Deep Clustering Hierarchical Latent Representation for Anomaly-Based Cyber-Attack Detection

Van Quan Nguyen[a], Long Thanh Ngo[a,*], Le Minh Nguyen[b], Viet Hung Nguyen[a], Nathan Shone[c]

[a]*Le Quy Don Technical University, Viet Nam*
[b]*Japan Advanced Institute of Science and Technology, Japan*
[c]*Liverpool John Moores University, United Kingdom*

Abstract

In the field of anomaly detection, well-known techniques and state-of-the-art models often face challenges when interpreting the latent space, which hinders their behavioral classification accuracy. Firstly, the sub-optimal distribution of data points within the latent space makes normal behavioral regions verbose and indistinguishable from abnormal regions. Secondly, within the latent space, it can be difficult to identify meaningful, separable, and indicative features. Finally, the processing time at the inference stage is still relatively slow.

This paper aims to improve the accuracy of network anomaly detection mechanisms by proposing two novel deep hierarchical representation learning models: Deep Nested Clustering Auto-Encoder (DNCAE) and Deep Clustering Hierarchical Auto-Encoder (DCHAE). Both models adopt a nested branch structure, utilizing dual deep auto-encoders to establish hierarchical latent spaces; in each, clustering algorithms are used to spatially optimize and refine the data points. This approach results in improved separation between normal and abnormal data points, and easier identification of notable and/or indicative features.

To ascertain the effectiveness of the approach and the quality of resulting features, both models were used in conjunction with ten different one-class anomaly detectors. Each of these ten anomaly detectors was evaluated on popular network intrusion datasets, notably: NSL-KDD, UNSW-NB15, CIC-IDS-2017, CSE-CIC-IDS-2018, and CTU13. Experimental results have confirmed that both of the proposed models produced higher levels of accuracy than existing baselines and current state-of-the-art models. Additionally, the processing time at the inference stage shows a significant reduction.

Keywords: Latent Representation, Deep Learning, Cyber-Attack Detection, Anomaly Detection, Deep Clustering

1. Introduction

In recent years, the world has witnessed a strong wave of digital transformation in almost all fields of the Industrial Revolution 4.0. The explosion of global internet usage and the advent of new technologies have facilitated unprecedented achievements such as intelligent transportation, smart cities, smart homes, online transactions, next-generation healthcare, and educational platforms. Consequently, this has resulted in vast amounts of data being stored in and transmitted through cyberspace [1]. Cyberspace therefore faces evolving security challenges from attackers targeting data and information infrastructure [2]. The scale and scope of cyber-attack campaigns targeting agencies, organizations, and businesses are increasing and causing more severe impacts [3]. Cyber-attacks continually evolve, embracing increasingly sophisticated, complex, modern, and multi-stage techniques to target data, services, applications, and infrastructure [4–6]. New types of malicious exploit code, novel vulnerabilities, and zero-day attacks are being created every day to deploy Advanced Persistent Threat (APT) campaigns and Distributed Denial of Service (DDoS) attacks causing significant damage worldwide [7–9]. In addition, the ubiquity of the Internet of Things (IoT) devices and cloud technology increasingly create larger attack surfaces and more attack vectors [10–12]. In particular, attackers create sophisticated payloads and malicious attack techniques that either closely resemble normal network traffic or appear with such low frequency to avoid detection [13]. The flexible use of obfuscation, encryption, and evasion techniques has caused great difficulties and challenges for network administrators [14].

Network Intrusion Detection Systems (NIDSs) have been widely recognized as an effective solution to prevent and detect cyber-attacks [15]. In general, NIDS can be divided into two categories: Signature-Based Intrusion Detection System (SIDS) and Anomaly-Based Intrusion Detection System (AIDS) [16]. SIDS detects cyber-attacks using a predefined database of attack signatures. Although SIDS can detect known attacks with high accuracy, it suffers from many inherent limitations. Firstly, it cannot detect zero-day attacks or even variants of known attacks. Secondly, matching attack signatures from a large database is a time-consuming task. Finally, it requires frequent attack signature updates and is ineffective when

*Corresponding authors
Email addresses: quannv@lqdtu.edu.vn (Van Quan Nguyen), ngotlong@mta.edu.vn (Long Thanh Ngo), nguyenml@jaist.ac.jp (Le Minh Nguyen), hungnv@lqdtu.edu.vn (Viet Hung Nguyen), n.shone@ljmu.ac.uk (Nathan Shone)

applied to new network topologies and environments [17]. In contrast, AIDS establishes a profile of expected behaviors, and deviations from this norm exceeding a predefined threshold are labeled as abnormal [18]. One of the key advantages of AIDS is its ability to detect zero-day attacks, as well as known attack variants.

In recent years, the use of Deep Learning (DL) models to build AIDS solutions has gained significant attention from the information security research community. Specifically, many model architectures have been investigated and applied to detect cyber-attacks, such as: Deep Belief Networks (DBNs) [19], Convolutional Neural Networks (CNNs) [20], Recurrent Neural Networks (RNNs) [21], Autoencoders (AEs) [22], and hybrid models. Among these architectures, AE is a model that is widely applied in research for AIDS development [23]. This comes from AE's powerful latent representation learning ability, which is beneficial when applied to network traffic. Despite this, there is still scope for improving the performance of such models, especially in terms of accuracy, false alarm rate, and processing time during inference.

This paper focuses on addressing three inherent limitations of state-of-the-art solutions, which are hindering the development of high-performance IDS systems:

1. Existing DL models have not learned the latent representation space adequately enough to describe normal network behavior, making the accurate detection of anomalies challenging. This can be attributed to the constantly changing definition of "normal" behavior in a network, the emergence of new device types and/or functionalities, and the diversity of communication protocols and applications. However, we argue that the root cause of this problem is that the proposed models cannot capture the core characteristics of normal network traffic. In other words, the learned latent representation spaces are not the most indicative features to represent normal network data and thereby clearly distinguish it from anomalies.

2. The distribution of normal network data points within the learned latent space is insufficiently compact and therefore leads to difficulties in determining the boundary between normal and abnormal data areas. This often means that the achievable accuracy and false alarm rates are insufficient for detecting zero-day attacks. Many reasons can contribute towards this, such as the diverse characteristics of normal network data and the imbalance of normal network data types. However, we argue that the existing methods have not exploited the power of appropriate regularizers to adequately consolidate normal network data points within the learned latent space.

3. The required processing time for ascertaining whether data points are normal or abnormal during the inference phase is still slow, and therefore unsuitable for application in real-time systems. There are many contributing factors causing this problem, such as existing approaches having high computational complexity, high dimensionality of the learned latent space, and the use of unsuitable features.

This paper proposes two novel DL-based models to overcome the aforementioned inherent limitations. Our first proposed model is called Deep Nested Clustering Auto-Encoder (DNCAE), referred to as DNCAE in this paper. It consists of two nested Deep Autoencoders (DAEs): Outer DAE and Inner DAE. In the latent space of the Outer DAE, the K-means clustering algorithm is utilized to group similar data points into sub-clusters, forcing these cluster centers closer together. Then, Inner DAE will compress this clustering architecture further to optimize the volume of the normal network data regions. This resultant architecture will significantly reduce the dimensionality of the latent layer of Inner DAE, thus improving inference time at processing, overcoming the outlined limitations. Our second proposed model is called Deep Clustering Hierarchical Auto-Encoder (DCHAE), referred to as DCHAE in this paper. This consists of two DAEs, called Big DAE and Small DAE. The difference between DNCAE and DCHAE, is that Big DAE and Small DAE are separate in structure. The latent space of Big DAE is the input layer of Small DAE, and the output of Small DAE is not connected to Big DAE. The bottleneck layer of Big DAE is also integrated with the K-means clustering algorithm to force the optimal distribution of normal network data points in the latent space. Then, Small DAE will compress this clustered architecture more tightly at its latent layer differently than the first model. Technical details of DNCAE and DCHAE will be presented in Section 4. The overall general architecture of DNCAE and DCHAE are shown in Figure 2 and Figure 3, respectively. We evaluate the performance of the proposed models using benchmark data sets, including NSL-KDD, UNSW-NB15, CIC-IDS-2017, CSE-CIC-IDS-2018, and CTU-13. In summary, the main contributions of this work are as follows:

- We propose two novel DL-based models for anomaly-based cyber-attack detection. The first model is DNCAE (DNCAE), consisting of two nested DAEs: Outer DAE and Inner DAE. Additionally, the K-means clustering algorithm is integrated into the latent layer of Outer DAE to classify normal network data points into appropriate clusters in the latent space. The second model, called DCHAE (DCHAE), consists of two DAEs: Big DAE and Small DAE. The two DAEs are linked in a branching architecture where their intersection is the Big DAE's latent layer and the Small DAE's input layer. The K-means algorithm is integrated at the intersection of two DAEs to explore the cluster architecture of data points in the latent space.

- We investigate the impact of the K-means algorithm's number of clusters on the proposed models' performance to expose the hidden cluster architecture of standard data sets.

- We evaluate the processing time at inference by observing the time taken by each model during training and testing.

- We conduct comprehensive experiments to evaluate the performance of the proposed models and compare these against baselines and state-of-the-art models [23] and [24]. This is undertaken using a range of established NIDS data

sets, including NSL-KDD, UNSW-NB15, CIC-IDS-2017, CSE-CIC-IDS-2018, and CTU-13.

This work is an extension of our previously published research [25],we note the following additions to this work: inclusion of CSE-CIC-IDS 2018 and CTU13 datasets in the evaluations; proposal of the DCHAE model; evaluation of the DCHAE model on the benchmark datasets; investigation into the impact of the number of clusters of the K-means algorithm; evaluation of training and testing time for both models on all datasets; and comparative analysis of DNCAE and DCHAE with baselines and state-of-the-art models.

The remainder of this paper is organized as follows. Section 2 briefly r eviews p rominent a nd c utting-edge research into anomaly-based cyber-attack detection. Then, Section 3 presents the mathematical background of the DAE model and K-means algorithm. Our proposed DNCAE and DHCAE models are described in Section 4. Experiments, results, and discussion are presented in Sections 5 and 6, respectively. Finally, we conclude our paper in Section 7 and highlight future research directions.

2. Related Works

This section will summarise some of the latest trends and cutting-edge research pertinent to this work.

2.1. Autoencoder-based Cyber Attack Detection

Nicolau et al. [23] proposed two latent representation learning models to improve the performance of network anomaly detection. The models, Shrink Autoencoder and Dirac Delta Variational Autoencoder feature new integrated regularizers, which force normal data closer to the origin of latent space. Evaluations undertaken on common NIDS datasets demonstrated detection accuracy improvements. Similarly, Vu et al. [26] also proposed two latent representation learning models for anomaly detection: Multidistribution Variational Autoencoder and Multidistribution Denoising Autoencoder. The paper proposes new versions of regularizers at the latent layers of VAE and DAE, respectively. These models are trained using normal data and known IoT attacks in a supervised manner. Experimental results on nine IoT datasets have shown that the proposed models improve the ability to detect unknown IoT attacks.

In [27], the authors introduced their Adversarial AutoEncoder (AAE) and Bidirectional Generative Adversarial Networks (BiGAN) models for cyber-attack detection. The evaluation was undertaken using the IoT-23 dataset, and yielded F-scores of 0.99 for both proposed models.

Vu et al. [28] proposed two deep generative learning models to detect cloud intrusion. The first model, Conditional Denoising Adversarial Autoencoder (CDAAE), generates specific attack data instances. The second model combines CDAAE and the KNN algorithm to generate borderline malicious data samples to improve cloud attack detection accuracy. The data generated from these two models is combined with existing data to create an augmented dataset. Three conventional classifiers, including Support Vector Machine (SVM), Decision Tree (DT), and Random Forest (RF), are trained using this augmented dataset. Experimental results on four benchmark data sets (cloud IDS, CIC-IDS 2017, NSL-KDD, and UNSW-NB15) have shown that the proposed methods have improved attack detection accuracy in cloud environments.

Another innovative approach is that proposed by Wu [29] with the Mislearned AutoEncoder (MisAE). Their solution aims to solve the problem of over-generalization by increasing the reconstruction error of abnormal data by incorrectly learning the pattern of these abnormal samples.

Other AE-related anomaly detection works have focused upon: comparing classification performance of DAE, AE, and One-Class Support Vector Machine (OC-SVM) when analyzing CIC-IDS 2017 [30]; the application of hierarchically structured AEs [31]; optimizing performance by using AE reconstruction feature residuals rather than summary residual metric [32]; and Deep Support Vector Data Description (DSVDD) combined with a Contractive Autoencoder (CAE) to outperform common algorithms including KMeans, OCSVM, and Isolation Forest [33].

2.2. CNN-based and RNN-based Cyber Attack Detection

In the work [34], the authors presented a study using the 1D-CNN model to detect network anomalies. In particular, they divided the network data into three subsets based on TCP, UDP, and Others protocols, which were trained separately. Several feature selection techniques are applied before training to improve the proposed model's performance. They present experimental results using the UNSW-NB15 dataset, achieving F-score results of 0.85, 0.97, and 0.86, for TCP, UDP, and Other protocols, respectively.

Kasongo in [35] presented research using RNN-based deep learning models to detect network intrusion. This solution uses different neural network architectures, including Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Vanilla RNN. To evaluate the performance of this solution, the author conducted experiments using NSL-KDD and UNSW-NB15 datasets. The XGBoost-based feature selection is applied to preprocess the data before fitting it into deep learning models. It is claimed that the proposed models yield better results than previous models regarding accuracy and F1 scores.

Similarly, Asmaa Halbouni et al. [36] proposed a hybrid CNN and RNN model, which leverages the the spatial feature learning ability of CNN and the temporal feature learning power of RNN. Various techniques such as batch normalization, dropout layers and standardization are used to avoid over-fitting and increase the accuracy. The experiments were conducted using three standard data sets: UNSW-NB15, CIC-IDS-2017, and WSN-DS datasets. The results showed that the CNN-LSTM model produces high detection rates and accuracy on several attack classes but encounters limitations in detecting Web-based attack classes.

2.3. Other Methods for Cyber Attack Detection

The authors in [37] have introduced an innovative extension of the Isolation Forest (iForest) algorithm for anomaly detec-

tion. Particularly, this solution uses a deep neural network-based random representation ensemble for a new representation scheme, enabling versatile data partitioning in various random directions across different subspaces. This approach enhances anomaly scoring by combining random representations with random partition-based isolation, leading to (i) more effective isolation of challenging anomalies in sparse and non-linear data, (ii) overcoming constraints to address artifact issues, and (iii) versatile handling of various data types. Extensive experiments show that their proposal significantly outperforms iForest and its extensions on tabular, graph, and time-series data, and offers promising improvements over state-of-the-art deep anomaly detectors.

Hongzuo Xu et al. [38] proposed a novel semi-supervised anomaly detection method aimed at addressing the shortcomings of current models including: (i) the presence of unlabeled anomalies (anomaly contamination), which can distort the learning process when all unlabeled data are considered normal, and (ii) the dependence on discrete supervision (binary or ordinal labels), which hinders the optimal learning of anomaly scores that naturally follow a continuous distribution. The key idea of this method is developing contamination-resilient continuous supervisory signals through a mass interpolation technique, which diffuses the abnormality of labeled anomalies to create new samples with continuous abnormal degrees. Additionally, the method covers contaminated areas by generating new samples through combinations of correctly labeled data. A feature learning-based objective is incorporated to regularize the network and enhance robustness against anomaly contamination. Extensive experiments on real-world datasets demonstrate that this approach significantly outperforms state-of-the-art methods and provides robust and superior performance across varying levels of anomaly contamination and different numbers of labeled anomalies.

In general, many deep learning-based architectures have been proposed for anomaly-based cyber-attack detection. However, these solutions still have limitations in discovering robust discriminative features that can improve the accuracy and false alarm rate when detecting zero-day attacks. In this work, we will propose two novel hierarchical latent representation learning models to overcome the above-mentioned limitations.

3. Background

In this section, we aim to present the mathematical background of the DAE model, K-means, and One-Class Classification (OCC) algorithms, which are the main components of our proposed solution.

3.1. Deep Auto-Encoder

DAE is a deep learning neural network architecture widely used for tasks such as data compression, dimensionality reduction, feature engineering, and image denoising [39–42]. With its powerful latent representation learning capabilities, DAE is the most commonly used architecture for anomaly detection purposes [23], [43–45]. It is trained to imperfectly copy input data to output data, prioritising the strongest features and approximating the input data. The DAE training process enables the discovery of powerful and meaningful representation spaces. Regarding the architecture, DAE consists of two main components: Encoder and Decoder. In modern DAE models, the Encoder and Decoder are essentially deep architecture models equipped with nonlinear activation functions to learn hierarchical abstract representations of data. The Encoder transforms the original raw data at the input layer into the latent space through the general function $\mathbf{F}(.)$. Whereas the Decoder will map samples from the latent space to the original space at the output layer, through the general function $\mathbf{G}(.)$. Let us denote the original training data set as $\mathbf{X} = \{\mathbf{x}_i | i = 1, 2, 3, ...n\}$; The latent representation of $\mathbf{X}$ at the output of the Encoder is $\mathbf{H} = \{\mathbf{h}_i | i = 1, 2, 3, ...n\}$, and the reconstruction of $\mathbf{X}$ at the output of the Decoder is $\hat{\mathbf{X}} = \{\hat{\mathbf{x}}_i | i = 1, 2, 3, ...n\}$. The responsibilities of the Encoder and Decoder parts are expressed through functions $\mathbf{F}(.)$ and $\mathbf{G}(.)$ as follows:

$$\mathbf{H} = \mathbf{F}_\theta(\mathbf{X}) \tag{1}$$

$$\hat{\mathbf{X}} = \mathbf{G}_\phi(\mathbf{H}) = \mathbf{G}_\phi(\mathbf{F}_\theta(\mathbf{X})) \tag{2}$$

Where, n is the number of instances in training set $\mathbf{X}$; $\mathbf{x}_i, \hat{\mathbf{x}}_\mathbf{i} \in \mathbf{R}^D$; $D \in \mathbf{N}$ is the dimension of original data; $\mathbf{h}_i \in \mathbf{R}^d$; $d \in \mathbf{N}$ is the dimension of latent representation space $\mathbf{H}$; θ and ϕ are weights and biases of Encoder and Decoder components, respectively. In the general case, $\mathbf{F}(.)$ and $\mathbf{G}(.)$ are nonlinear functions. The DAE training process optimizes the difference between the original data in set $\mathbf{X}$ and the reconstructed version in set $\hat{\mathbf{X}}$ at the output of the Decoder. However, the more important purpose is that through this optimization process, it is possible to learn condensed, prominent, and meaningful features in the latent space $\mathbf{H}$. More specifically, the DAE training process involves finding optimal solutions for functions $\mathbf{F}(.)$ and $\mathbf{G}(.)$ to satisfy:

$$\underset{\mathbf{F},\mathbf{G}}{\operatorname{argmin}} \mathbf{E}\Big[\mathbf{L}\Big(\mathbf{x}_i, \mathbf{G}\Big[\mathbf{F}(\mathbf{x}_i)\Big]\Big)\Big] \tag{3}$$

Here, $\mathbf{L}$ is the objective function to measure the dissimilarity between the original data and its reconstructed version at the output of the Decoder; $\mathbf{E}$ is the expectation value calculated over the entire training set $\mathbf{X}$. Choosing a specific form for the objective function depends mainly on the nature of the training data. If the training data is real values, the chosen objective function is Mean Squared Error (MSE), while if the data is in binary form, the used loss function is Binary Cross-Entropy (BCE). Specifically, the potential candidates for the objective function for DAE are shown as follows:

$$\mathbf{L}_{\text{MSE}}\Big(\mathbf{X}, \hat{\mathbf{X}}\Big) = \frac{1}{n}\sum_{i=1}^{n}\Big(\mathbf{x}_i - \hat{\mathbf{x}}_i\Big)^2 \tag{4}$$

$$\mathbf{L}_{\text{BCE}}\Big(\mathbf{X}, \hat{\mathbf{X}}\Big) = -\frac{1}{n}\sum_{i=1}^{n}\Big(\mathbf{x}_i \log(\hat{\mathbf{x}}_i) + (1 - \mathbf{x}_i)\log(1 - \hat{\mathbf{x}}_i)\Big) \tag{5}$$

In this article, we propose two novel variants for the DAE model architecture, in which two DAEs are connected in the same architecture in two different ways and integrate the clustering algorithm into the latent layer of a DAE (Outer DAE for DNCAE and Big DAE for DCHAE).

3.2. K-means

The K-means is one of the most popular clustering algorithms used to classify unsupervised data into a predefined number (K) of clusters [46],[47]. Let $\mathbf{X} = \{\mathbf{x}_i | i = 1, 2, 3, ...n\}$ is the D-dimensional dataset; $\mathbf{x}_i \in \mathbf{R}^D$; $D \in \mathbf{N}$; K is given the natural number ($K \in \mathbf{N}$). Our goal is to divide the dataset $\mathbf{X}$ into K clusters of samples, such that the distance between data points in the same cluster is smaller than the distance between data points outside the cluster. We further assume that each data point is in only one cluster. For each data point $\mathbf{x}_i$, we give a binary index $z_{ik} \in \{0, 1\}$ to determine which of the K clusters the data point belongs to. Each cluster is represented by a cluster center $\mathbf{c}_k$, which is also in D-dimensional space. The division of data set $\mathbf{X}$ into K clusters is the process of optimizing the following objective function:

$$\mathcal{L}_{Kmeans}(\mathbf{C}, \mathbf{Z}) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \parallel \mathbf{x}_i - \mathbf{c}_k \parallel^2 \tag{6}$$

Where, $\mathbf{C} = \{\mathbf{c}_k | k = 1, 2, 3...K\}$ is the centers of the K clusters; $\mathbf{Z} = \{z_{ik} | i = 1, 2, 3...n; k = 1, 2, 3..K\}$ is the binary index for dataset $\mathbf{X}$. These are two successive optimization steps with respect to z_{ik} and $\mathbf{c}_k$. First, while keeping $\mathbf{c}_k$ unchanged, we minimize $\mathcal{L}_{Kmeans}(\mathbf{C}, \mathbf{Z})$ with respect to z_{ik}. $\mathcal{L}_{Kmeans}(\mathbf{C}, \mathbf{Z})$ is a linear function of z_{ik}, so it is easy to obtain the assignment index of the data points in each cluster as follows:

$$z_{ik} = \begin{cases} 1 & \text{if k } = argmin_j \parallel \mathbf{x}_i - \mathbf{c}_k \parallel^2 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Next, while keeping $\mathbf{Z}$ fixed, we optimize $\mathcal{L}_{Kmeans}(\mathbf{C}, \mathbf{Z})$ with respect to $\mathbf{C}$, which is the quadratic function of $\mathbf{C}$. Taking derivative of $\mathcal{L}_{Kmeans}(\mathbf{C}, \mathbf{Z})$ with respect to $\mathbf{c}_k$ and give to zero we gain:

$$2 \sum_{i=1}^{n} z_{ik}(\mathbf{x}_i - \mathbf{c}_k) = 0 \tag{8}$$

The solution for $\mathbf{c}_k$ has the form:

$$\mathbf{c}_k = \frac{\sum_i z_{ik} \mathbf{x}_i}{\sum_i z_{ik}} \tag{9}$$

Generally, the process assigns data points to the nearest cluster and then recalculates the cluster center until the assignment is unchanged or a predetermined number of iterations is reached.

3.3. One-Class Classification Algorithms

This section will briefly present the One-Class Classification (OCC) algorithms [48] applied in our solution, including: One-Class Support Vector Machine (OCSVM), Local Outlier Factor (LOF), Kernel Density Estimation (KDE), Cetroid (CEN), Isolation Forest (IF) and Elliptic Envelope (EE).

3.3.1. One-Class Support Vector Machine (OCSVM)

OCSVM [49] is a particular case of SVM. During the training process, OCSVM tries to find a hyperplane that separates normal network instances and the origin as far as possible. Its core idea is to build the separation between normal and attack network regions as clearly as possible in the feature space, where anomalous data points appear closer to the origin.

3.3.2. Local Outlier Factor (LOF)

LOF [50] is a commonly used technique in anomaly detection tasks. LOF identifies anomalous data points by measuring their local deviation from surrounding neighbors. LOF is defined based on local density, in which the distance of K neighbors is used to estimate the density. Additionally, LOF identifies anomalous data points by assuming their local density is generally lower than the local density of normal data points.

3.3.3. Kernel Density Estimation (KDE)

KDE [51] is an effective method used for anomaly detection. KDE is a non-parametric method used to estimate the probability density function of data points based on kernels and weights. Essentially, the performance of the KDE classifier depends mainly on the kernel function and bandwidth selection.

3.3.4. Centroid (CEN)

CEN is a method based on the assumption that normal network data points are distributed in a hyperspherical shape [48]. It will then calculate the distance from the center of the hypersphere to determine whether it is a normal or anomalous data point. The distance threshold is a vital hyperparameter that determines the accuracy of the CEN one-class classifier.

3.3.5. Isolation Forest (IF)

IF is an effective, fast anomaly detection method based on binary tree mechanism [52]. IF operates on the assumption that abnormal data points are rare and contain different characteristics in comparison with normal ones. IF has linear time computational complexity, low memory requirements, and can work with large datasets.

3.3.6. Elliptic Envelope (EE)

EE is a method of identifying anomalous data points based on the shape of known distributions [53]. This method assumes the data follows a Gaussian distribution. Specifically, it will try to find the hyper-ellipse to contain most of the normal data points and identify points that lie outside the hyper-ellipse as anomalies. The accuracy of the EE classifier depends on estimating the size and shape of the hyper-ellipse.

4. PROPOSED METHODOLOGY

We aim to find a latent representation space of normal network data to support OCC classifiers that further improves anomaly detection performance. To overcome the limitations of recent state-of-the-art methods discussed previously, we propose two novel deep learning-based models to explore the more meaningful and concise features of normal network behavior. The motivations for proposing two different models in this paper are firstly, to validate that any feature quality improvements offered by our underlying method of dual refinement are consistent across different architectures. Secondly, we wanted to explore how the different architecture influence the latent spaces generated using our proposed method.

Our proposed models (DNCAE and DCHAE) are both based on the idea of combining of two hierarchically connected DAEs (using two different architectures as outlined above) with the K-means clustering algorithm integrated into the latent representation space of a DAE to explore the clustering structure of normal samples in this space. An optimal arrangement of normal network data points on the first latent space will be formed with K-means' support. Then, the second DAE will further compress this arrangement to minimize the normal space region, thereby clarifying the border between the normal and abnormal data areas. As a result, powerful and expressive latent features will be learned to better support OCC classifiers. In addition, the dimensionality in the latent space will be smaller than the original data, meaning the latent space is a more pure, condensed, and concise representation of the original data. The training of DAE and K-means is carried out in a co-training manner, which will be presented in detail in subsection 4.1. The two proposed models, DNCAE and DCHAE, will be presented in detail in subsections 4.2, 4.3, respectively.

4.1. Clustering in Latent Representation Space

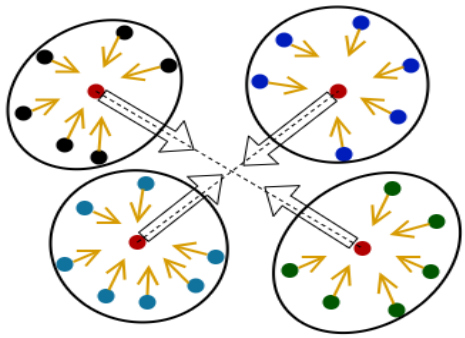


Figure 1: Clustering at Latent Representation Space

Previous DAE models used for anomaly detection only push normal network samples towards one cluster in the latent representation space. We argue that the underlying architecture of normal network data will consist of many sub-clusters. The reason for this phenomenon is that normal network observations are generated from various common user behaviors and many different services, applications and protocols. As a result, these data instances have not only the typical characteristics of normal network data but also the unique signatures of each sub-cluster. Therefore, to explore the optimal arrangement of normal network data points in the latent space, we integrate the K-means clustering algorithm into the bottleneck layer of a DAE. The objective function in this clustering training process is called the clustering loss and has the following form:

$$\Omega(x_i, h) = \lambda_1 \frac{1}{n} \sum_{1}^{n} \left\| \mathbf{F}^{(t)}(x_i) - c_i^* \right\|^2 + \lambda_2 \frac{1}{k} \sum_{1}^{k} \left\| c_j^t - c_0^t \right\|^2 \quad (10)$$

$$c_i^* = \underset{c_j^{t-1}}{argmin} \left\| \mathbf{F}^{(t)}(x_i) - c_j^{t-1} \right\|^2 \quad (11)$$

$$c_j^t = \frac{\sum_{x_i \in C_j^{t-1}} \mathbf{F}^t(x_i)}{\left| C_j^{t-1} \right|} \quad (12)$$

$$c_0^t = \frac{1}{k} \sum_{1}^{k} c_j^t \quad (13)$$

Where $\mathbf{X} = \{\mathbf{x}_i | i = 1, 2, 3, ...n\}$ is training set with n samples; $\mathbf{F}^{(t)(.)}$ is Encoder function of a DAE at i^{th} iteration; c_i^* is the the closest sub-cluster center of $\mathbf{F}^t(x_i)$ sample at t iteration; c_j^{t-1} and c_j^t are j^{th} sub-cluster centers at $(t-1)^{th}$ and t^{th} iterations, respectively; c_0^t is expectation over all k sub-cluster centers at t^{th} iteration; $\Omega(x_i, h)$ is a clustering loss containing two terms. The first term is to minimize the distance from the data points to their corresponding sub-cluster centers. The second term is to force the sub-cluster centers to move closer to each other. λ_1 and λ_2 are the coefficients to trade off these two loss components in the clustering objective function. We have designed a Figure 1 to support the description of the clustering procedure at the latent representation in two-dimensional space. Specifically, data instances in the same sub-cluster are pushed to their center, and the cluster centers are also forced closer together within this space.

4.2. Deep Nested Clustering Auto-Encoder (DNCAE)

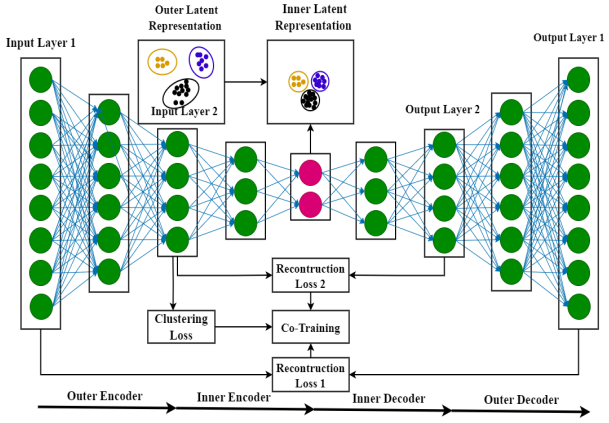


Figure 2: Deep Nested Clustering Auto-Encoder (DNCAE)

We aim to build a DL-based latent representation learning method for normal network data, to produce meaningful and highly separable features, thereby better supporting the use of OCC classifiers. Our model is expected to overcome the limitations of recent state-of-the-art models [23], [24]. To do this, the following goals must be achieved simultaneously: i) our proposed model requires the capability to find latent, outstanding features to distinguish between normal and abnormal network data more efficiently; and ii) our proposed model must minimize the spatial region of normal network data in the latent space so that the decision boundary between normal and abnormal data is more apparent. Furthermore, it is based on the observation that data has an intrinsic clustering structure [54]. Consequently, although the network data for training anomaly-based network attack detectors is just normal data, it will also

have an internal clustering structure. This phenomenon can be explained as normal network data being generated from many different types of common user behavior and a diversity of protocols, applications, and devices.

The DNCAE model includes two nested DAEs: Outer DAE and Inner DAE. The Outer DAE's latent layer is used as the input layer of the Inner DAE. The output of the Inner DAE is the input of the Decoder part of the Outer DAE. Additionally, the K-means clustering algorithm is integrated into the latent layer of Outer DAE, as described in the previous section 4.1 before entering into the Encoder part of Inner DAE. The general architecture of the model is illustrated in Figure 2. DNCAE is expected to combine the powerful feature learning ability of the two DAE models, with the fast and effective clustering of K-means. The task of the Decoder part of Outer DAE is to map the original normal network data to a richer and more abstract feature space. Here, K-means will assist in discovering an optimal clustering structure, so that data points in the same cluster will be forced closer to the corresponding center, and at the same time, the cluster centers will also move closer together. After that, the Encoder part of Inner DAE will continue to compress this space to minimize this normal region's hypervolume and distill the core, most quintessential latent features of normal data points. The DNCAE training process aims to optimize the three-component objective function as follows:

$$\mathbf{L}_{\text{DNCAE}} = \alpha_1 \mathcal{L}_{\text{Outer}}\left(\mathbf{X}, \hat{\mathbf{X}}\right) + \alpha_2 \mathcal{L}_{\text{Inner}}\left(\mathbf{H}, \hat{\mathbf{H}}\right) + \alpha_3 \Omega(\mathbf{H}) \quad (14)$$

Here, the first and second components are the reconstruction error of Outer DAE and Inner DAE, respectively, and the third component is the clustering loss $\Omega(\mathbf{H})$ calculated by the expression 10. In our work, $\mathcal{L}_{\text{Outer}}\left(\mathbf{X}, \hat{\mathbf{X}}\right)$ and $\mathcal{L}_{\text{Inner}}\left(\mathbf{H}, \hat{\mathbf{H}}\right)$ have the following particular forms:

$$\mathcal{L}_{\text{Outer}}\left(\mathbf{X}, \hat{\mathbf{X}}\right) = \mathcal{L}_{\text{MSE}}\left(\mathbf{X}, \hat{\mathbf{X}}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}_i - \hat{\mathbf{x}}_i\right)^2 \quad (15)$$

$$\mathcal{L}_{\text{Inner}}\left(\mathbf{H}, \hat{\mathbf{H}}\right) = \mathcal{L}_{\text{MSE}}\left(\mathbf{H}, \hat{\mathbf{H}}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{h}_i - \hat{\mathbf{h}}_i\right)^2 \quad (16)$$

$\mathbf{X} = \{\mathbf{x}_i | i = 1, 2, 3, ...n\}$ is training data of n normal network instances. $\hat{\mathbf{X}} = \{\hat{\mathbf{x}}_i | i = 1, 2, 3, ...n\}$ is the reconstructed version of $\mathbf{X}$ at the output layer of Outer DAE; $\mathbf{H} = \{\mathbf{h}_i | i = 1, 2, 3, ...n\}$ is a latent representation of Outer DAE and also the input for the Encoder part of Inner DAE; $\hat{\mathbf{H}} = \{\hat{\mathbf{h}}_i | i = 1, 2, 3, ...n\}$ is the reconstructed version of $\mathbf{H}$ at the output layer of Inner DAE and also an input for the Decoder part of Outer DAE; Coefficients α_1, α_2, and α_3 are used to trade-off three loss terms in the objective function. The process of optimizing the objective function of DNCAE is carried out in a co-training manner, in which the weights and biases are updated using the Stochastic Gradient Descent (SGD) technique [39].

4.3. Deep Clustering Hierarchical Auto-Encoder (DCHAE)

The DCHAE model includes two DAEs, which are called Big DAE and Small DAE; its overall architecture is illustrated in Figure 3. These two DAEs are connected to each other by

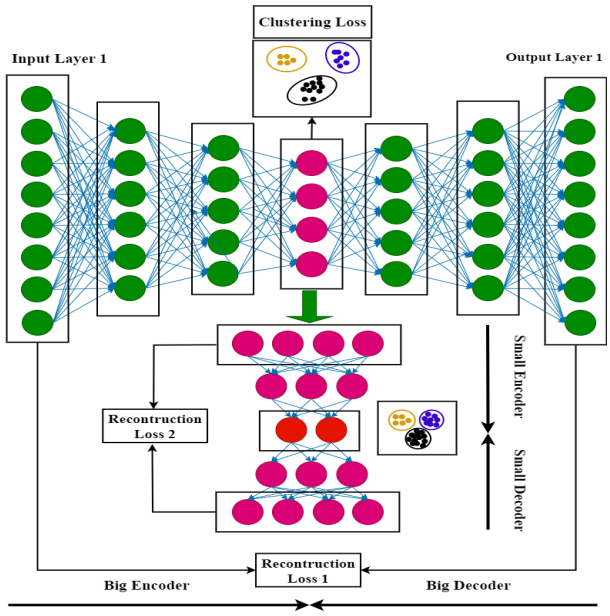


Figure 3: Deep Clustering Hierarchical Auto-Encoder (DCHAE)

the latent layer at the bottleneck of Big DAE being used as the input layer of Small DAE. The output layer of the Small DAE is left free and not connected to the Big DAE as was the case in DNCAE. At the bottleneck of Big DAE, we also integrate the K-means clustering algorithm as described in the section 4.1 to find the optimal distribution of normal network data points according to the appropriate structure of sub-clusters. Then, Small DAE will use this space as input data to distill the core, most quintessential latent features of normal data points.

The core difference between DNCAE and DCHAE is that in DNCAE, the Outer DAE, and Inner DAE are designed to be nested (Inner DAE is completely inside Outer DAE). However, in DCHAE, Big DAE and Small DAE are designed in two separate branches. The common point of these two branches is that the bottleneck layer of Big DAE is also the input of Small DAE.

The objective function in the training process DCHAE has the following form:

$$\mathbf{L}_{\text{DCHAE}} = \beta_1 \mathcal{L}_{\text{Big}}\left(\mathbf{X}, \hat{\mathbf{X}}\right) + \beta_2 \mathcal{L}_{\text{Small}}\left(\mathbf{H}, \hat{\mathbf{H}}\right) + \beta_3 \Omega(\mathbf{H}) \quad (17)$$

$$\mathbf{L}_{\text{DCHAE}} = \beta_1 \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}_i - \hat{\mathbf{x}}_i\right)^2 + \beta_2 \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{h}_i - \hat{\mathbf{h}}_i\right)^2 + \beta_3 \Omega(\mathbf{H}) \quad (18)$$

Here, the first two components of the objective function are the reconstruction errors of Big DAE and Small DAE, respectively. The third component of the objective function is the clustering error determined as described in Section 4.1.

$\mathbf{X} = \{\mathbf{x}_i | i = 1, 2, 3, ...n\}$ is a training set of n normal network observations. $\hat{\mathbf{X}} = \{\hat{\mathbf{x}}_i | i = 1, 2, 3, ...n\}$ is reconstructed version of $\mathbf{X}$ at the output layer of Big DAE; $\mathbf{H} = \{\mathbf{h}_i | i = 1, 2, 3, ...n\}$ is a latent representation of Big DAE and also the input for Encoder part of Small DAE; $\hat{\mathbf{H}} = \{\hat{\mathbf{h}}_i | i = 1, 2, 3, ...n\}$ is a reconstructed version of $\mathbf{H}$ at the output layer of Small DAE; β_1, β_2, and β_3 are coefficients used to trade-off three loss terms in the objective function.

Both DNCAE and DCHAE are trained in a co-training manner, to explore hierarchical latent data representation spaces from only normal network data, to overcome the limitations of contemporary state-of-the-art models [23], [24].

The training process of our proposed models is presented in Algorithm 1.

Algorithm 1 Training Process of DNCAE and DCHAE

Input: A given dataset X, the number of clusters k, two sets of parameters Θ_1 and Θ_2 for two DAE in each of our proposed models, the maximum number of iterations T
Randomly choose k cluster centers (C^o)
Initialize model parameters Θ_1 and Θ_2
$t = 1$
repeat
    Train DNCAE and DCHAE on X to optimize the loss functions (14) and (17), respectively
    Classify X into k clusters
    Update new k cluster centers (C^t) via formula (12)
    $t = t + 1$
until an early-stopping criterion is met or $t < T$
Output: DNCAE and DCHAE models

The algorithm begins with the dataset $\mathbf{X}$ and a predefined number of clusters k. It also requires two sets of parameters $\mathbf{\Theta}_1$ and $\mathbf{\Theta}_2$ for the two DAEs in the proposed models, as well as a maximum number of iterations T . The process starts by randomly initializing the cluster centers and setting the parameters for both DAEs. The iteration counter is initialized to 1. During each iteration, the DNCAE and DCHAE models are trained on the dataset $\mathbf{X}$ to minimize their respective loss functions. The trained models are then used to classify the dataset $\mathbf{X}$ into k clusters. The cluster centers are updated using the specified formula 12 based on the models' current state. The iteration counter is incremented, and the process is repeated until an early-stopping criterion is met or the maximum number of iterations is reached.

4.4. General Flow of Proposed Approach

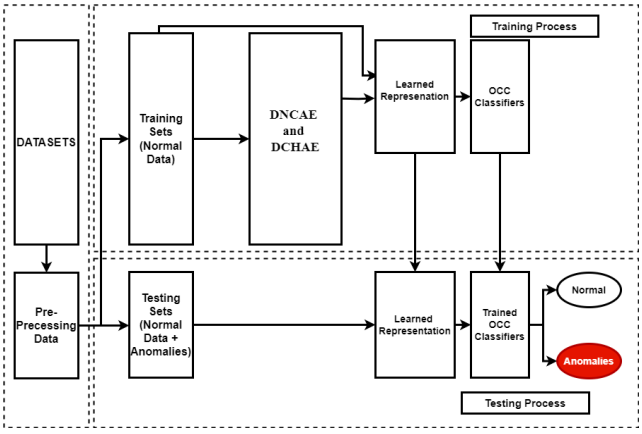


Figure 4: General Flow of Proposed Approach

Our proposed DNCAE (4.2) and DCHAE (4.3) approaches are based on the hierarchical latent representation learning model. Fundamentally, DNCAE and DCHAE have harmoniously combined the powerful and elegant latent representation learning ability of DAE models with the efficiency and fast clustering capability of K-means. As a result, DNCAE and DCHAE offer a richer, condensed, and distilled data representation space and simultaneously minimize the super-volume of normal network regions at latent space. Therefore, it better supports the performance of OCC classifiers and overcomes the inherent limitations of previous works. To evaluate the effectiveness of DNCAE and DCHAE, a set of OCC classifiers is used, including: One-Class Support Vector Machine (OC-SVM with Linear, Poly, Rbf, and Sigmoid Kernels), Local Outlier Factor (LOF), Kernel Density Estimation (KDE), Cetroid (CEN), Isolation Forest (IF) and Elliptic Envelope (EE). In this work, OCC classifiers are trained using the learned representation produced by DNCAE and DCHAE during the training process. The general working flow of our proposed method is illustrated in Figure 4. More specifically, our solution is divided into two training stages: The first stage is learning the latent representing space from only normal network instances by training DNCAE and DCHAE models. Then, in the second stage, OCC classifiers are trained using the learned features from the previous stage.

5. EXPERIMENTS

In this section, we will briefly present the benchmark datasets used to evaluate the performance of our proposed models and compare them with baselines and state-of-the-art methods. After that, the configurations, hyperparameters, and programming environment used for the experimental implementation will be described.

5.1. Datasets

To evaluate the performance of the proposed model, we conducted experiments using the following benchmark datasets: NSL-KDD, UNSW-NB15, CIC-IDS-2017, CSE-CIC-IDS-2018, and CTU-13.

- **NSL – KDD** : The NSL-KDD dataset is an improved version of the KDD99 dataset that addresses its inherent limitations, which are thoroughly analyzed in [55]. Although NSL-KDD still has many shortcomings, it is still one of the most widely used datasets in the research community to evaluate proposed methods for building intrusion detection systems. The NSL-KDD dataset includes normal traffic and 22 different attack patterns, which are divided into four sub-classes, including Probe, Denial of Service (DoS), Remote to Local (R2L), and User to Root (U2R). Each data record has 41 features. Categorical features, including protocol type, service, and flag are preprocessed using a one-hot-encoding technique before training the deep learning model. In addition, the NSL-KDD set is predivided into two subsets for training and testing purposes: $KDDTrain^+$ and $KDDTest^+$, respectively.

- **UNSW – NB15** : The UNSW-NB15 was published at the Cyber Range Lab in New South Wales in 2015 [56]. This data set consists of normal network instances and synthesized cyber-attack behaviors generated from the IXIA PerfectStorm system. The UNSW-NB15 was envisaged to overcome the limitations of the KDD98, KDDCUP99 and NSLKDD data sets, with many new attack data samples reflecting the current state of cyberspace. Specifically, this dataset includes normal network data and nine different attack patterns, in which each record has 49 features. Categorical attributes are also preprocessed using a one-hot encoding technique before being pushed into the training process.

- **CIC – IDS – 2017** : The CIC-IDS-2017 dataset was published at the Canadian Institute of Cybersecurity (CIC) in 2017 [57]. CIC-IDS-2017 includes benign network observations and many newly updated attack samples that resemble real network environments. Diverse attack data types include DoS, DDoS, Brute Force, XSS, SQL Injection, Infiltration, Port Scan, and Botnets. The data set includes eight different scenarios, in which each network data point has 78 features.

- **CSE – CIC – IDS – 2018** : The CSE-CIC-IDS-2018 is the output of a collaborative project between the Communications Security Establishment (CSE) and The Canadian Institute for Cybersecurity (CIC) in 2018 [57]. This dataset includes common network data streams and seven different attack scenarios, including Brute-force, Heartbleed, Bot, DoS, DDoS, Web attack, and infiltration [58]. The infrastructure to deploy the attack includes 50 machines, and the target's network topology includes five different departments with 420 personal computers and 30 servers. Network data generated from attack situations is intercepted and stored as Pcap files. Then, these Pcap files are processed using the CICFlowMeter-V3 tool; each data point has 78 features.

- **CTU13** : CTU13 is a dataset consisting of Botnet traffic published at CTU University, Czech Republic, in 2011 [59]. In fact, this dataset is a mixture of normal network data traffic, background traffic, and Botnet traffic in thirteen scenarios) with different Botnet samples. In these scenarios, the categorical attributes are protocol, sTos, and dTos, which are preprocessed using one-hot encoding techniques before input to the training process.

The reasons for selecting these datasets to evaluate the performance of the proposed models are: i) NSL-KDD and UNSW-NB15 are commonly-used NIDS datasets, which aids comparability; ii) CIC-IDS-2017 and CSE-CIC-IDS-2018 are state-of-the-art datasets containing new attack patterns similar to real network environments; iii) CTU13 contains popular attacks using Botnets against IoT environments.

During the training process, the NSL-KDD and UNSW-NB15 data sets have been pre-divided into training and testing datasets. We use only normal network data in the training set for training purposes, while both normal and abnormal data in the testing set are used to verify the quality of the proposed model. For the CIC-IDS-2017, CSE-CIC-IDS-2018, and CTU13 data sets, we take 60 % of the normal network data for training, while the remaining 40 % of the normal network data and all anomalies are reserved for testing purposes.

Table 1 provides a summary of the datasets used in the experiments presented in this paper. It also details the which specific subsets were utilised from the larger datasets such as CIC-IDS2017 and CTU13.

5.2. Experimental Settings

In this work, we conduct experiments to verify the performance of proposed models DNCAE and DCHAE to improve the quality of anomaly-based cyber-attack detectors. To clarify the superiority of the proposed models DNCAE and DCHAE, the experimental results are compared with the following:

1. Baseline models, including DAE+OCCs and DCAE+OCCs [60]

2. State-of-the-art models, including PCADCAE +OCCs [24], SAE+OCCs and DVAE+OCCs [23]

Additionally, a number of additional experiments are conducted to study various aspects of the proposed model and its results, these include:

- Investigating the influence of the number of clusters in the K-means algorithm on the performance of models DNCAE and DCHAE for anomaly-based cyber-attack detection.

- Comparing the training and testing times of the proposed models against previous models to evaluate the ability to deploy into real-world situations.

- Determining the impact of the loss components in the objective function on the performance of the proposed models.

- Visualizing the latent representation of the proposed models in 2-dimensional space to clarify the experimental results and observe the distribution of samples more clearly.

5.2.1. Deep Auto-Encoder parameters

The architecture of the DNCAE model is designed as follows: The Encoder part of Outer DAE and Inner DAE has four layers. The output of the Encoder part of the Outer DAE is the input of the Encoder part of the Inner DAE. Both Outer DAE and Inner DAE are designed with a symmetrical architecture between the Encoder and Decoder parts. The number of dimensions at the bottleneck of Outer DAE and Inner DAE is calculated using the formula $d_h = [1 + \sqrt{D}]$ as in [23], [24]. Where D is the input dimension of the Encoder parts, d_h is the dimension in the latent space.

The architecture of DCHAE is designed similarly to DNCAE, wherein, the Encoder parts of Big DAE and Small

Table 1: Benchmark datasets for evaluating the proposed models

No	Datasets	Dimension	Normal Training	Normal Testing	Anomaly Testing	Type of Attack
1	NSL-KDD	122	67343	9711	12833	**DoS** (Back, Land, Neptune, Pod, Smurf, Teardrop, Apache2, Udpstorm, Processtable, Worm), **Probe**(Satan, Ipsweep, Nmap, Portsweep, Mscan, Saint), **U2L**(Guess_Password , Ftp_write , Imap , Phf , Multihop,Warezmaster ,Warezclient, Spy,Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named), **U2R** (Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps)
2	UNSW-NB15	196	55999	33999	45332	Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, Worms
3	CIC-IDS-2017 Tuesday	78	260830	171244	13835	SSH-Patator, FTP-Patator
4	Wednesday	78	264016	176015	252672	DoS Hulk, DoS GoldenEye, DoS Slowloris, DoS Slowhttptest, Heartbleed
5	Thursday	78	100910	67276	2180	Web AttackBrute Force, Web Attack-Sql Injection,Web Attack-XSS
6	Friday-BOT	78	113500	75567	1966	BOT
7	Friday-PortScan	78	48859	78678	158930	PortScans (sS, sT, sF, sX, sN, sP, sV, sU, sO, sA, sW, sR, sL and B)
8	Friday-DDoS	78	58630	39088	128027	DDoS
9	CSE-CIC-IDS-2018 Wed-14-02-2018	78	400575	267051	380949	FTP-BruteForce, SSH-Bruteforce
10	Fri-16-02-2018	78	268063	178709	601802	DoS-SlowHTTPTest, DoS-Hulk
11	Wed-21-02-2018	78	216499	144334	687742	DDOS-LOIC-UDP, DDOS-HOIC
12	Thurs-22-02-2018	78	628927	419286	362	Brute Force -Web, Brute Force -XSS, SQL Injection
13	Fri-23-02-2018	78	628805	419204	566	Brute Force -XSS, Brute Force -Web, SQL Injection
14	Wed-28-02-2018	78	326520	217680	68871	Infiltration
15	Thursday-01-03-2018	78	142822	95215	93063	Infiltration
16	Friday-02-03-2018	78	457430	304954	286191	Bot
17	CTU13-06	38	2997	4496	4630	Bot Menti, PS
18	CTU13-07	34	670	1006	63	Bot Sogou, HTTP
19	CTU13-08	40	29128	43693	6127	Bot Murlo, PS
20	CTU13-09	41	11986	17980	184987	Bot Neris, IRC, SPAM, CF, PS
21	CTU13-10	38	6338	9508	106352	Bot Rbot, IRC, DDoS, US
22	CTU13-12	36	3050	4577	2168	Bot NSIS.ay, P2P botnet
23	CTU13-13	40	12775	19163	40003	Bot Virus, SPAM, PS, HTTP

DAE also include four layers, the way to calculate the number of dimensions at the bottleneck layer is still the same as in DNCAE.

For both DNCAE and DCHAE, we use the Xavier initialization technique for the weights and biases of the models to speed up the convergence process. The activation function used is Tanh. The batch size is set to 256, the Adadelta optimization algorithm is used, and the learning rate is set to 0.01.

To prevent overfitting during our experiments, we implemented early stopping and L2 regularization techniques [23]. The L2 regularization helps to decrease the model's complexity, enhancing its generalization capability for unseen data. The coefficient of L2 regularization is set to 0.05. Additionally, the training process terminates if the loss function value does not improve by an absolute value of 0.001 within 5 epochs (patience).

5.2.2. OCC parameters

In terms of OC-SVM, the four kernels used are: 'linear', 'poly', 'rbf', and 'sigmoid', respectively. The degree of the polynomial kernel function for the 'poly' kernel is set to 3. In addition, other hyperparameters are set to the same default. The number of base estimators in the ensemble is set to 100 for the IF classifier. The value for the contamination hyperparameter is set up as 'auto'. All hyperparameters are set to default for the EE classifier. The number of neighbors is set to 10 for the LOF classifier. The value for the contamination hyperparameter is set up to 'auto' as for IF. For KDE, the threshold value is set at 0.95, the kernel chosen is 'Gaussian', and the metric used is 'euclidean'. The distance threshold of the CEN classifier is set up to 0.95, and the Euclidean metric is used.

5.2.3. Evaluation Metrics and Environmental Infrastructure

We use the area under the resulting ROC curve (AUC) as the primary metric for comparing performance between baselines, state-of-the-art models, and proposed models. This is a metric widely used in network anomaly detection works [23], thus enabling better comparisons.

All experiments were conducted under the same environmental conditions. Experiments were implemented in Python 3.10 using the Pytorch framework and executed in an Anaconda en-

Table 2: AUCs of DAE+OCCs, DCAE+OCCs, PCADCAE+OCCs, SAE+OCCs, DVAE+OCCs, DNCAE+OCCs, DHCAE+OCCs models on NSL-KDD, UNSW-NB15 and six scenarios of CIC-IDS-2017

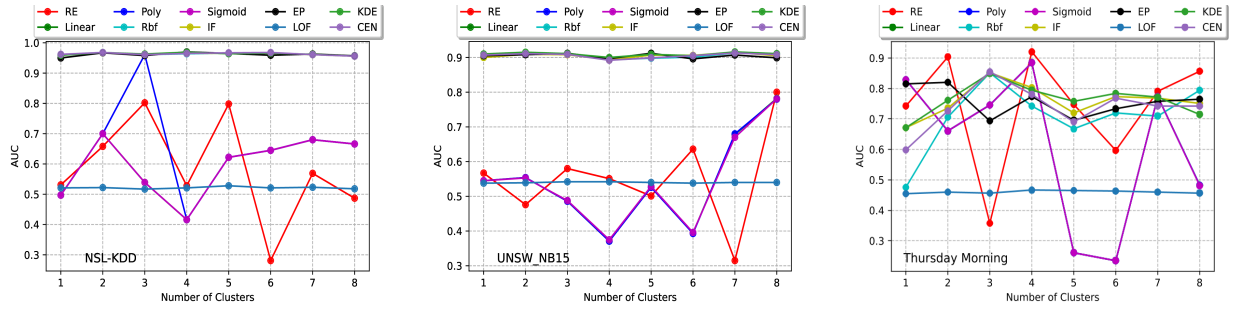
Latent Representation	**One-Class Classifiers**	**Datasets**							
		NSL-KDD	UNSW-NB15	Tuesday	Wednesday	Thurday Morning	Friday BOT	Friday PortScan	Friday DDoS
DAE	RE	0.623	0.622	0.711	0.710	0.534	0.523	0.612	0.637
	OCSVM Linear	0.869	0.615	0.490	0.082	0.584	0.619	0.195	0.181
	OCSVM Poly	0.871	0.618	0.311	0.156	0.461	0.356	0.451	0.264
	OCSVM Rbf	0.937	0.823	0.435	0.751	0.465	0.517	0.246	0.610
	OCSVM Sigmoid	0.871	0.619	0.493	0.213	0.463	0.256	0.082	0.343
	IF	0.951	0.834	0.747	0.873	0.588	0.676	0.571	0.793
	EE	0.936	0.789	0.692	0.889	0.797	0.671	0.729	0.815
	LOF	0.513	0.536	0.449	0.469	0.450	0.471	0.458	0.452
	KDE	0.946	0.857	0.740	0.879	0.643	0.632	0.651	0.773
	CEN	0.943	0.794	0.452	0.869	0.621	0.704	0.276	0.598
DCAE	RE	0.647	0.630	0.580	0.694	0.320	0.741	0.670	0.808
	OCSVM Linear	0.455	0.380	0.641	0.901	0.235	0.319	0.173	0.483
	OCSVM Poly	0.455	0.380	0.641	0.901	0.235	0.320	0.173	0.483
	OCSVM Rbf	0.967	0.888	0.508	0.897	0.677	0.677	0.761	0.871
	OCSVM Sigmoid	0.455	0.380	0.641	0.901	0.235	0.319	0.173	0.483
	IF	0.966	0.892	0.674	0.905	0.696	0.725	0.755	0.893
	EE	0.957	0.908	0.361	0.890	0.802	0.752	0.824	0.851
	LOF	0.521	0.537	0.468	0.466	0.459	0.476	0.465	0.455
	KDE	0.970	0.893	0.657	0.897	0.708	0.706	0.766	0.884
	CEN	0.967	0.878	0.639	0.877	0.669	0.707	0.755	0.880
PCADCAE	RE	0.756	0.589	0.619	0.579	0.436	0.523	0.401	0.657
	OCSVM Linear	0.768	0.746	0.727	0.444	0.072	0.605	0.406	0.270
	OCSVM Poly	0.748	0.735	0.727	0.444	0.072	0.605	0.406	0.270
	OCSVM Rbf	0.965	0.846	0.750	0.904	0.629	0.667	0.650	0.848
	OCSVM Sigmoid	0.768	0.746	0.428	0.444	0.072	0.605	0.406	0.300
	IF	0.963	0.843	0.732	0.904	0.813	0.639	0.657	0.849
	EE	0.962	0.852	0.793	0.807	0.783	0.711	0.728	0.885
	LOF	0.514	0.531	0.493	0.483	0.473	0.497	0.455	0.467
	KDE	0.962	0.845	0.743	0.907	0.819	0.700	0.674	0.860
	CEN	0.964	0.846	0.746	0.905	0.830	0.702	0.655	0.846
SAE	RE	0.650	0.584	0.459	0.619	0.583	0.558	0.600	0.552
	OCSVM Linear	0.726	0.727	0.718	0.346	0.207	0.602	0.403	0.290
	OCSVM Poly	0.942	0.733	0.652	0.745	0.631	0.605	0.793	0.598
	OCSVM Rbf	0.957	0.841	0.748	0.900	0.618	0.664	0.645	0.840
	OCSVM Sigmoid	0.748	0.742	0.438	0.429	0.208	0.601	0.405	0.304
	IF	0.935	0.830	0.711	0.881	0.671	0.559	0.812	0.619
	EE	0.941	0.829	0.705	0.720	0.675	0.653	0.804	0.752
	LOF	0.519	0.540	0.451	0.470	0.451	0.482	0.449	0.458
	KDE	0.943	0.827	0.715	0.886	0.700	0.654	0.803	0.613
	CEN	0.942	0.832	0.744	0.715	0.621	0.654	0.769	0.582
DVAE	RE	0.396	0.456	0.616	0.621	0.578	0.572	0.265	0.280
	OCSVM Linear	0.785	0.646	0.252	0.164	0.266	0.448	0.270	0.381
	OCSVM Poly	0.780	0.622	0.370	0.211	0.265	0.435	0.264	0.398
	OCSVM Rbf	0.812	0.655	0.560	0.789	0.726	0.739	0.740	0.588
	OCSVM Sigmoid	0.787	0.668	0.260	0.175	0.265	0.455	0.288	0.374
	IF	0.817	0.672	0.493	0.803	0.483	0.473	0.457	0.531
	EE	0.816	0.736	0.494	0.544	0.459	0.453	0.448	0.737
	LOF	0.530	0.541	0.453	0.464	0.448	0.476	0.455	0.448
	KDE	0.954	0.881	0.701	0.912	0.681	0.698	0.757	0.843
	CEN	0.955	0.743	0.602	0.893	0.650	0.545	0.497	0.632
DNCAE	RE	**0.802**	**0.800**	0.680	0.436	**0.920**	0.644	**0.922**	0.748
	OCSVM Linear	0.700	*0.782*	**0.925**	0.398	**0.885**	**0.759**	**0.953**	*0.966*
	OCSVM Poly	**0.961**	*0.782*	**0.925**	0.398	**0.885**	**0.759**	**0.953**	*0.966*
	OCSVM Rbf	**0.969**	**0.913**	0.711	**0.938**	*0.742*	**0.746**	*0.816*	**0.927**
	OCSVM Sigmoid	0.700	*0.782*	**0.925**	0.398	**0.885**	**0.759**	**0.953**	*0.966*
	IF	*0.967*	**0.915**	*0.753*	**0.936**	0.803	0.712	**0.838**	*0.917*
	EE	**0.970**	**0.912**	0.708	**0.917**	0.774	0.649	0.798	**0.951**
	LOF	0.528	*0.542*	0.476	*0.471*	0.466	*0.491*	*0.467*	*0.461*
	KDE	*0.969*	**0.916**	0.735	**0.929**	0.795	0.700	*0.804*	**0.937**
	CEN	*0.968*	*0.913*	**0.788**	**0.933**	0.799	**0.722**	*0.812*	**0.923**
DCHAE	RE	0.737	*0.747*	**0.899**	**0.869**	*0.888*	**0.794**	*0.841*	0.694
	OCSVM Linear	0.758	**0.796**	*0.850*	**0.902**	*0.766*	*0.735*	*0.899*	**0.976**
	OCSVM Poly	0.761	**0.796**	*0.801*	**0.902**	*0.769*	*0.732*	*0.895*	**0.976**
	OCSVM Rbf	**0.969**	*0.900*	**0.780**	*0.912*	**0.808**	0.700	**0.853**	*0.918*
	OCSVM Sigmoid	**0.875**	**0.796**	*0.823*	**0.902**	*0.768*	*0.737*	*0.897*	**0.976**
	IF	**0.969**	*0.898*	**0.761**	*0.922*	**0.815**	**0.739**	*0.827*	**0.918**
	EE	**0.970**	*0.910*	**0.795**	*0.898*	**0.882**	**0.752**	**0.855**	*0.917*
	LOF	**0.530**	**0.584**	0.472	**0.483**	**0.548**	0.483	**0.470**	0.459
	KDE	**0.970**	**0.916**	**0.749**	*0.926*	0.783	**0.754**	**0.828**	*0.923*
	CEN	**0.969**	**0.914**	*0.775*	*0.927*	**0.831**	*0.708*	**0.844**	*0.916*

Table 3: AUCs of DAE+OCCs, DCAE+OCCs, PCADCAE+OCCs, SAE+OCCs, DVAE+OCCs, DNCAE+OCCs, DHCAE+OCCs models on eight scenarios of CSE-CIC-IDS-2018

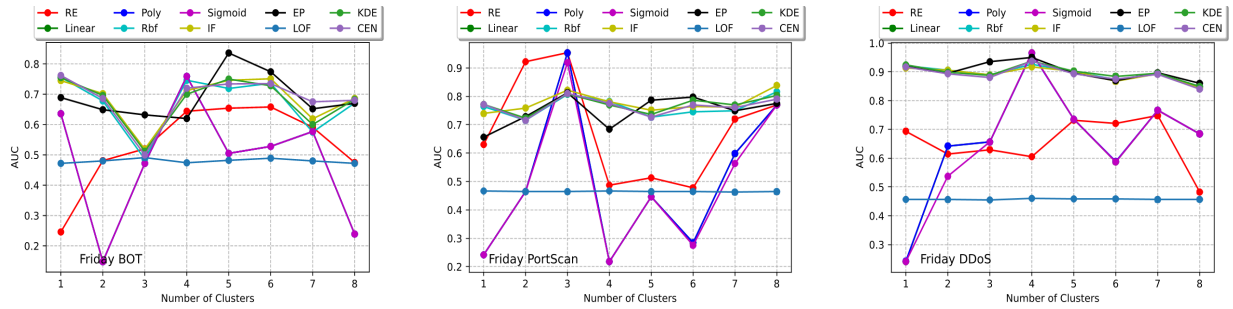
Latent Representation	One-Class Classifiers	Datasets							
		14-02-2018	16-02-2018	21-02-2018	22-02-2018	23-02-2018	28-02-2018	01-03-2018	02-03-2018
DAE	RE	0.446	0.798	0.798	0.570	0.415	0.491	0.471	0.454
	OCSVM Linear	0.143	0.988	0.998	0.395	0.783	0.505	0.550	0.505
	OCSVM Poly	0.206	0.988	0.998	0.450	0.553	0.512	0.541	0.497
	OCSVM Rbf	0.734	0.979	0.979	0.563	0.620	0.474	0.457	0.649
	OCSVM Sigmoid	0.151	0.988	0.998	0.405	0.801	0.520	0.545	0.505
	IF	0.737	0.995	0.995	0.561	0.504	0.473	0.573	0.679
	EE	0.810	0.995	0.995	0.553	0.358	0.464	0.593	0.762
	LOF	0.448	0.496	0.496	0.579	0.557	0.504	0.506	0.451
	KDE	0.809	0.963	0.963	0.520	0.543	0.470	0.534	0.663
	CEN	0.660	0.979	0.979	0.560	0.653	0.500	0.466	0.627
DCAE	RE	0.468	0.837	0.824	0.646	0.348	0.489	0.486	0.505
	OCSVM Linear	0.979	0.757	0.759	0.799	0.492	0.533	0.466	0.482
	OCSVM Poly	0.979	0.757	0.759	0.799	0.491	0.533	0.466	0.482
	OCSVM Rbf	0.943	0.965	1.000	0.571	0.435	0.472	0.578	0.471
	OCSVM Sigmoid	0.979	0.757	0.799	0.799	0.492	0.533	0.466	0.482
	IF	0.951	0.964	1.000	0.548	0.419	0.473	0.581	0.453
	EE	0.922	0.479	1.000	0.564	0.446	0.473	0.575	0.579
	LOF	0.446	0.501	0.504	0.547	0.548	0.505	0.500	0.454
	KDE	0.957	0.958	1.000	0.562	0.457	0.472	0.577	0.483
	CEN	0.948	0.965	1.000	0.571	0.447	0.473	0.576	0.478
PCADCAE	RE	0.647	0.625	0.437	0.734	0.487	0.494	0.490	0.526
	OCSVM Linear	0.726	0.424	0.473	0.424	0.564	0.478	0.603	0.552
	OCSVM Poly	0.726	0.424	0.473	0.424	0.564	0.478	0.603	0.552
	OCSVM Rbf	0.646	0.528	0.528	0.555	0.499	0.477	0.552	0.603
	OCSVM Sigmoid	0.726	0.423	0.473	0.424	0.564	0.478	0.603	0.552
	IF	0.700	0.612	0.732	0.562	0.509	0.479	0.550	0.577
	EE	0.612	0.376	0.793	0.549	0.449	0.481	0.545	0.530
	LOF	0.444	0.501	0.493	0.536	0.508	0.505	0.496	0.453
	KDE	0.701	0.386	0.743	0.575	0.519	0.475	0.552	0.609
	CEN	0.699	0.467	0.746	0.566	0.522	0.481	0.551	0.597
SAE	RE	0.491	0.846	0.945	0.745	0.443	0.501	0.538	0.408
	OCSVM Linear	0.244	0.244	0.500	0.807	0.607	0.511	0.603	0.162
	OCSVM Poly	0.106	0.028	1.000	0.424	0.515	0.492	0.589	0.512
	OCSVM Rbf	0.871	0.964	1.000	0.614	0.538	0.486	0.552	0.706
	OCSVM Sigmoid	0.461	0.080	0.762	0.673	0.278	0.517	0.399	0.686
	IF	0.852	0.973	1.000	0.579	0.496	0.480	0.572	0.674
	EE	0.867	0.971	1.000	0.552	0.400	0.486	0.559	0.607
	LOF	0.449	0.497	0.509	0.577	0.539	0.505	0.503	0.451
	KDE	0.943	0.960	1.000	0.564	0.523	0.478	0.575	0.606
	CEN	0.856	0.963	1.000	0.584	0.560	0.487	0.547	0.634
DVAE	RE	0.166	0.803	0.697	0.425	0.587	0.502	0.484	0.546
	OCSVM Linear	0.489	0.728	0.762	0.333	0.630	0.478	0.466	0.506
	OCSVM Poly	0.489	0.728	0.762	0.333	0.630	0.479	0.466	0.506
	OCSVM Rbf	0.930	0.981	1.000	0.594	0.476	0.472	0.590	0.458
	OCSVM Sigmoid	0.489	0.728	0.762	0.333	0.630	0.478	0.466	0.506
	IF	0.932	0.976	1.000	0.547	0.449	0.473	0.585	0.481
	EE	0.896	0.974	1.000	0.527	0.434	0.474	0.581	0.545
	LOF	0.449	0.501	0.493	0.569	0.559	0.504	0.500	0.452
	KDE	0.938	0.956	1.000	0.555	0.463	0.471	0.589	0.459
	CEN	0.928	0.981	1.000	0.568	0.466	0.473	0.586	0.454
DNCAE	RE	*0.679*	*0.848*	0.839	0.625	0.406	**0.508**	*0.691*	**0.756**
	OCSVM Linear	**0.979**	**0.989**	**1.000**	*0.811*	*0.682*	*0.624*	*0.616*	*0.661*
	OCSVM Poly	0.964	**0.989**	**1.000**	*0.815*	*0.682*	**0.623**	*0.616*	*0.661*
	OCSVM Rbf	*0.943*	*0.981*	**1.000**	0.567	0.439	**0.677**	**0.684**	**0.727**
	OCSVM Sigmoid	**0.979**	*0.959*	**1.000**	*0.821*	0.682	*0.624*	*0.616*	**0.691**
	IF	0.936	*0.982*	**1.000**	**0.582**	*0.532*	*0.676*	*0.681*	**0.697**
	EE	*0.908*	**0.995**	**1.000**	*0.565*	**0.491**	*0.675*	**0.690**	*0.765*
	LOF	**0.450**	**0.502**	0.496	**0.598**	**0.569**	*0.605*	**0.680**	**0.752**
	KDE	0.945	*0.964*	**1.000**	*0.569*	**0.553**	**0.674**	*0.693*	**0.692**
	CEN	*0.948*	*0.979*	**1.000**	**0.591**	0.642	*0.677*	**0.684**	*0.638*
DCHAE	RE	**0.690**	**0.902**	**0.946**	0.400	**0.646**	*0.507*	**0.694**	*0.617*
	OCSVM Linear	*0.974*	*0.988*	0.762	**0.907**	**0.783**	**0.627**	**0.619**	**0.720**
	OCSVM Poly	**0.980**	**0.989**	0.764	**0.907**	**0.783**	*0.607*	**0.619**	**0.675**
	OCSVM Rbf	**0.947**	**0.983**	**1.000**	**0.615**	0.457	*0.675*	*0.681*	*0.713*
	OCSVM Sigmoid	*0.964*	0.958	0.767	**0.907**	*0.783*	**0.627**	**0.619**	*0.690*
	IF	0.942	0.973	**1.000**	*0.579*	**0.542**	**0.678**	**0.690**	*0.688*
	EE	0.890	**0.995**	**1.000**	**0.727**	*0.482*	**0.678**	*0.683*	**0.784**
	LOF	**0.450**	*0.501*	0.498	**0.597**	*0.563*	**0.606**	**0.680**	**0.752**
	KDE	**0.960**	**0.969**	**1.000**	*0.569*	**0.553**	*0.672*	**0.697**	*0.690*
	CEN	**0.949**	*0.979*	**1.000**	*0.590*	**0.655**	**0.679**	**0.684**	**0.699**

Table 4: AUCs of DAE+OCCs, DCAE+OCCs, PCADCAE+OCCs, SAE+OCCs, DVAE+OCCs, DNCAE+OCCs, DHCAE+OCCs models on seven scenarios of CTU13

Latent Representation	**One-Class Classifiers**	**Datasets**						
		CTU13-06	CTU13-07	CTU13-08	CTU13-09	CTU13-10	CTU13-12	CTU13-13
DAE	RE	0.952	0.675	0.925	0.542	0.222	0.669	0.758
	OCSVM Linear	0.988	0.999	0.964	0.931	0.960	0.751	0.948
	OCSVM Poly	0.988	0.933	0.957	0.929	0.958	0.749	0.916
	OCSVM Rbf	0.993	0.666	0.946	0.565	0.918	0.864	0.933
	OCSVM Sigmoid	0.988	0.999	0.951	0.931	0.960	0.751	0.953
	IF	0.992	0.998	0.973	0.946	0.956	0.953	0.959
	EE	0.673	0.880	0.938	0.622	0.828	0.785	0.916
	LOF	0.502	0.523	0.502	0.488	0.470	0.531	0.499
	KDE	0.990	0.999	0.974	0.766	0.998	0.901	0.940
	CEN	0.991	0.953	0.946	0.654	0.926	0.874	0.911
DCAE	RE	0.372	0.324	0.884	0.715	0.756	0.666	0.861
	OCSVM Linear	0.073	0.981	0.179	0.878	0.590	0.602	0.486
	OCSVM Poly	0.075	0.981	0.179	0.878	0.584	0.602	0.484
	OCSVM Rbf	0.986	0.998	0.990	0.950	0.995	0.960	0.971
	OCSVM Sigmoid	0.072	0.981	0.179	0.878	0.604	0.601	0.501
	IF	0.983	0.998	0.982	0.954	0.994	0.964	0.969
	EE	0.985	0.766	0.969	0.755	0.997	0.866	0.911
	LOF	0.497	0.522	0.502	0.484	0.461	0.518	0.495
	KDE	0.986	0.999	0.990	0.954	0.998	0.969	0.973
	CEN	0.986	0.998	0.988	0.953	0.995	0.956	0.971
PCADCAE	RE	0.538	0.524	0.881	0.725	0.786	0.695	0.832
	OCSVM Linear	0.927	0.976	0.959	0.173	0.999	0.386	0.749
	OCSVM Poly	0.927	0.976	0.959	0.173	0.999	0.386	0.749
	OCSVM Rbf	0.981	0.995	0.985	0.944	0.998	0.943	0.968
	OCSVM Sigmoid	0.927	0.976	0.959	0.173	0.999	0.386	0.749
	IF	0.981	0.994	0.986	0.947	0.998	0.955	0.969
	EE	0.922	0.891	0.969	0.766	0.999	0.892	0.901
	LOF	0.526	0.546	0.502	0.495	0.483	0.542	0.492
	KDE	0.992	0.999	0.986	0.946	0.998	0.961	0.972
	CEN	0.981	0.996	0.988	0.953	0.998	0.950	0.969
SAE	RE	0.402	0.406	0.793	0.726	0.906	0.648	0.721
	OCSVM Linear	0.968	0.971	0.958	0.945	0.862	0.046	0.617
	OCSVM Poly	0.901	0.760	0.574	0.630	0.138	0.043	0.534
	OCSVM Rbf	0.978	0.997	0.975	0.857	0.975	0.964	0.946
	OCSVM Sigmoid	0.964	0.980	0.787	0.336	0.974	0.093	0.761
	IF	0.987	0.996	0.988	0.865	0.976	0.977	0.957
	EE	0.916	0.899	0.971	0.747	0.985	0.757	0.799
	LOF	0.505	0.481	0.507	0.489	0.467	0.526	0.499
	KDE	0.991	0.999	0.984	0.811	0.997	0.939	0.955
	CEN	0.987	0.995	0.937	0.766	0.943	0.921	0.924
DVAE	RE	0.254	0.621	0.876	0.047	0.965	0.387	0.754
	OCSVM Linear	0.980	0.983	0.191	0.961	0.996	0.673	0.787
	OCSVM Poly	0.979	0.983	0.191	0.961	0.996	0.673	0.787
	OCSVM Rbf	0.993	0.995	0.993	0.946	0.994	0.909	0.966
	OCSVM Sigmoid	0.980	0.983	0.191	0.961	0.994	0.674	0.787
	IF	0.992	0.997	0.994	0.953	0.990	0.916	0.964
	EE	0.991	0.832	0.991	0.754	0.994	0.870	0.917
	LOF	0.507	0.480	0.503	0.482	0.461	0.536	0.502
	KDE	0.995	0.998	0.993	0.950	0.999	0.941	0.971
	CEN	0.993	0.995	0.992	0.942	0.994	0.908	0.966
DNCAE	RE	0.761	**0.844**	*0.909*	0.671	*0.927*	**0.814**	**0.888**
	OCSVM Linear	**0.988**	*0.986*	0.910	**0.987**	0.954	*0.913*	**0.959**
	OCSVM Poly	*0.988*	0.971	0.910	**0.987**	0.954	*0.913*	**0.959**
	OCSVM Rbf	**0.994**	**0.998**	**0.993**	*0.950*	0.954	0.955	0.968
	OCSVM Sigmoid	**0.988**	0.981	0.910	**0.987**	0.954	*0.913*	**0.959**
	IF	*0.993*	0.997	**0.994**	**0.959**	*0.994*	0.961	**0.971**
	EE	*0.987*	*0.911*	*0.992*	**0.782**	0.797	**0.903**	0.914
	LOF	**0.527**	**0.546**	*0.508*	*0.489*	**0.490**	0.515	0.493
	KDE	**0.996**	**0.999**	*0.991*	*0.955*	0.958	*0.969*	*0.972*
	CEN	**0.995**	*0.996*	**0.993**	*0.953*	0.954	*0.956*	0.969
DCHAE	RE	*0.897*	*0.810*	0.865	**0.824**	0.811	0.677	0.763
	OCSVM Linear	**0.988**	**0.999**	**0.984**	*0.985*	**0.999**	**0.930**	0.901
	OCSVM Poly	**0.989**	**0.998**	**0.984**	**0.987**	**0.999**	**0.930**	0.901
	OCSVM Rbf	*0.993*	**0.998**	0.981	**0.956**	**0.999**	**0.974**	**0.972**
	OCSVM Sigmoid	**0.988**	**0.999**	**0.984**	*0.980*	*0.994*	**0.930**	0.901
	IF	**0.994**	**0.999**	*0.990*	**0.959**	0.982	*0.975*	*0.969*
	EE	**0.991**	**0.990**	**0.993**	*0.766*	*0.997*	0.877	**0.922**
	LOF	0.501	*0.526*	**0.509**	*0.489*	*0.488*	0.530	0.498
	KDE	*0.995*	**0.999**	0.987	**0.961**	**1.000**	**0.981**	**0.973**
	CEN	**0.995**	**0.999**	0.987	**0.954**	**0.998**	**0.974**	**0.973**

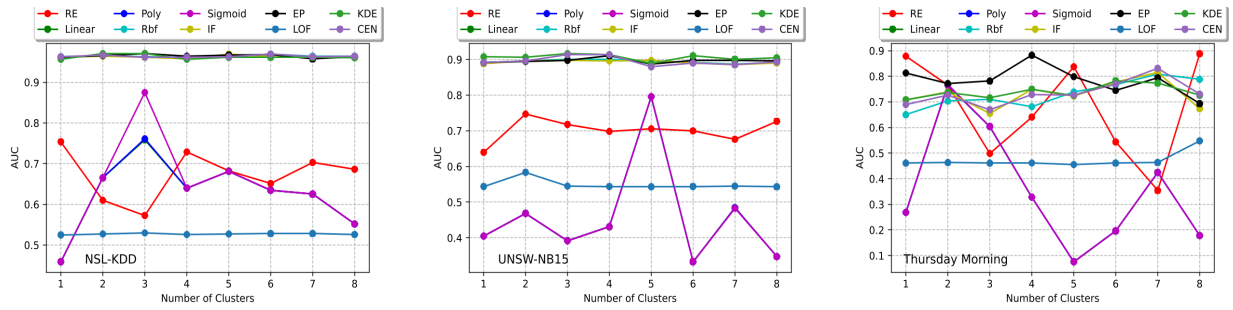


(a) NSL-KDD (b) UNSW-NB15 (c) Thursday Morning

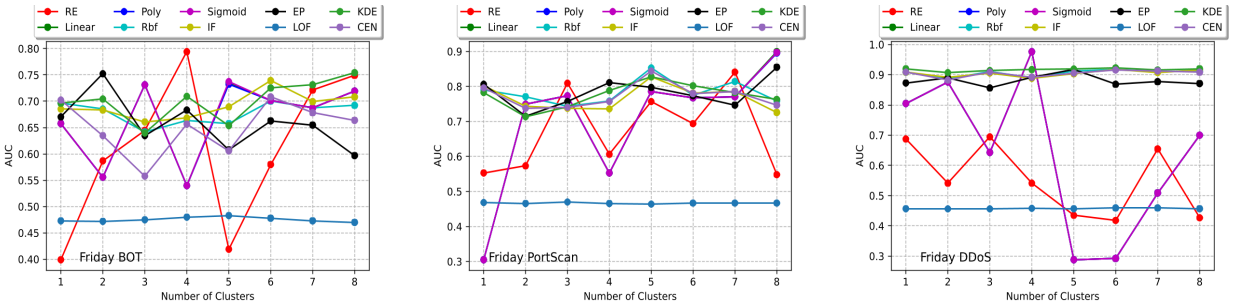


(d) Friday-BOT (e) Friday-PortScan (f) Friday-DDoS

Figure 5: Investigation number of Clusters of DNCAE



(a) NSL-KDD (b) UNSW-NB15 (c) Thurday Morning



(d) Friday BOT (e) Friday PortScan (f) Friday DDoS

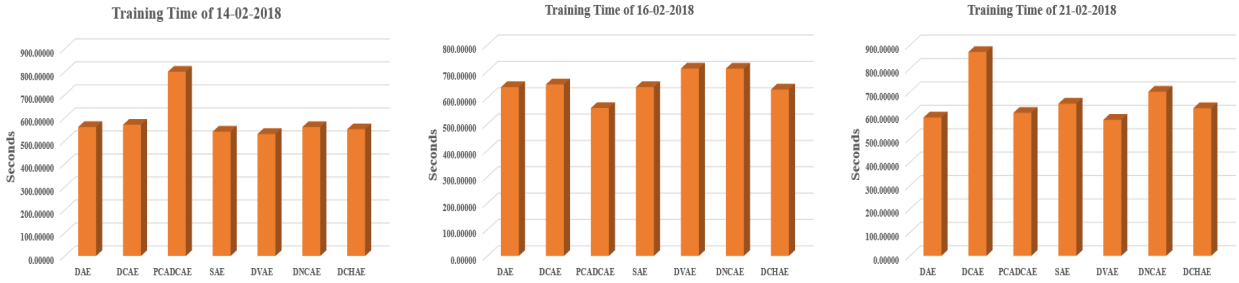
Figure 6: Investigation number of Clusters of DCHAE

vironment using an Nvidia GeForce RTX 3060 GPU.

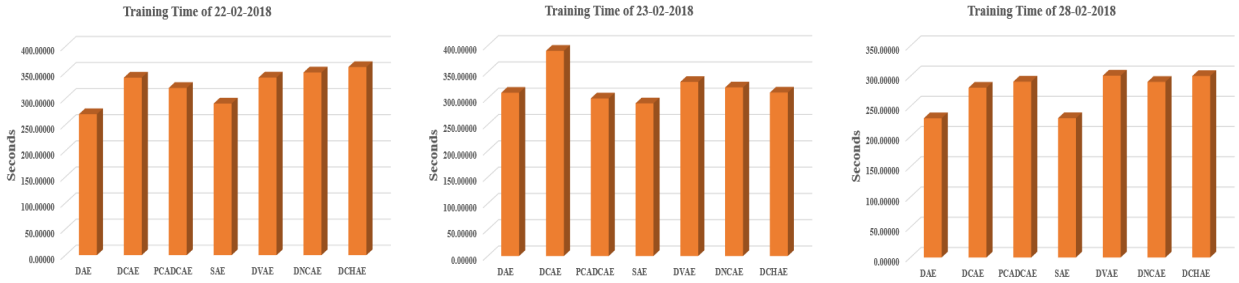
6. RESULTS AND DISCUSSION

In this section, we will present the experimental results of the proposed models DNCAE and DCHAE as tested on the chosen datasets outlined in Section 5.1, as well as comparing these results against baseline and state-of-the-art models. The experimental results are shown in Tables 2, 3, 4, respectively.

In order to facilitate the analysis of experimental results and aid performance comparisons between models, a bold font is used to highlight the highest results, and an italic font with a background fill is used to represent the second-highest results.

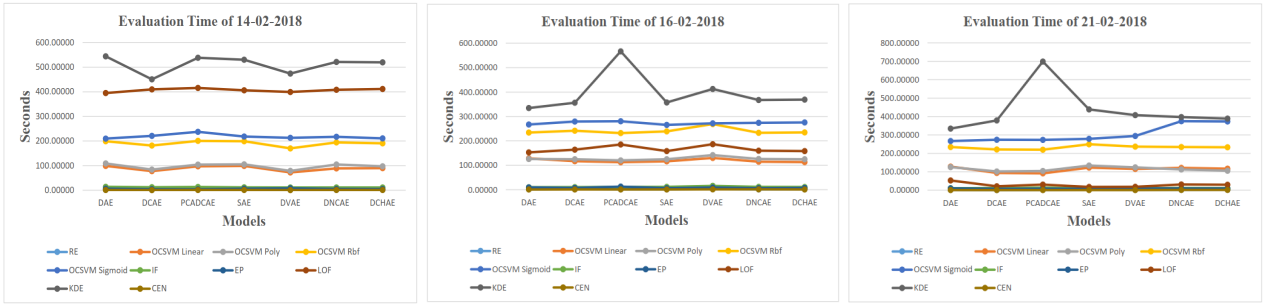


(a) 14-02-2018 (b) 16-02-2018 (c) 21-02-2018

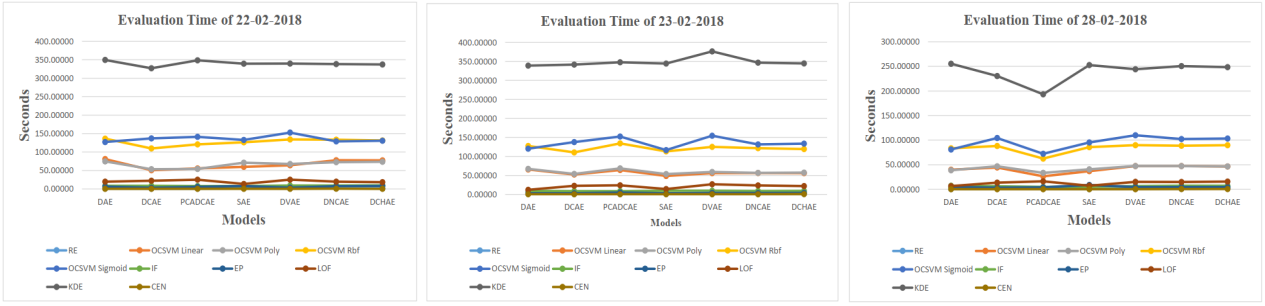


(d) 22-02-2018 (e) 23-02-2018 (f) 28-02-2018

Figure 7: Training Time Investigation



(a) 14-02-2018 (b) 16-02-2018 (c) 21-02-2018



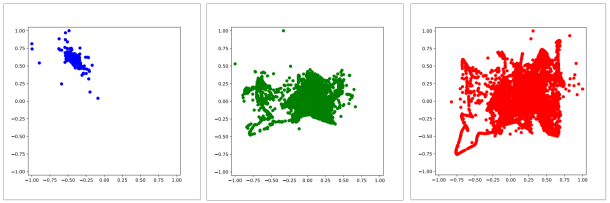
(d) 22-02-2018 (e) 23-02-2018 (f) 28-02-2018

Figure 8: Testing Time Investigation

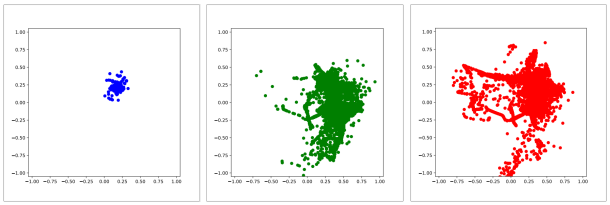
6.1. Performance Analysis of DNCAE and DCHAE

6.1.1. NSL-KDD, UNSW-NB15 and CIC-IDS-2017 Scenarios

Overall, the experimental results in Table 2 indicate that models DNCAE and DCHAE produced promising results and outperformed baseline and state-of-the-art models. With the NSL-KDD set, the learned latent representation space of DNCAE enabled five anomaly detectors to produce the highest results, whilst the latent representation space of DCHAE combined with eight anomaly detectors produced the highest results. Specifically, the learned latent features of DNCAE combined with the RE-method, OCSVM Poly, OCSVM Rbf, and EE detectors have produced the best results of 0.802, 0.961, 0.969, and 0.970, respectively. Model DCHAE produced the highest results combined with seven out of ten selected methods. In detail, DCHAE combined with OCSVM Rbf, OCSVM

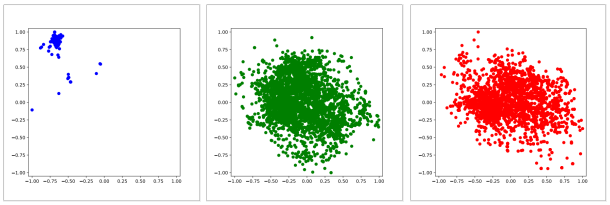


(a) Normal Train-DNCAE (b) Normal Test-DNCAE (c) Abnormal Test-DNCAE

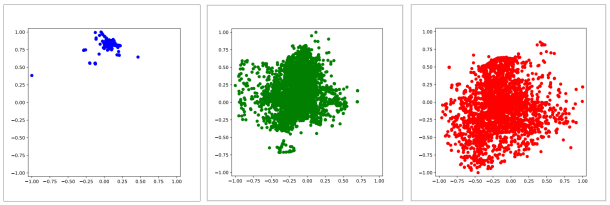


(d) Normal Train-DCHAE (e) Normal Test-DCHAE (f) Abnormal Test-DCHAE

Figure 9: Visualization of NSL-KDD

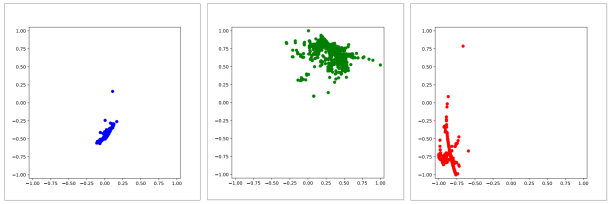


(a) Normal Train-DNCAE (b) Normal Test-DNCAE (c) Abnormal Test-DNCAE

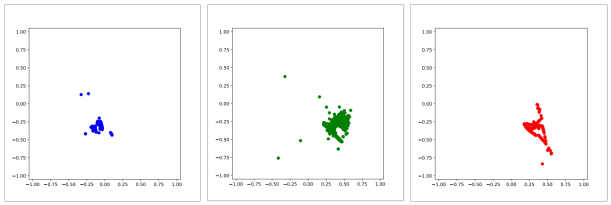


(d) Normal Train-DCHAE (e) Normal Test-DCHAE (f) Abnormal Test-DCHAE

Figure 10: Visualization of UNSW-NB15

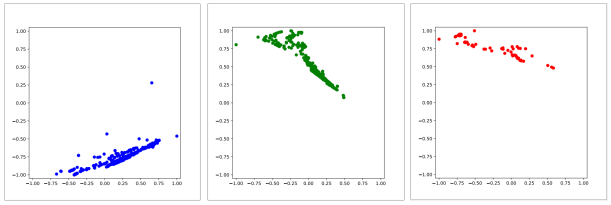


(a) Normal Train-DNCAE (b) Normal Test-DNCAE (c) Abnormal Test-DNCAE

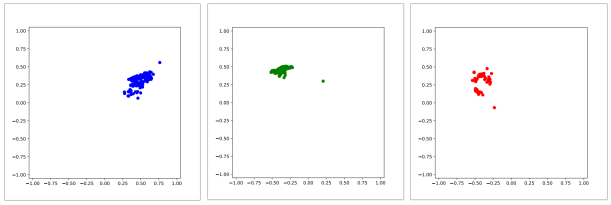


(d) Normal Train-DCHAE (e) Normal Test-DCHAE (f) Abnormal Test-DCHAE

Figure 11: Visualization of CTU13-06



(a) Normal Train-DNCAE (b) Normal Test-DNCAE (c) Abnormal Test-DNCAE



(d) Normal Train-DCHAE (e) Normal Test-DCHAE (f) Abnormal Test-DCHAE

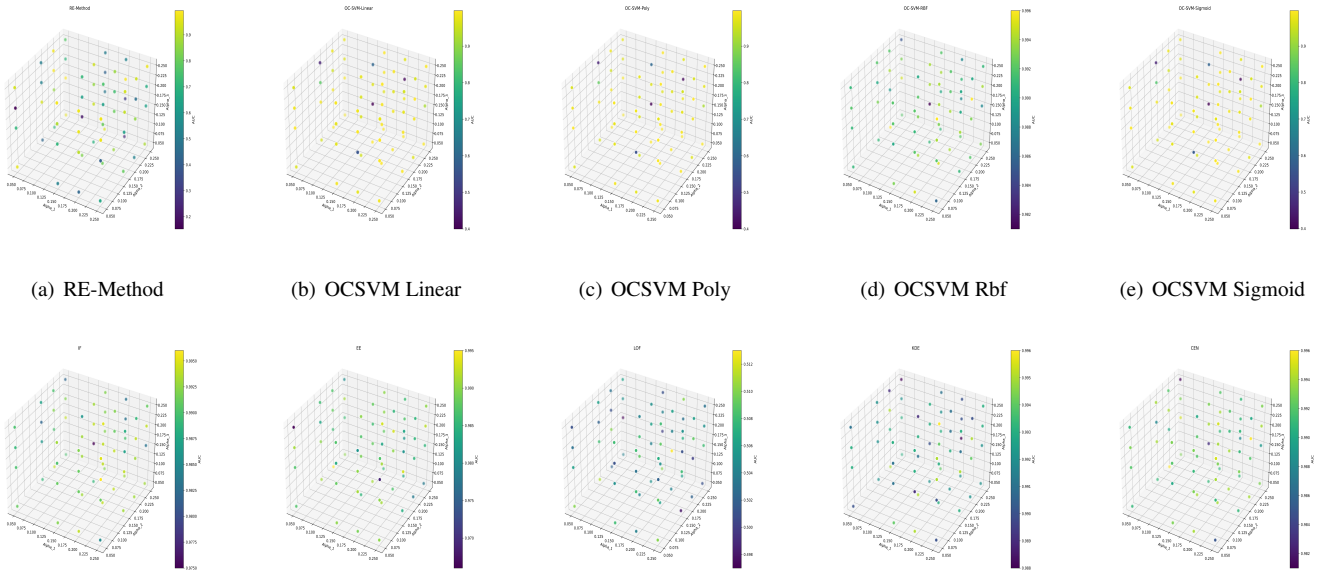
Figure 12: Visualization of CTU13-07

Sigmoid, IF, EE, LOF, KDE, and CEN produces results of 0.969, 0.875, 0.969, 0.970, 0.530, 0.970, 0.969, respectively. The highest experimental result on the NSL-KDD data set when using DNCAE and DCHAE is both 0.970, outperforming the two state-of-the-art models SAE (0.957) and DVAE (0.955).

Similar results were obtained when conducting experiments with the UNSW-NB15 dataset. Models DNCAE and DCHAE have learned the latent representation space and are able to support anomaly detectors much better than baseline models and both state-of-the-art models. Using the DNCAE learned latent features, five of the anomaly detectors (RE-method, OCSVM Rbf, IF, EE, and KDE) gave the highest AUC results. Also, with these features, the remaining five anomaly detectors also produced the second-highest results compared to all other models. The highest result across all anomaly detectors was 0.916, this was achieved with the combination of DNCAE+KDE and DCHAE+KDE, while from the baselines and state-of-the-art models, the best result produced was 0.908.

For scenario Tuesday (CIC-IDS-2017), DNCAE combined with OCSVM Linear, OCSVM Poly, and OCSVM Sigmoid produced the highest result of 0.925. DNCAE combined with CEN also gives the highest result of 0.788, while DNCAE combined with IF gives the second highest result of 0.753. DCHAE generated meaningful features and, when fed into anomaly detectors, produced the five best results and four second-highest results compared to other models. When comparing all models and all classifiers, the best result of 0.925 was achieved when DNCAE was combined with OCSVM Linear, OCSVM Poly and OCSVM Sigmoid.

For the Wednesday scenario, DNCAE combined with five anomaly detectors gives the best results, including OCSVM Rbf, IF, EE, KDE, and CEN. Additionally, when combined with LOF, the model gives the second-highest result. Meanwhile, the latent space generated by DCHAE also supports five anomaly



(a) RE-Method (b) OCSVM Linear (c) OCSVM Poly (d) OCSVM Rbf (e) OCSVM Sigmoid

(f) IF (g) EE (h) LOF (i) KDE (j) CEN

Figure 13: Investigaion of Loss Components for DNCAE

detectors to accomplish the highest results and five other detectors with the second-highest results, (RE-Method, OCSVM Linear, OCSVM Poly, OCSVM Sigmoid, LOF), and (OCSVM Rbf, IF, EE, KDE, CEN), respectively. When comparing all models and all anomaly detectors, the best result is 0.938, when DNCAE is combined with OCSVM Rbf.

For the remaining scenarios of the CIC-IDS-2017 dataset (Thursday Morning, Friday-Bot, Friday-DDoS, Friday PortScan Models), DNCAE and DCHAE both give superior results compared to the baselines and state-of-the-art models. This shows that DNCAE and DCHAE's ability to learn latent representations from network data is better than both baseline models and state-of-the-art models. In other words, DNCAE and DCHAE can learn the most significant, condensed, and meaningful features, supporting more powerful anomaly detectors. Therefore, the performance of these models is also significantly improved.

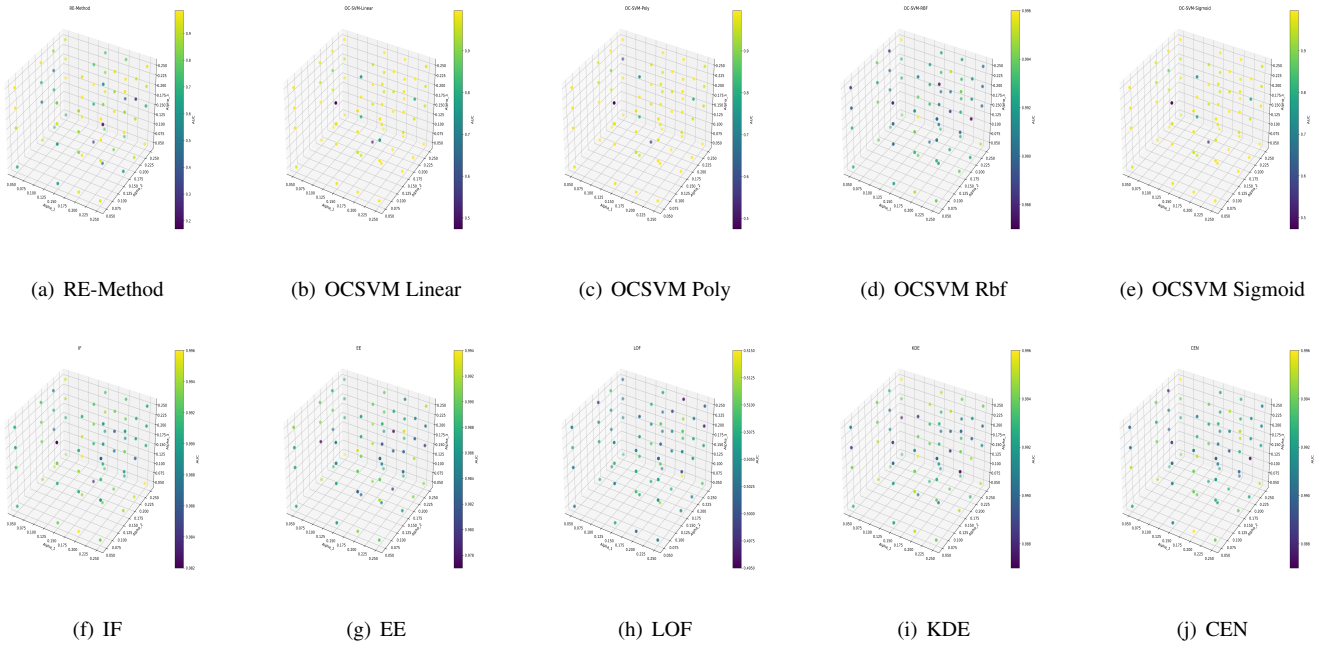
6.1.2. CSE-CIC-IDS-2018 Scenarios

The experimental results from the eight scenarios of the CSE-CIC-IDS-2018 data set are provided in Table 3.

For the 14-02-2018 scenario, the latent features generated from DNCAE supported three anomaly detectors (OCSVM Linear, OCSVM Sigmoid, LOF) that gained the highest results and four anomaly detectors (RE-method, OCSVM Rbf, EE, CEN) that scored the second highest. Specifically, DNCAE, when combined with OCSVM Linear and OCSVM Sigmoid, achieves an AUC of 0.979, and when combined with LOF, the result is 0.450. The AUC scores obtained when using the learned features of DNCAE with anomaly detectors, including RE-method, OCSVM Rbf, EE, and CEN, are 0.679, 0.943, 0.908, and 0.948, respectively. Similarly, the DCHAE model, when combined with anomaly detectors, also gives much higher results than the baselines, SAE, and DVAE models. More specifically, DCHAE, when combined with RE-method, OCSVM Poly, OCSVM Rbf, LOF, KDE, and CEN, produces the highest AUC results of (0.690, 0.980, 0.947, 0.450, 0.960, 0.949) respectively. When comparing all of the models and all anomaly detectors, the maximum result achieved is 0.980 in the DCHAE+OCSVM Poly scenario. This result is much higher than that of SAE and DVAE when combined with the ten selected anomaly detectors.

For the 16-02-2018 scenario, while DNCAE produced four highest values and six second-highest values, model DCHAE produced five highest values and three second-highest values. The maximum result considering all options combining models and anomaly detectors is 0.995 when DNCAE and DCHAE are combined with EE. Meanwhile, the best result achieved by the DVAE model combined with anomaly detectors is 0.981 when DVAE combined with CEN and OCSVM Rbf. Furthermore, the highest result produced by the SAE model in terms of AUC measure is 0.973 when SAE is combined with the IF detector.

For the 21-02-2018 scenario, models DNCAE and DCHAE generally have shown promising results. Specifically, DNCAE produced eight highest values of 1.000, and DCHAE produced five highest values of 1.000. DNCAE has demonstrated strong latent space learning ability, through which the most meaningful, condensed features are extracted and improve the performance of almost all anomaly detectors, especially OCSVM-based ones. Specifically, with the OCSVM Linear detector, the AUC scores have been improved from 0.500 and 0.762 to 1.000 from the SAE, DVAE, and DNCAE models, respectively. Meanwhile, when combining the IF, EE, KDE, and CEN anomaly detectors, model DNCAE also produces a re-



(a) RE-Method (b) OCSVM Linear (c) OCSVM Poly (d) OCSVM Rbf (e) OCSVM Sigmoid

(f) IF (g) EE (h) LOF (i) KDE (j) CEN

Figure 14: Investigation of Loss Components for DCHAE

sult of 1.000, equal to the combination of SAE and DVAE with these detectors. The RE-method applied with the DCHAE model produces the highest AUC measure of 0.946 compared to the DNCAE model of 0.839, DVAE of 0.697, and SAE model of 0.945. Experimental results on the remaining scenarios of the CSE-CIC-IDS-2018 have shown that models DNCAE and DCHAE both produce better scores when combined with the anomaly detector compared to baselines and state-of-the-art models. We argue that the clustering algorithm is helpful in finding the optimal arrangement of data points in the latent space while designing two nested DAEs of DNCAE and 2 branched DAEs of DCHAE has aided in minimizing the volume of the normal data area in the latent space. As a result, the best, most expressive features of normal network data will be learned, making anomaly detection easier.

6.1.3. CTU-13 Scenarios

Experimental results on seven situations of CTU-13 dataset are provided in Table 4. Similar to the above datasets, in general, the latent features generated from the DNCAE and DCHAE models support network anomaly detectors much better than the baselines and state-of-the-art methods when using the CTU-13 dataset.

For the CTU13-06 scenario, six out of the ten anomaly detectors using features from the DNCAE model produced the best results, including OCSVM Linear, OCSVM Rbf, OCSVM Sigmoid, LOF, KDE and CEN. The AUC score values achieved are 0.988, 0.994, 0.988, 0.527, 0.996 and 0.995, respectively. Meanwhile, the OCSVM Poly, IF, and EE classifiers produced the second-highest results of 0.988, 0.993 and 0.987, respectively. The latent space explored from DCHAE also supports six anomaly detectors that obtained the highest AUC including OCSVM Linear (0.988), OCSVM Poly (0.989), OCSVM Sigmoid (0.988), IF (0.994), EE (0.991), and CEN (0.995). Additionally, the RE-method, OCSVM Rbf, and KDE classifiers also achieved the second-highest results of 0.897, 0.993, and 0.995, respectively. When compared to all combinations of models and anomaly detectors, the global highest value of 0.996 is achieved when KDE uses the latent features generated from DNCAE.

Experimental results on the scenario CTU13-07 have demonstrated the ability of DNCAE and DCHAE to learn the latent space better than other models. In particular, the latent features learned from DCHAE supported eight out of the ten highest-performing anomaly detectors and the two second-highest-performing anomaly detectors. The global maximum value achieved is 0.999 when DCHAE is combined with anomaly detectors, including OCSVM Linear, OCSVM Sigmoid, IF, KDE, and CEN. The results obtained when experimenting with other situations of the CTU-13 showed that the latent features discovered by DNCAE and DCHAE after the training process provided stronger support for anomaly detectors in the IoT environment. In other words, the learned features are more meaningful and are able to separate normal network data and attack network data better.

In summary, the experimental results have shown that the latent representation generated from DNCAE and DCHAE can separate normal and abnormal network data better than baselines and state-of-the-art models. These results are demonstrated by using ten different OCC anomaly detectors. DNCAE and DCHAE have significantly improved network anomaly detection accuracy when testing across a diverse range of commonly-used NIDS datasets. These results can be explained by the following points:

- (i) The presence of the clustering loss component at the first hierarchical bottleneck in models DNCAE and DCHAE has pushed similar data points into the same cluster, and these cluster centers closer together. Although the training data is completely normal data, we believe that its internal architecture still contains small clusters, which are generated from different devices and protocols, and the differing normal behaviors of users. In other words, in the latent variable space of OuterDAE and Bigger DAE in DNCAE and DCHAE respectively, the optimal arrangement of normal observations has been established, thereby supporting the minimization of the super-volume for the "normal region" in this space.

- (ii) The reconstruction loss components in the objective function DNCAE and DCHAE are $\mathcal{L}_{\text{Outer}}\left(\mathbf{X}, \hat{\mathbf{X}}\right)$ and $\mathcal{L}_{\text{Big}}\left(\mathbf{X}, \hat{\mathbf{X}}\right)$ respectively, which are responsible for keeping normal data points from overlapping in the latent space and thereby learning meaningful, concise features representing normal data so that the original data can be recreated at the output layer.

- (iii) With the presence of Inner DAE and Small DAE in the model architecture of DNCAE and DCHAE, the process of distilling the condensed, core, abstract features of normal network data is performed at the second hierarchy. As a result, outstanding features with high separability are extracted as a more reliable basis for classifying normal and abnormal network data. This implies that by using latent features in this second hierarchy, the boundaries between normal and abnormal data regions become clearer, which helps anomaly detectors improve accuracy in terms of AUC scores. This process is similar to the evolutionary development process of animals in nature when the characteristics of the first generation will be filtered, distilled, and condensed for the next generation. As a result, the characteristics of the next generation are often better than the characteristics of the previous generation.

The experimental results of almost all scenarios confirm that the ability of DNCAE and DCHAE to learn latent representations from normal network data is better than that of state-of-the-art models, including SAE, DVAE, and PCADCAE. These results can be explained by the following points:

- (i) The SAE model only tries to minimize the normal data area over the latent space. However, in that model, clustering techniques are not applied, so the arrangement of data points is not optimal and tight. Consequently, abnormal samples may appear in the gaps between normal data points or interspersed with normal data points in adjacent border areas.

- (ii) The DVAE model makes assumptions about the prior probability of the latent representation space that are too strict. Therefore, it may not accurately reflect the probability distribution of very diverse normal data types.

- (iii) The PCADCAE model integrates a clustering algorithm in the latent space but only uses the latent space in the first hierarchy. This implies that the feature quality in the second hierarchy has improved compared to the first. In other words, stronger features were distilled out in the second hierarchy.

6.2. Investigating the influence of number clusters for DNCAE and DCHAE

Separate experiments were conducted to study the influence of the number of clusters in the K-means algorithm at the first hierarchical latent space in models DNCAE and DCHAE, on the performance of anomaly detectors. In other words, we want to learn about the internal architecture of normal network data, to find the most optimal and reasonable arrangement. The datasets selected for this experiment were NSL-KDD, UNSW-NB15, Thursday Morning, Friday-BOT, Friday-PortScan, and Friday-DDoS. For both DNCAE and DCHAE, we incrementally evaluated the number of clusters used, within the range of 1 to 8.

Figure 5 and Figure 6 demonstrate the effect of the number of clusters in models DNCAE and DCHAE on the performance of the 10 anomaly detectors. Based on the experimental results, we can conclude that the selection of the number of clusters for models DNCAE and DCHAE has an important influence on the arrangement of normal data points in the latent space. Consequently, it will affect the performance of each anomaly detector. In most experimental situations, anomaly detectors achieve their best values at one or two specific numbers of clusters. For example, in the set NSL-KDD, the suitable number of clusters will be 3, in the set Thursday Morning, the suitable number of clusters will be 5. Therefore, each data set will have a certain number of best-fit clusters. This also confirms that the internal architecture of each data set is different because it is collected from different network environments.

6.3. Comparing training and testing times of DNCAE and DCHAE with other models

Experiments were also conducted to evaluate the training and testing times of DNCAE and DCHAE compared to previous models. For the purpose of evaluation, we trained and tested the models using the same data sets under the identical experimental conditions, these were: 14-02-2018, 16-02-2018, 21-02-2018, 22-02-2018, 23-02-2018, and 28-02-2018. The graphs in Figure 7 and Figure 8 compare the training time and testing time of models, respectively.

Regarding the training time of the models, we note two important points: i) Generally, the training time of DNCAE and DCHAE is equivalent to the training time of other models, but in **some situations it is shorter**, such as experiments on 14-02-2018, 21-02-2018, and 23-02-2018; ii) By comparing the training time of DNCAE and DCHAE, we see that the training time is equivalent, meaning both models have almost the same convergence speed on selected data sets.

When considering the testing time of the models, we also recognize the following points: i) In all experimental situations,

the KDE anomaly detector always consumes the most testing time, followed by OCSVM Sigmoid. Additionally, the CEN anomaly detector is always the method with the shortest testing time; ii) There is no difference in testing times using data generated from either DNCAE or DCHAE; iii) The testing times from DNCAE and DCHAE are shorter than those taken using latent representation generated from other models. This can be explained by the dimensionality at the latent space of DNCAE and DCHAE being smaller than the dimensionality of the latent space of other models.

6.4. Determining the influence of loss components in the objective function of models DNCAE and DCHAE

Experiments were conducted to study the influence o f the loss components in the objective function of models DNCAE and DCHAE on the quality of the latent representation generated by these two models. In other words, investigating the effect these loss components have on the accuracy of 10 selected anomaly detectors. To implement this, the values α_1, α_2, α_3 are chosen in the objective function of DNCAE, and β_1, β_2, β_3 in the objective function of DCHAE in the range of values [0.05, 0.15, 0.2, 0.25].

The CTU13-06 data set was selected to conduct these experiments for both DNCAE and DCHAE. AUC scores are recorded using 10 network anomaly detectors with each combination of hyperparameters α_1, α_2, α_3 for DNCAE and β_1, β_2, β_3 for DCHAE. Experimental results are depicted on a 3D graph with the 3 axes being the corresponding values of the 3 hyperparameters, and using additional color columns to represent the AUC values. Figure 13 and Figure 14 show the results for 10 anomaly detectors using the latent space of DNCAE and DCHAE, respectively. Based on experimental results, we find that there are three anomaly detectors that have stable accuracy results on most combinations of hyperparameters when using features generated from DNCAE and DCHAE: OCSVM Linear, OCSVM Poly and OCSVM Sigmoid. As with the remaining anomaly detectors, only very few combinations of hyperparameters produce the largest AUC measure. In other words, it is necessary to harmoniously control the effects of the three loss components to create good latent features that are desirable and suitable for each anomaly detector.

6.5. Visualizing the latent space of DNCAE and DCHAE

In order to observe the arrangement of observations in the latent space of models DNCAE and DCHAE, we visualize in 2D space three types of data including: Normal data used for training, Normal data used for testing, and Abnormal data used for testing. We selected 4 datasets for visualization purposes: NSL-KDD, UNSW-NB15, CTU13-06 and CTU13-07. The arrangement of data points from selected datasets in the latent space of DNCAE and DCHAE is illustrated in Figures 9, 10, 11 and 12. Visualization is used to supplement the results presented in Tables 2, 3 and 4. Overall, it can be seen that the data in the hidden space of DCHAE is neater and tighter than the hidden space of DNCAE.

In summary, our proposed models achieve superior results because latent feature extraction is performed twice. This reduces the dimensionality of the latent space, improving time in the testing phase. Additionally, the double latent feature distillation process captures more abstract, high-level, and essential features to characterize normal network data, thereby enhancing the performance of anomaly detectors.

The proposed models have demonstrated the capability to efficiently and effectively detect conventional attacks in well-known datasets such as NSL-KDD and UNSW-NB15. They are also adept at identifying new varieties of modern attack patterns from real environments, like those found in CIC-IDS-2017 and CSE-CIC-IDS-2018, as well as Botnet-based attacks against IoT networks, which are detailed in the Table 1. In terms of the computational complexity of the proposed models, during the training phase, we have to use two DAEs and K-means clustering techniques. Particularly, the complexity of one DAE is $O(N \times \sum_{l=1}^{L} n_l \times n_{l-1})$. The complexity of K-means is $O(N \times k \times d)$. Where L is the number of hidden layers of DAE; n_l is number the number of neurons in layer l; N is the number of samples in the training set; k is the number of clusters, and d is the dimensionality of the latent space. However, during the inference phase, only the Encoder parts of the DAEs are utilized. Therefore, the computational complexity for each Encoder part for each data point is $O(\sum_{l=1}^{L/2} n_l \times n_{l-1})$. The complexity for assigning a point to a cluster is $O(k \times d)$. Furthermore, the inference processing time has been significantly reduced and improved compared to baselines and state-of-the-art models. This improvement highlights the potential for deploying the proposed models in real-world applications, making them adaptable to various network environments for detecting a wide range of anomaly-based cyber attack techniques.

7. CONCLUSION AND FUTURE WORK

In this paper, we have proposed two novel latent representation learning models, DNCAE and DCHAE, to improve network anomaly detection accuracy and the required processing time during the testing phase. The two models are designed in unprecedented ways, including 2 DAEs nested together and divided into two separate branches. In addition, the clustering technique is integrated at the bottleneck layer of OuterDAE in DNCAE and Big DAE in DCHAE with the purpose of pushing normal network data points into suitable small clusters. These two models combine the power for latent representation learning of DAE with the fast, effective clustering ability of K-means to create a hierarchical representation space of normal network data. As a result, DNCAE and DCHAE have distilled high-level abstract, condensed, meaningful, and separated latent features, which strongly support anomaly detectors to handle high-dimensional data, speed up processing time and especially increase accuracy.

We evaluate the performance of the proposed models using a selection of the most common NIDS datasets. We chose various anomaly detectors (10 techniques) to evaluate the quality of the latent features generated from the proposed models. Experimental results in most situations have demonstrated that the

proposed models produced higher AUC scores than baseline models, and state-of-the-art models (PCADCAE [24], SAE, and DVAE [23]).

We also investigated the impact of the number of clusters chosen for K-means, on the performance of the proposed models when combined with 10 anomaly detectors. The training and testing times of DNCAE and DCHAE were also recorded, analyzed, and compared with existing models. The results show that the proposed models have training time equivalent to the previous model, but improve data processing time during testing of query data points. The hyperparameters that control the dominance of loss components in the objective function of models DNCAE and DCHAE are also studied to clarify their influence on model performance. Additionally, we conduct visualization in 2D space to clearly see the arrangement of data points in the latent space at the second hierarchy of the proposed models.

In the future, we plan to propose other data representation learning solutions to further improve network anomaly detection performance. In addition, methods to optimize the hyperparameters of the proposed models will be studied.

References

[1] A. Khang, S. K. Gupta, S. Rani, D. A. Karras, Smart Cities: IoT Technologies, big data solutions, cloud platforms, and cybersecurity techniques, CRC Press, 2023.

[2] Ö. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz, E. Akin, A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions, Electronics 12 (6) (2023) 1333.

[3] T. Yi, X. Chen, Y. Zhu, W. Ge, Z. Han, Review on the application of deep learning in network attack detection, Journal of Network and Computer Applications 212 (2023) 103580.

[4] Y. Shah, S. Sengupta, A survey on classification of cyber-attacks on iot and iiot devices, in: 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), IEEE, 2020, pp. 0406–0413.

[5] W. Duo, M. Zhou, A. Abusorrah, A survey of cyber attacks on cyber physical systems: Recent advances and challenges, IEEE/CAA Journal of Automatica Sinica 9 (5) (2022) 784–800.

[6] M. A. Haque, S. Haque, K. Kumar, N. K. Singh, A comprehensive study of cyber security attacks, classification, and countermeasures in the internet of things, in: Handbook of research on digital transformation and challenges to data security and privacy, IGI Global, 2021, pp. 63–90.

[7] M. Mittal, K. Kumar, S. Behal, Deep learning approaches for detecting ddos attacks: A systematic review, Soft computing 27 (18) (2023) 13039–13075.

[8] A. Sharma, B. B. Gupta, A. K. Singh, V. Saraswat, Advanced persistent threats (apt): evolution, anatomy, attribution and countermeasures, Journal of Ambient Intelligence and Humanized Computing (2023) 1–27.

[9] S. Singh, P. K. Sharma, S. Y. Moon, D. Moon, J. H. Park, A comprehensive study on apt attacks and countermeasures for future networks and communications: challenges and solutions, The Journal of Supercomputing 75 (2019) 4543–4574.

[10] S. M. Tahsien, H. Karimipour, P. Spachos, Machine learning based solutions for security of internet of things (iot): A survey, Journal of Network and Computer Applications 161 (2020) 102630.

[11] M. Devi, A. Majumder, Side-channel attack in internet of things: A survey, in: Applications of Internet of Things: Proceedings of ICCCIOT 2020, Springer, 2021, pp. 213–222.

[12] A. Khraisat, A. Alazab, A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges, Cybersecurity 4 (2021) 1–27.

[13] X. Li, M. Xu, P. Vijayakumar, N. Kumar, X. Liu, Detection of low-frequency and multi-stage attacks in industrial internet of things, IEEE Transactions on Vehicular Technology 69 (8) (2020) 8820–8831.

[14] M. Pawlicki, M. Choraś, R. Kozik, Defending network intrusion detection systems against adversarial evasion attacks, Future Generation Computer Systems 110 (2020) 148–154.

[15] S. Gamage, J. Samarabandu, Deep learning methods in network intrusion detection: A survey and an objective comparison, Journal of Network and Computer Applications 169 (2020) 102767.

[16] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, F. Ahmad, Network intrusion detection system: A systematic study of machine learning and deep learning approaches, Transactions on Emerging Telecommunications Technologies 32 (1) (2021) e4150.

[17] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, H. Janicke, Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study, Journal of Information Security and Applications 50 (2020) 102419.

[18] M. Bhavsar, K. Roy, J. Kelly, O. Olusola, Anomaly-based intrusion detection system for iot application, Discover Internet of Things 3 (1) (2023) 5.

[19] H. Zhang, Y. Li, Z. Lv, A. K. Sangaiah, T. Huang, A real-time and ubiquitous network attack detection based on deep belief network and support vector machine, IEEE/CAA Journal of Automatica Sinica 7 (3) (2020) 790–799.

[20] M. Pawlicki, R. Kozik, M. Choraś, A survey on neural networks for (cyber-) security and (cyber-) security of neural networks, Neurocomputing 500 (2022) 1075–1087.

[21] A. Drewek-Ossowicka, M. Pietrołaj, J. Rumiński, A survey of neural networks usage for intrusion detection systems, Journal of Ambient Intelligence and Humanized Computing 12 (2021) 497–514.

[22] G. Pang, C. Shen, L. Cao, A. V. D. Hengel, Deep learning for anomaly detection: A review, ACM computing surveys (CSUR) 54 (2) (2021) 1–38.

[23] M. Nicolau, J. McDermott, et al., Learning neural representations for network anomaly detection, IEEE transactions on cybernetics 49 (8) (2018) 3074–3087.

[24] V. Q. Nguyen, V. H. Nguyen, N.-A. Le Khac, N. Shone, et al., A robust pca feature selection to assist deep clustering autoencoder-based network anomaly detection, in: 2021 8th NAFOSTED Conference on Information and Computer Science (NICS), IEEE, 2021, pp. 335–341.

[25] V. Q. Nguyen, T. L. Ngo, V. H. Nguyen, N. Shone, et al., Deep nested clustering auto-encoder for anomaly-based network intrusion detection, in: 2023 RIVF International Conference on Computing and Communication Technologies (RIVF), IEEE, 2023, pp. 289–294.

[26] L. Vu, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang, E. Dutkiewicz, et al., Learning latent representation for iot anomaly detection, IEEE Transactions on Cybernetics 52 (5) (2020) 3769–3782.

[27] N. Abdalgawad, A. Sajun, Y. Kaddoura, I. A. Zualkernan, F. Aloul, Generative deep learning to detect cyberattacks for the iot-23 dataset, IEEE Access 10 (2021) 6430–6441.

[28] L. Vu, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang, E. Dutkiewicz, Deep generative learning models for cloud intrusion detection systems, IEEE Transactions on Cybernetics 53 (1) (2022) 565–577.

[29] W. Wu, Traffic anomaly detection method based on mislearned autoencoder, in: 2023 5th International Academic Exchange Conference on Science and Technology Innovation (IAECST), IEEE, 2023, pp. 112–116.

[30] S. Zavrak, M. İskefiyeli, Anomaly-based intrusion detection from network flow features using variational autoencoder, IEEE Access 8 (2020) 108346–108358.

[31] H. Kye, M. Kim, M. Kwon, Hierarchical autoencoder for network intrusion detection, in: ICC 2022 - IEEE International Conference on Communications, IEEE, 2022, pp. 2700–2705.

[32] B. Lewandowski, R. Paffenroth, K. Campbell, Improving network intrusion detection using autoencoder feature residuals, in: 2022 4th International Conference on Data Intelligence and Security (ICDIS), IEEE, 2022, pp. 31–39.

[33] S. Aktar, A. Y. Nur, Robust anomaly detection in iot networks using deep svdd and contractive autoencoder, in: 2024 IEEE International Systems Conference (SysCon), IEEE, 2024, pp. 1–8.

[34] M. K. Hooshmand, D. Hosahalli, Network anomaly detection using deep learning techniques, CAAI Transactions on Intelligence Technology 7 (2) (2022) 228–243.

[35] S. M. Kasongo, A deep learning technique for intrusion detection system using a recurrent neural networks based framework, Computer Commu-

nications 199 (2023) 113–125.
[36] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, R. Ahmad, Cnn-lstm: hybrid deep neural network for network intrusion detection system, IEEE Access 10 (2022) 99837–99849.
[37] H. Xu, G. Pang, Y. Wang, Y. Wang, Deep isolation forest for anomaly detection, IEEE Transactions on Knowledge and Data Engineering 35 (12) (2023) 12591–12604.
[38] H. Xu, Y. Wang, G. Pang, S. Jian, N. Liu, Y. Wang, Rosas: Deep semi-supervised anomaly detection with contamination-resilient continuous supervision, Information Processing & Management 60 (5) (2023) 103459.
[39] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.
[40] U. Michelucci, An introduction to autoencoders, arXiv preprint arXiv:2201.03898 (2022).
[41] T. Cemgil, S. Ghaisas, K. Dvijotham, S. Gowal, P. Kohli, The autoencoding variational autoencoder, Advances in Neural Information Processing Systems 33 (2020) 15077–15087.
[42] D. Bank, N. Koenigstein, R. Giryes, Autoencoders, Machine Learning for Data Science Handbook: Data Mining and Knowledge Discovery Handbook (2023) 353–374.
[43] V. Q. Nguyen, V. H. Nguyen, T. H. Hoang, N. Shone, A novel deep clustering variational auto-encoder for anomaly-based network intrusion detection, in: 2022 14th International Conference on Knowledge and Systems Engineering (KSE), IEEE, 2022, pp. 1–7.
[44] Z. Chen, C. K. Yeo, B. S. Lee, C. T. Lau, Autoencoder-based network anomaly detection, in: 2018 Wireless telecommunications symposium (WTS), IEEE, 2018, pp. 1–5.
[45] C. Zhou, R. C. Paffenroth, Anomaly detection with robust deep autoencoders, in: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, 2017, pp. 665–674.
[46] C. M. Bishop, N. M. Nasrabadi, Pattern recognition and machine learning, Vol. 4, Springer, 2006.
[47] K. P. Sinaga, M.-S. Yang, Unsupervised k-means clustering algorithm, IEEE access 8 (2020) 80716–80727.
[48] P. Perera, P. Oza, V. M. Patel, One-class classification: A survey, arXiv preprint arXiv:2101.03064 (2021).
[49] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, J. Platt, Support vector method for novelty detection, Advances in neural information processing systems 12 (1999).
[50] M. M. Breunig, H.-P. Kriegel, R. T. Ng, J. Sander, Lof: identifying density-based local outliers, in: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, 2000, pp. 93–104.
[51] G. R. Terrell, D. W. Scott, Variable kernel density estimation, The Annals of Statistics (1992) 1236–1265.
[52] F. T. Liu, K. M. Ting, Z.-H. Zhou, Isolation-based anomaly detection, ACM Transactions on Knowledge Discovery from Data (TKDD) 6 (1) (2012) 1–39.
[53] P. J. Rousseeuw, K. V. Driessen, A fast algorithm for the minimum covariance determinant estimator, Technometrics 41 (3) (1999) 212–223.
[54] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE transactions on pattern analysis and machine intelligence 35 (8) (2013) 1798–1828.
[55] M. Tavallaee, E. Bagheri, W. Lu, A. A. Ghorbani, A detailed analysis of the kdd cup 99 data set, in: 2009 IEEE symposium on computational intelligence for security and defense applications, Ieee, 2009, pp. 1–6.
[56] N. Moustafa, J. Slay, Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set), in: 2015 military communications and information systems conference (MilCIS), IEEE, 2015, pp. 1–6.
[57] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization., ICISSp 1 (2018) 108–116.
[58] X. Li, W. Chen, Q. Zhang, L. Wu, Building auto-encoder intrusion detection system based on random forest feature selection, Computers & Security 95 (2020) 101851.
[59] S. Garcia, M. Grill, J. Stiborek, A. Zunino, An empirical comparison of botnet detection methods, computers & security 45 (2014) 100–123.
[60] V. Q. Nguyen, V. H. Nguyen, N.-A. Le-Khac, V. L. Cao, Clustering-based deep autoencoders for network anomaly detection, in: Future Data and Security Engineering: 7th International Conference, FDSE 2020, Quy Nhon, Vietnam, November 25–27, 2020, Proceedings 7, Springer, 2020, pp. 290–303.