023-05073-x

[50] Hooshmand, M.K., Hosahalli, D. (2022). Network anomaly detection using deep learning techniques. CAAI Transactions on Intelligence Technology, 7(2): 228-243. https://doi.org/10.1049/cit2.12078

[51] Ameen, Z.H., AL-Bakri, N.F., Al-zubidi, A.F., Hashim, S.H., Jaaz, Z.A. (2024). A new COVID-19 patient detection strategy based on hidden naïve bayes classifier. Iraqi Journal of Science, 6705-6724. https://doi.org/10.24996/ijs.2024.65.11.41

[52] Alsaedi, E.M., Kadhim Farhan, A. (2023). Retrieving encrypted images using convolution neural network and fully homomorphic encryption. Baghdad Science Journal, 20(1): 206. https://doi.org/10.21123/bsj.2022.6550

[53] Andresini, G., Appice, A., Di Mauro, N., Loglisci, C., Malerba, D. (2019). Exploiting the auto-encoder residual error for intrusion detection. In 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), pp. 281-290. https://doi.org/10.21123/bsj.2022.6550

[54] Zoppi, T., Gharib, M., Atif, M., Bondavalli, A. (2021). Meta-learning to improve unsupervised intrusion detection in cyber-physical systems. ACM Transactions on Cyber-Physical Systems (TCPS), 5(4): 1-27. https://doi.org/10.1145/3467470

[55] Zhang, S., Ye, F., Wang, B., Habetler, T.G. (2019). Semi-supervised learning of bearing anomaly detection via deep variational autoencoders. arXiv preprint arXiv:1912.01096. https://doi.org/10.48550/arXiv.1912.01096

[56] Hassan Zaib, M. NSL-KDD. https://www.kaggle.com/datasets/hassan06/nslkdd.