


ROBUST FRAUD DETECTION VIA SUPERVISED CONTRASTIVE LEARNING


A PREPRINT

 **Vinay M.S.**

University of Arkansas
Fayetteville, AR 72701, USA
vmadanbh@uark.edu

 **Shuhan Yuan**

Utah State University
Logan, UT 84322, USA
Shuhan.Yuan@usu.edu

 **Xintao Wu**

University of Arkansas
Fayetteville, AR 72701, USA
xintaowu@uark.edu

ABSTRACT

Deep learning models have recently become popular for detecting malicious user activity sessions in computing platforms. In many real-world scenarios, only a few labeled malicious and a large amount of normal sessions are available. These few labeled malicious sessions usually do not cover the entire diversity of all possible malicious sessions. In many scenarios, possible malicious sessions can be highly diverse. As a consequence, learned session representations of deep learning models can become ineffective in achieving a good generalization performance for unseen malicious sessions. To tackle this open-set fraud detection challenge, we propose a robust supervised contrastive learning based framework called *ConRo*, which specifically operates in the scenario where only a few malicious sessions having limited diversity is available. ConRo applies an effective data augmentation strategy to generate diverse potential malicious sessions. By employing these generated and available training set sessions, ConRo derives separable representations w.r.t open-set fraud detection task by leveraging supervised contrastive learning. We empirically evaluate our ConRo framework and other state-of-the-art baselines on benchmark datasets. Our ConRo framework demonstrates noticeable performance improvement over state-of-the-art baselines.

Keywords fraud detection · contrastive learning · open-set · augmentation

1 Introduction

Computing platforms, such as **social networking sites and cloud systems**, experience large volumes of malicious or fraudulent activities due to the anonymity and openness of the Internet. It is critical **to identify such malicious activities** in order to protect legitimate users. In practice, the activities of a user are usually modeled as an activity session. For example, in a computer system, an activity session is a sequence of user activities starting with log-in and ending with log-out. A popular approach for detecting malicious sessions is through deep learning models [Yuan and Wu, 2021]. **The main idea is to derive session representations by making normal sessions deviate from malicious ones in the representation space for deriving anomaly scores.**

In many real-world fraud detection scenarios, only a few labeled malicious and an abundance of normal sessions are available [Yuan and Wu, 2021, Yuan et al., 2020]. These few available malicious sessions usually do not sufficiently cover the entire diversity of all possible malicious sessions. It is well known that malicious sessions can be highly diverse [Yuan and Wu, 2021]. Many attackers keep evolving their activity patterns to avoid detection. Such malicious sessions are usually not available for training a deep learning model. Suppose a deep learning model is trained by utilizing a few available malicious sessions. Now in the testing phase, due to the large diversity in the possible malicious sessions, the test set distribution might be different from the training set distribution. For example, the training set might contain only a few types of malicious sessions, and **the test set might include other types of malicious sessions that are not observed in the training set.** Hence, the learned session representations by using these few malicious sessions in the training set might not be discriminative enough to achieve good generalization on detecting unseen malicious sessions. Clearly, **the fraud detection task is essentially an open-set detection task.**

The existing deep anomaly detection approaches which operate on the setting of a few available anomalous samples, employ metric learning [Ruff et al., 2018, 2020] or deviation loss based learning [Pang et al., 2021a, 2019]. These approaches attempt to obtain a decision boundary by using a few available anomalies. However, these approaches can easily overfit w.r.t seen anomalies and can suffer from poor generalization performance if the anomalies encountered during the testing stage deviate from the training set anomalies [Ding et al., 2022]. To address this challenge, recently, Ding et al. [2022] presented a novel open-set deep anomaly detection approach. They train their model to detect unseen anomalies by jointly employing: (1) a data augmentation strategy through which they generate augmented samples that can closely resemble unseen anomalies, and (2) learning in the latent residual representation space. However, their approach has been specifically designed to operate on image data. In the fraud detection domain, we have additional challenges when compared to the image domain. For example in image data, the normal samples are assumed to have shared features. However, in the fraud detection domain, even normal sessions can also exhibit large diversity. Therefore, learning separable representations for the open-set fraud detection task is challenging.

We address these challenges by leveraging contrastive learning. The vanilla contrastive learning model operates in a self-supervised format. The main goal is to push a sample and its augmented versions closer and contrast with other samples and their corresponding augmented versions in the representation space. However, the employed augmentation strategies are constrained to produce augmented samples that are closely similar to their original versions. Due to this constraint, we cannot employ strong augmentation strategies to generate diverse malicious sessions. Recently, Khosla et al. [2020] presented a supervised contrastive learning model which extends contrastive learning to the supervised setting. The main goal is to push the samples belonging to the same class together and contrast with other class samples in the representation space. Due to this class-specific clustering effect in the representation space, the session diversity challenge in our fraud detection task can be effectively addressed. Hence, we leverage this supervised contrastive learning model to build our new robust fraud detection framework called *ConRo*.

However, the challenge here is to generate those augmented sessions which are similar and can be effective replacements for unseen malicious sessions. *ConRo* addresses this challenge by employing a two-stage training framework. In the first stage, *ConRo* trains the session encoder by using the available training set which contains a few malicious and a large amount of normal sessions. Specifically, it performs first stage training by employing a combination of both supervised contrastive and Deep Support Vector Data Description (DeepSVDD) [Ruff et al., 2018] losses. Through the supervised contrastive loss, *ConRo* learns shared features for normal sessions in the representation space, and through DeepSVDD loss, it pushes normal sessions in a minimum volume hyper-sphere in the representation space. After this stage, *ConRo* creates a representation space with suitable topological properties which aid in generating potentially diverse malicious sessions. In the second stage, by employing suitable augmentation strategies, *ConRo* generates diverse potential malicious sessions in the representation space, filters those generated sessions which are false positives/normal, and further trains the encoder through supervised contrastive loss by employing available and generated potential malicious sessions. We summarize our main contributions below:

- We propose a novel framework called *ConRo* which is specifically designed for the open-set fraud detection task. Our *ConRo* framework operates in a scenario where only a few malicious and a large amount of normal sessions are available.
- We propose a Long-Short Term Memory (LSTM) based session encoder which is trained by employing both supervised contrastive and DeepSVDD losses.
- We propose a data augmentation strategy to generate diverse potential malicious sessions in the representation space. We propose a strategy to filter generated false positive sessions.
- We theoretically analyze the generalization performance of our *ConRo* framework and highlight important factors influencing its performance. We present an empirical study on three benchmark fraud detection datasets: CERT [Glasser and Lindauer, 2013], UMD-Wikipedia [Kumar et al., 2015], and Open-stack [Du et al., 2017] in which, we show superior performance of our *ConRo* framework over state-of-the-art baselines.

2 Related Work

Anomaly Detection. Anomaly detection is to detect data that significantly deviate from the majority of data [Pang et al., 2021b]. Recently, many deep anomaly detection approaches are developed by leveraging deep neural networks to learn representations of data so that, anomalies can be easily differentiated from the normal samples [Pang et al., 2021b, Ruff et al., 2021]. One common setting of anomaly detection assumes the availability of normal samples and aims to learn a decision boundary based on the normal data distribution [Ruff et al., 2018, 2020, Pang et al., 2018]. Pang et al. [2019, 2021a] proposed an end-to-end anomaly detection framework called *deviation network* which combines representation learning with anomaly scoring. However, all these deep learning-based anomaly detection approaches have been designed for closed-set anomaly detection task. In our empirical analysis study, we select some of these

approaches [Ruff et al., 2018, 2020, Pang et al., 2019, 2021a] as baselines, and show that they fail to deliver noticeable results on open-set fraud detection task.

Insider Threat Detection. It is a specific case of fraud detection wherein, the frauds are committed by organizational insiders. Deep learning based approaches have become popular in detecting insider threats. We direct the interested readers to Yuan and Wu [2021] for a comprehensive survey on deep learning based insider threat detection approaches. All these deep learning based approaches have not specifically addressed the dataset imbalance challenge in detecting insider threats. Recently, many deep learning based approaches [Yuan et al., 2020, Vinay et al., 2022, Zhou et al., 2022, Guo et al., 2021, Qi et al., 2022] have specifically addressed the dataset imbalance challenge. However, all these approaches address the closed-set fraud detection task.

Open-Set Recognition (OSR). Salehi et al. [2021] have extensively discussed about contemporary OSR approaches. Pang et al. [2021c] have addressed anomaly detection in the OSR scenario by employing unlabelled samples. Hendrycks et al. [2019] trained their model by exposing it to a small set of unseen class samples. However, in our fraud detection task, we do not utilize unlabelled [Pang et al., 2021c] or unseen [Hendrycks et al., 2019] malicious sessions for learning. Ding et al. [2022] proposed an open-set anomaly detection framework that learns disentangled representations for different groups of anomalies. In our empirical study, we select this open-set anomaly detection approach [Ding et al., 2022] as a baseline and show that it under-performs on open-set fraud detection task. Recently, Pang et al. [2023] proposed an open-set anomaly detection framework which operates in the semi-supervised setting. However, our fraud detection task does not operate in the semi-supervised setting.

Contrastive Learning. Jaiswal et al. [2020] presented an in-depth discussions on the applications of self-supervised contrastive learning on computer vision and NLP domains. In the literature, there is no work studying benefits of supervised contrastive learning for the open-set fraud detection task.

3 ConRo Framework

The user activities are modeled through activity sessions. Each session can consist of T user activities. Let e_{i_t} ($1 \leq t \leq T$) denote the t^{th} activity of the i^{th} session. Each activity in a session is represented by an embedding vector, which can be trained based on the word-to-vector model. Let $\mathbf{x}_{i_t} \in \mathbb{R}^d$ denote the word-to-vector representation of activity e_{i_t} , where d denotes the number of representation dimensions. Here, $\mathbf{x}_i = \{\mathbf{x}_{i_t}\}_{t=1}^T$ denotes the raw representation of the i^{th} session. Let \mathcal{X} and \mathcal{Y} denote the raw input representation of sessions and label set, respectively. Here, $\mathcal{Y} = \{0, 1\}$ where $y = 0$ and $y = 1$ denote normal and malicious sessions, respectively. Let \mathcal{D} denote the test set distribution over $\mathcal{X} \times \mathcal{Y}$ wherein, the test samples are drawn from \mathcal{D} . The training set \mathcal{T} contains a large amount of normal and a few malicious sessions. Let \mathcal{T}^0 and \mathcal{T}^1 denote sets of normal and malicious sessions in \mathcal{T} , respectively. The malicious sessions sampled from \mathcal{D} will also contain those unseen malicious sessions which are not present in \mathcal{T}^1 . Our ConRo framework has an encoder network that maps a session from its raw representation \mathbf{x} to an encoded representation vector \mathbf{z} . We adopt LSTM as the foundation of our encoder to derive the encoded session representations. Our encoder consists of two hidden layers with the same dimensions. The hidden representations derived from the top layer of LSTM for the activities in the session \mathbf{x}_i are denoted as $\{\mathbf{h}_{i_t}\}_{t=1}^T$. Here, $\mathbf{h}_{i_t} \in \mathbb{R}^d$. Then, the encoded session representation $\mathbf{z}_i \in \mathbb{R}^d$ is computed as $\mathbf{z}_i = \frac{1}{T} \sum_{t=1}^T \mathbf{h}_{i_t}$.

The main challenge which we are addressing is to design a procedure to obtain malicious sessions which are sampled from the test set distribution \mathcal{D} . To address this challenge, we construct potential malicious sessions which can be similar to malicious sessions sampled from \mathcal{D} . There are two main objectives for generating these potential malicious sessions:

1. **MO1.** Malicious sessions usually form multiple clusters in the encoded representation space [Ju et al., 2020]. Malicious sessions belonging to the same cluster usually share close similarities. Hence, we need to generate potential malicious sessions which are similar to a seen malicious session $\mathbf{x}_i \in \mathcal{T}^1$.
2. **MO2.** Suppose there are K malicious session clusters. However, the training set \mathcal{T} might only contain N ($N < K$) session clusters, and sessions belonging to remaining $K - N$ clusters are not present in \mathcal{T} . Note that the malicious sessions from these $K - N$ unseen clusters can diverge significantly from seen malicious sessions. We need to generate potential malicious sessions which belong to those $K - N$ clusters to effectively train our encoder.

ConRo achieves these main objectives by employing a two stage encoder training procedure. An illustration of this training procedure is shown in Figure 1. We provide detailed descriptions of both these stages below.

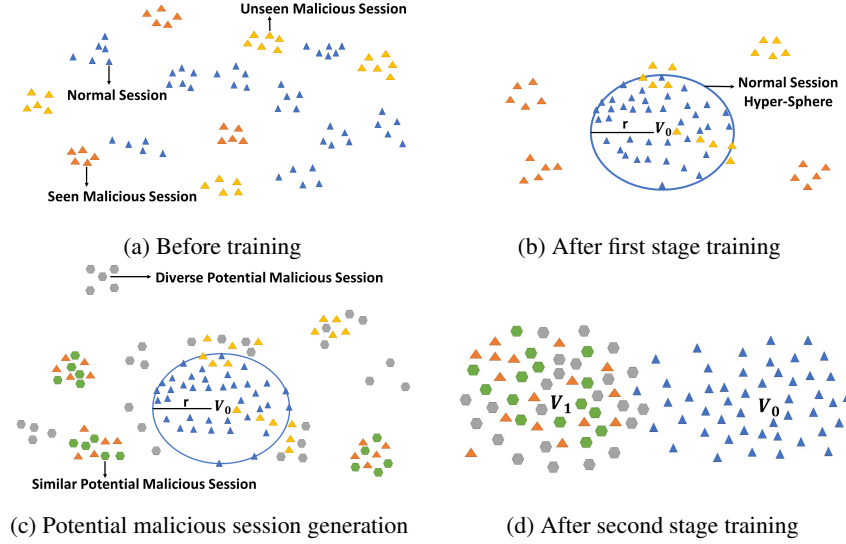


Figure 1: Illustration of encoded representation space for ConRo.

3.1 First Stage

In the first stage, our encoder achieves two goals: (1) It learns shared features for normal sessions and learns to contrast normal sessions with seen malicious sessions in the encoded representation space. As a consequence, our encoder learns separable representations w.r.t to normal and seen malicious sessions. (2) It compresses normal session representations inside a minimum volume hyper-sphere in the encoded representation space. To achieve the first goal, we leverage the idea of supervised contrastive learning, which can learn separable representations w.r.t to normal and seen malicious sessions. Then, to achieve the second goal, we leverage the DeepSVDD loss [Ruff et al., 2018], which pushes the normal samples inside a minimum volume hyper-sphere.

Supervised contrastive loss. We construct a training batch denoted as $S = \{\mathbf{x}_i\}_{i=1}^R$ by obtaining R random samples from \mathcal{T} . Since ConRo is specifically designed to operate on imbalanced training data, in order to effectively contrast malicious sessions with normal sessions, for each training batch S , we create a corresponding auxiliary batch $S^1 = \{\mathbf{x}_i^1\}_{i=1}^M$, by randomly sampling M malicious sessions from \mathcal{T}^1 . We leverage a supervised contrastive loss function similar to the one presented by Khosla et al. [2020]. This loss function is given by:

$$\mathcal{L}^{Sup} = \frac{1}{R} \sum_{i=1}^R (1 - y_i) \left(\frac{1}{|B^0(\mathbf{x}_i)|} \sum_{\mathbf{x}_p \in B^0(\mathbf{x}_i)} l(\mathbf{z}_i, \mathbf{z}_p, A(\mathbf{x}_i)) \right) \quad (1)$$

Here, the set $A(\mathbf{x}_i)$ is defined as $(S \cup S^1) - \{\mathbf{x}_i\}$, and the set $B^0(\mathbf{x}_i) = \{\mathbf{x}_p \in A(\mathbf{x}_i) | y_p = 0\}$ indicates samples \mathbf{x}_p in $A(\mathbf{x}_i)$ with labels $y_p = 0$. The individual loss $l(\mathbf{z}_i, \mathbf{z}_p, A(\mathbf{x}_i))$ between the pair $(\mathbf{x}_i, \mathbf{x}_p)$ is defined as:

$$l(\mathbf{z}_i, \mathbf{z}_p, A(\mathbf{x}_i)) = -\log \left(\frac{\exp(\cos(\mathbf{z}_i \cdot \mathbf{z}_p)/\alpha)}{\sum_{\mathbf{x}_j \in A(\mathbf{x}_i)} \exp(\cos(\mathbf{z}_i \cdot \mathbf{z}_j)/\alpha)} \right), \quad (2)$$

where α denotes the temperature parameter.

DeepSVDD loss. We leverage a DeepSVDD loss function which is similar to the one presented by Ruff et al. [2018]. Let $\mathbf{v}_0 = \frac{1}{R_0} \sum_{i=1}^R (1 - y_i) \mathbf{z}_i$ denote the estimated center of normal sessions in the encoded representation space and $R_0 = \sum_{i=1}^R \mathbb{I}(y_i = 0)$, where $\mathbb{I}(\cdot)$ is an indicator function. This loss function is given by:

$$\mathcal{L}^{SV} = \frac{1}{R} \sum_{i=1}^R (1 - y_i) (\|\mathbf{z}_i - \mathbf{v}_0\|_2) \quad (3)$$

The loss function for the first stage is given by:

$$\mathcal{L}_1 = \mathcal{L}^{Sup} + \mathcal{L}^{SV} \quad (4)$$

To effectively address the session diversity challenge, we employ **an alternating approach** to optimize our encoder through \mathcal{L}_1 , **instead of joint optimization**. In our ablation analysis study described in Section 4.3.1, we show that the alternating optimization approach provides **significant performance improvement over the joint optimization approach**. For each training batch $S = \{\mathbf{x}_i\}_{i=1}^R$, we first train our encoder through \mathcal{L}^{Sup} . As a result, we force the encoder to learn shared features for normal sessions and contrast with seen malicious sessions in the encoded representation space (goal 1). Then, by using the same batch S , we train the encoder through \mathcal{L}^{SV} , which forces the encoder to compress normal session representations inside a minimum volume hyper-sphere in the encoded representation space (goal 2).

After the first stage, an encoded representation space having certain topological properties is created (refer to Figures 1a and 1b). For example, in the encoded representation space, most of the normal sessions are pushed inside a minimum volume hyper-sphere. **The seen malicious sessions are found outside this hyper-sphere in multiple clusters and pulled apart from the normal session hyper-sphere**. An attractive option now is to directly deploy our first stage trained encoder for test case inference wherein, a test case session \mathbf{x} is predicted as malicious if $\|\mathbf{z} - \mathbf{v}_0\| > r$ where r is the radius of the normal session hyper-sphere. Otherwise, it is predicted as normal. Note that after the first stage, normal sessions have been contrasted with only seen malicious sessions belonging to \mathcal{T}^1 , and not unseen malicious sessions. **Thus, it is possible that many unseen malicious session clusters overlap with the normal session hyper-sphere, especially those which are similar to normal sessions** (refer to Figure 1b). Hence, by directly deploying the first stage trained encoder for test case inference might negatively affect generalization performance. In our **ablation** analysis study, we show that this option **provides sub-optimal results**. Hence, we require a strategy to generate diverse potential malicious sessions which can be similar to unseen malicious sessions in the encoded representation space, further train our encoder by employing these sessions, and learn separable representations w.r.t to the open-set fraud detection task.

3.2 Second Stage

In the second stage, for each seen malicious session $\mathbf{x}_i \in \mathcal{T}^1$, **we achieve the first objective (MO1) by generating potential/augmented malicious sessions closely resembling \mathbf{x}_i** . The first objective can be achieved by leveraging existing augmentation techniques [Verma et al., 2021]. However, the second objective **(MO2) cannot be achieved straightforwardly due to the lack of required augmentation strategies in the literature** [Jaiswal et al., 2020]. We achieve the second objective by using an **effective augmentation strategy to generate a large amount of potential malicious sessions that span diverse regions** of the encoded representation space (refer to Figure 1c). Note that many of these generated sessions might fall inside the **normal session hyper-sphere**. In such scenario, we have **two choices**: (1) a **pessimistic choice** where we consider such sessions as **potential malicious sessions**, and (2) an **optimistic choice** where we consider such sessions as **false positives**, and filter these sessions. Since deep learning models are sensitive to label noise, choosing the pessimistic choice could result in reduced generalization performance. Hence, we opt for the optimistic choice. In our ablation analysis study, we show that **making the pessimistic choice yields sub-optimal results**. We design **a session filtering mechanism** to filter such generated sessions which fall inside the normal session hyper-sphere. Since we generate a large amount of diverse potential malicious sessions, many of them can be located **just outside the boundary of the normal session hyper-sphere**, which could **partially cover unseen malicious session clusters** that overlap this hyper-sphere and aid in learning separable representations w.r.t to open-set fraud detection task. We do not individually train our encoder by considering normal sessions from \mathcal{T} to avoid over-fitting.

First objective. We **generate similar potential malicious sessions** which are similar to a **seen malicious session** $\mathbf{x}_i \in \mathcal{T}^1$. Recently, Verma et al. [2021] proposed a mix-up based data augmentation strategy for sequential data. Their augmentation strategy is inspired by the concept of *convex sets* and generates augmented samples that are similar to their original version. **Specifically, they generate augmented samples by performing a mix-up operation on the encoded representations of original samples**. Hence, we leverage a mix-up based augmentation strategy which is similar to the one presented by Verma et al. [2021] for generating similar potential malicious sessions which are similar to a seen malicious session $\mathbf{x}_i \in \mathcal{T}^1$.

Let $\hat{G}^1(\mathbf{x}_i)$ denote this set of generated similar potential malicious sessions. The set $\hat{G}^1(\mathbf{x}_i)$ is defined as $\hat{G}^1(\mathbf{x}_i) = \{\hat{\mathbf{z}} | \hat{\mathbf{z}} = \lambda_1 \mathbf{z}_i + (1 - \lambda_1) \mathbf{z}_j\}$. Here, $\hat{\mathbf{z}}$ denotes the encoded representation of a generated similar potential malicious session, $\mathbf{x}_j \in B^1(\mathbf{x}_i) = \{\mathbf{x}_p \in A(\mathbf{x}_i) | y_p = 1\}$ indicates samples \mathbf{x}_p in $A(\mathbf{x}_i)$ with labels $y_p = 1$, λ_1 is sampled from the Uniform distribution $U(\beta_1, 1)$ where $\beta_1 \in [0, 1]$, and β_1 is set closer to 1 to ensure that generated potential malicious sessions have close similarities with \mathbf{x}_i .

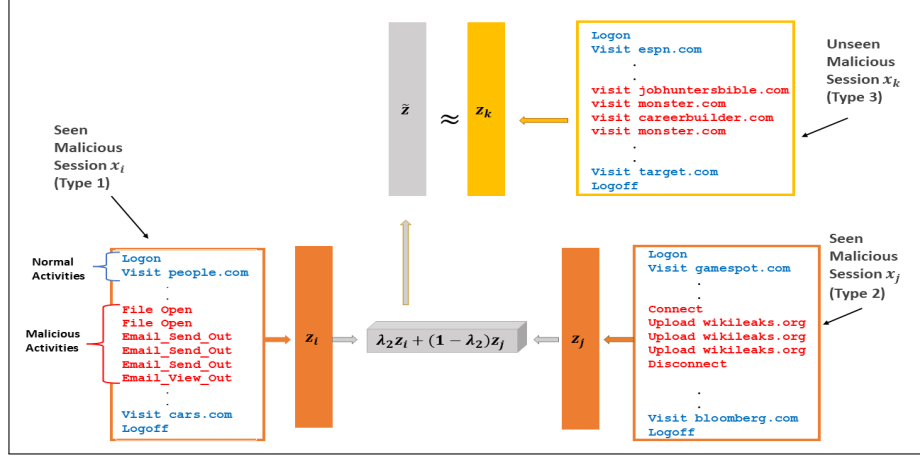


Figure 2: Illustration of generating a diverse potential malicious session (CERT dataset).

Intuitions behind the design of $\hat{G}^1(\mathbf{x}_i)$. Since $\beta_1 \in [0, 1]$ and β_1 is closer to 1, and λ_1 is sampled from the Uniform distribution $U(\beta_1, 1)$, by using $\lambda_1 \mathbf{z}_i$, we get a potential malicious session which is similar to and in the same direction of \mathbf{z}_i . Now, we want to generate potential malicious sessions which are not just confined to the same direction of \mathbf{z}_i and are surrounding \mathbf{z}_i in different directions. Thus, the operation $\tilde{\mathbf{z}} = \lambda_1 \mathbf{z}_i + (1 - \lambda_1) \mathbf{z}_j$, aids in achieving this goal. Additionally, performing mix-up with malicious session \mathbf{x}_j will aid in learning separable representations.

Second objective. We generate *diverse potential malicious sessions* which can diverge significantly from a seen malicious session $\mathbf{x}_i \in \mathcal{T}^1$. Let $\hat{G}^1(\mathbf{x}_i)$ denote this set of generated diverse potential malicious sessions. Our session augmentation strategy is inspired by the concept of *affine sets*. The set $\hat{G}^1(\mathbf{x}_i)$ is defined as $\hat{G}^1(\mathbf{x}_i) = \{\tilde{\mathbf{z}} | \tilde{\mathbf{z}} = \lambda_2 \mathbf{z}_i + (1 - \lambda_2) \mathbf{z}_j, fp(\tilde{\mathbf{z}}) = 0\}$. Here, $\tilde{\mathbf{z}}$ denotes the encoded representation of a generated diverse potential malicious session, $\lambda_2 \sim U(-\beta_2, \beta_2)$, and $\beta_2 \in \mathbb{R}$. We treat β_2 as a hyper-parameter in our empirical studies. We filter a false positive through the function $fp(\cdot)$ as:

$$fp(\tilde{\mathbf{z}}) = \begin{cases} 1, & \text{if } \|\tilde{\mathbf{z}} - \mathbf{v}_0\|_2 \leq r \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

Intuitions behind the design of $\hat{G}^1(\mathbf{x}_i)$. We describe our intuitions by using the CERT insider threat dataset. In this dataset, there are five types of malicious sessions. For the ease of description, we consider only three types. Type-1 sessions are related to frauds involving stealing sensitive information and emailing it to a malicious agent. Type-2 sessions are related to devising frauds which involve misusing hardware devices such as removable drives. Type-3 sessions are related to visiting unethical websites such as job-portals with an intent to abandon the current organization. Consider two malicious sessions \mathbf{x}_i and \mathbf{x}_j which belong to type-1 and type-2, respectively. These sessions are illustrated in Figure 2. Note that both \mathbf{x}_i and \mathbf{x}_j , follow a similar activity sequence pattern wherein, normal activities are followed by malicious activities, and finally again followed by normal activities.

For the malicious session \mathbf{x}_i , we can express \mathbf{z}_i as $\mathbf{z}_i = \mathbf{z}_i^0 \cup \mathbf{z}_i^1$ where \mathbf{z}_i^0 and \mathbf{z}_i^1 denote encoded representation feature sets corresponding to normal and malicious activities, respectively. Let $\mathbf{z}_i^0 \in \mathbb{R}^n$, $\mathbf{z}_i^1 \in \mathbb{R}^m$, and $n + m = d$. Since sessions \mathbf{x}_i and \mathbf{x}_j follow a similar activity sequence pattern, we can hypothesize that $\mathbf{z}_j = \mathbf{z}_j^0 \cup \mathbf{z}_j^1$ with $\mathbf{z}_j^0 \in \mathbb{R}^n$ and $\mathbf{z}_j^1 \in \mathbb{R}^m$. Due to the effect of first stage training, normal activity features are tightly clustered in the encoded representation space. Hence, we can infer that $\mathbf{z}_i^0 \approx \mathbf{z}_j^0$. Consider an unseen malicious session \mathbf{x}_k belonging to type-3 which is not present in \mathcal{T}^1 . This session \mathbf{x}_k is shown in Figure 2. Clearly, \mathbf{x}_k also follows a similar activity sequence pattern as \mathbf{x}_i and \mathbf{x}_j . Assume that \mathcal{T}^1 does not contain any type-3 malicious sessions. Now we will describe how $\hat{G}^1(\mathbf{x}_i)$ can aid in approximating unseen malicious sessions such as \mathbf{x}_k . Since \mathbf{x}_k follows a similar activity sequence pattern as \mathbf{x}_i and \mathbf{x}_j , we can hypothesize that $\mathbf{z}_k = \mathbf{z}_k^0 \cup \mathbf{z}_k^1$ with $\mathbf{z}_k^0 \in \mathbb{R}^n$ and $\mathbf{z}_k^1 \in \mathbb{R}^m$. We can infer the result $\mathbf{z}_i^0 \approx \mathbf{z}_j^0 \approx \mathbf{z}_k^0$. Then for any $\lambda_2 \in \mathbb{R}$, we have that: $\mathbf{z}_k^0 \approx \lambda_2 \mathbf{z}_i^0 + (1 - \lambda_2) \mathbf{z}_j^0$. Note that \mathbf{x}_i , \mathbf{x}_j and \mathbf{x}_k belong to different malicious session types and hence, \mathbf{z}_i^1 , \mathbf{z}_j^1 , and \mathbf{z}_k^1 can diverge significantly from each other. By sampling a suitable value for $\lambda_2 \sim U(-\beta_2, \beta_2)$, we employ λ_2 as a coefficient, and aim to obtain the result: $\mathbf{z}_k^1 \approx \lambda_2 \mathbf{z}_i^1 + (1 - \lambda_2) \mathbf{z}_j^1$.

Second stage loss. We again leverage supervised contrastive loss to design our second stage loss function which is given by:

$$\mathcal{L}_2 = \frac{1}{R} \sum_{i=1}^R \left[y_i \left(\frac{1}{|D(\mathbf{x}_i)|} \sum_{\mathbf{x}_p \in D(\mathbf{x}_i)} l(\mathbf{z}_i, \mathbf{z}_p, C(\mathbf{x}_i)) \right) \right] \quad (6)$$

Here, $C(\mathbf{x}_i) = A(\mathbf{x}_i) \cup \hat{G}^1(\mathbf{x}_i) \cup \tilde{G}^1(\mathbf{x}_i)$, $D(\mathbf{x}_i) = B^1(\mathbf{x}_i) \cup \hat{G}^1(\mathbf{x}_i) \cup \tilde{G}^1(\mathbf{x}_i)$ and $l(\mathbf{z}_i, \mathbf{z}_p, C(\mathbf{x}_i))$ denotes the individual loss between the pair $(\mathbf{x}_i, \mathbf{x}_p)$ corresponding to the malicious sessions defined as:

$$l(\mathbf{z}_i, \mathbf{z}_p, C(\mathbf{x}_i)) = -\log \left(\frac{\exp(\cos(\mathbf{z}_i \cdot \mathbf{z}_p)/\alpha)}{\sum_{\mathbf{x}_j \in C(\mathbf{x}_i)} \exp(\cos(\mathbf{z}_i \cdot \mathbf{z}_j)/\alpha)} \right) \quad (7)$$

3.3 Training

Algorithm 1 Training procedure for ConRo.

Inputs: $\mathcal{T} = \mathcal{T}^1 \cup \mathcal{T}^0$, R , M , \hat{M} , \tilde{M} , β_1 , β_2 and our untrained encoder.
Output: well trained encoder.

- 1: generate raw representations of all the sessions in \mathcal{T} ;
- [First Stage]**
- 2: **for** each training batch $S = \{\mathbf{x}_i\}_{i=1}^R$ generated from \mathcal{T} **do**
- 3: create the auxiliary batch $S^1 = \{\mathbf{x}_i^1\}_{i=1}^M$ from \mathcal{T}^1 ;
- 4: calculate $R_0 = \sum_{i=1}^R \mathbb{I}(y_i = 0)$ and $\mathbf{v}_0 = \frac{1}{R_0} \sum_{i=1}^R (1 - y_i) \mathbf{z}_i$;
- 5: **for** each normal session $(\mathbf{x}_i, y_i = 0) \in S$ **do**
- 6: construct set $A(\mathbf{x}_i) = (S \cup S^1) - \{\mathbf{x}_i\}$;
- 7: construct set $B^0(\mathbf{x}_i) = \{\mathbf{x}_p \in A(\mathbf{x}_i) | y_p = 0\}$;
- 8: calculate $l(\mathbf{z}_i, \mathbf{z}_p, A(\mathbf{x}_i))$ for each session $\mathbf{x}_p \in B^0(\mathbf{x}_i)$ by using Equation 2;
- 9: calculate \mathcal{L}^{Sup} by using Equation 1 and train the encoder;
- 10: calculate \mathcal{L}^{SV} by using Equation 3 and train the encoder;
- [Second Stage]**
- 11: **for** each training batch $S = \{\mathbf{x}_i\}_{i=1}^R$ generated from \mathcal{T} **do**
- 12: create the auxiliary batch $S^1 = \{\mathbf{x}_i^1\}_{i=1}^M$ from \mathcal{T}^1 ;
- 13: **for** each malicious session $(\mathbf{x}_i, y_i = 1) \in S$ **do**
- 14: construct set $A(\mathbf{x}_i) = (S \cup S^1) - \{\mathbf{x}_i\}$;
- 15: construct set $B^1(\mathbf{x}_i) = \{\mathbf{x}_p \in A(\mathbf{x}_i) | y_p = 1\}$;
- 16: $\hat{G}^1(\mathbf{x}_i) = \phi$, $\tilde{G}^1(\mathbf{x}_i) = \phi$;
- 17: **for** $k = 1$ to \hat{M} **do**
- 18: sample λ_1 from $U(\beta_1, 1.0)$ and \mathbf{x}_j from $B^1(\mathbf{x}_i)$;
- 19: $\hat{G}^1(\mathbf{x}_i) = \hat{G}^1(\mathbf{x}_i) \cup \{\lambda_1 \mathbf{z}_i + (1 - \lambda_1) \mathbf{z}_j\}$;
- 20: **for** $k = 1$ to \tilde{M} **do**
- 21: sample λ_2 from $U(-\beta_2, \beta_2)$ and \mathbf{x}_j from $B^1(\mathbf{x}_i)$;
- 22: $\tilde{\mathbf{z}} = \lambda_2 \mathbf{z}_i + (1 - \lambda_2) \mathbf{z}_j$ and calculate $fp(\tilde{\mathbf{z}})$ by using Equation 5;
- 23: **if** $fp(\tilde{\mathbf{z}}) = 0$ **then**
- 24: $\tilde{G}^1(\mathbf{x}_i) = \tilde{G}^1(\mathbf{x}_i) \cup \tilde{\mathbf{z}}$;
- 25: construct set $C(\mathbf{x}_i) = A(\mathbf{x}_i) \cup \hat{G}^1(\mathbf{x}_i) \cup \tilde{G}^1(\mathbf{x}_i)$
- 26: construct set $D(\mathbf{x}_i) = B^1(\mathbf{x}_i) \cup \hat{G}^1(\mathbf{x}_i) \cup \tilde{G}^1(\mathbf{x}_i)$
- 27: calculate $l(\mathbf{z}_i, \mathbf{z}_p, C(\mathbf{x}_i))$ for each session $\mathbf{x}_p \in D(\mathbf{x}_i)$ by using Equation 7;
- 28: calculate \mathcal{L}_2 using Equation 6 and train the encoder;
- 29: **return** well trained encoder;

The ConRo training procedure is outlined in Algorithm 1. Let $\hat{M} = |\hat{G}^1(\mathbf{x}_i)|$ and $\tilde{M} \geq |\tilde{G}^1(\mathbf{x}_i)|$. Initially, we generate raw representations for all the sessions in the training set \mathcal{T} . In the first stage, we generate training batches

from \mathcal{T} . For each training batch $S = \{\mathbf{x}_i\}_{i=1}^R$, we create the corresponding auxiliary batch $S^1 = \{\mathbf{x}_i^1\}_{i=1}^M$ from \mathcal{T}^1 . We calculate $R_0 = \sum_{i=1}^R \mathbb{I}(y_i = 0)$ and $\mathbf{v}_0 = \frac{1}{R_0} \sum_{i=1}^R (1 - y_i) \mathbf{z}_i$ where, $\mathbb{I}(\cdot)$ is an indicator function. Then, for each normal session $\mathbf{x}_i \in S$, we construct the sets $A(\mathbf{x}_i) = (S \cup S^1) - \{\mathbf{x}_i\}$ and $B^0(\mathbf{x}_i) = \{\mathbf{x}_p \in A(\mathbf{x}_i) | y_p = 0\}$. For each session $\mathbf{x}_p \in B^0(\mathbf{x}_i)$, we calculate $l(\mathbf{z}_i, \mathbf{z}_p, A(\mathbf{x}_i))$ by using Equation 2. We calculate the batch supervised contrastive loss using Equation 1 and train the encoder by using this batch loss. Then, we calculate the batch DeepSVDD loss using Equation 3 and again train the encoder by using this batch loss.

In the second stage, we again generate training batches from \mathcal{T} . For each training batch $S = \{\mathbf{x}_i\}_{i=1}^R$, we create the auxiliary batch $S^1 = \{\mathbf{x}_i^1\}_{i=1}^M$ from \mathcal{T}^1 . For each malicious session $\mathbf{x}_i \in S$, we construct sets $A(\mathbf{x}_i) = (S \cup S^1) - \{\mathbf{x}_i\}$ and $B^1(\mathbf{x}_i) = \{\mathbf{x}_p \in A(\mathbf{x}_i) | y_p = 1\}$. We construct the set $\hat{G}^1(\mathbf{x}_i)$ consisting of \hat{M} potential malicious sessions which are similar to \mathbf{x}_i . Specifically, we take a sample λ_1 from $U(\beta_1, 1.0)$, we sample a session \mathbf{x}_j from $B^1(\mathbf{x}_i)$, and generate a similar potential malicious session by applying the operation $\lambda_1 \mathbf{z}_i + (1 - \lambda_1) \mathbf{z}_j$. We create another set $\tilde{G}^1(\mathbf{x}_i)$ consisting of a maximum of \tilde{M} diverse potential malicious sessions. Specifically, we take a sample λ_2 from $U(-\beta_2, \beta_2)$, we sample a session \mathbf{x}_j from $B^1(\mathbf{x}_i)$, and generate a diverse potential malicious session $\tilde{\mathbf{z}}$ by applying the operation $\lambda_2 \mathbf{z}_i + (1 - \lambda_2) \mathbf{z}_j$. Then, we calculate \mathbf{v}_0 and the radius of the normal session hyper-sphere r , and calculate $fp(\tilde{\mathbf{z}})$ using Equation 5. If $fp(\tilde{\mathbf{z}}) = 0$ then, $\tilde{\mathbf{z}}$ is considered as a true positive session and we include $\tilde{\mathbf{z}}$ into $\tilde{G}^1(\mathbf{x}_i)$. Then, we construct sets $C(\mathbf{x}_i) = A(\mathbf{x}_i) \cup \hat{G}^1(\mathbf{x}_i) \cup \tilde{G}^1(\mathbf{x}_i)$ and $D(\mathbf{x}_i) = B^1(\mathbf{x}_i) \cup \hat{G}^1(\mathbf{x}_i) \cup \tilde{G}^1(\mathbf{x}_i)$. For each session $\mathbf{x}_p \in D(\mathbf{x}_i)$, we calculate $l(\mathbf{z}_i, \mathbf{z}_p, C(\mathbf{x}_i))$ by using Equation 7. We calculate the batch loss by using \mathcal{L}_2 shown in Equation 6 and train the encoder by using this batch loss. Finally, the training algorithm returns the well trained encoder.

Time complexity analysis. We analyze the time complexity of our ConRo training procedure by considering the forward pass and the number of times the individual loss $l(\cdot, \cdot, \cdot)$ is invoked in both stages. This time complexity is given by: $O(|\mathcal{T}^0|R + |\mathcal{T}^1|(M + |\hat{G}^1(\mathbf{x}_i)| + |\tilde{G}^1(\mathbf{x}_i)|))$.

Inference. After the second stage training, our encoder has learnt to push seen malicious sessions, similar and diverse potential malicious sessions closer in the encoded representation space, and as a consequence, all these sessions form a tight cluster in the encoded representation space (refer to Figure 1d). Normal sessions are also tightly clustered in the encoded representation space due to the effect of first stage training. Hence, we design our inference strategy by analyzing the proximities of a test case session to the centers of normal and malicious sessions in the encoded representation space. Let \mathbf{v}_1 denote the estimated center of malicious sessions in the encoded representation space, which is given by $\mathbf{v}_1 = \frac{1}{M} \sum_{i=1}^M \mathbf{z}_i^1$, where $\{\mathbf{x}_i^1\}_{i=1}^M$ denotes M randomly sampled malicious sessions from \mathcal{T}^1 . For any test case session \mathbf{x} , ConRo predicts its label as:

$$label(\mathbf{x}) = \begin{cases} 1 & \text{if } \|\mathbf{z} - \mathbf{v}_1\|_2 < \|\mathbf{z} - \mathbf{v}_0\|_2 \\ 0 & \text{otherwise} \end{cases}$$

3.4 Theoretical Analysis

We present a theoretical analysis study to highlight the important factors which influence the generalization performance of our ConRo framework. We introduce a set definition called $\epsilon\text{-span}(\mathbf{x}_j)$ for a session \mathbf{x}_j which is defined as:

$$\epsilon\text{-span}(\mathbf{x}_j) = \left\{ \mathbf{x}_k \left| \|\mathbf{z}_k - \mathbf{z}_j\|_2 \leq \epsilon, P \left(\|\mathbf{v}_y - \mathbf{z}_k\|_2 < \|\mathbf{v}_{1-y} - \mathbf{z}_k\|_2 \mid \|\mathbf{v}_y - \mathbf{z}_j\|_2 < \|\mathbf{v}_{1-y} - \mathbf{z}_j\|_2 \right) = 1 \right. \right\} \quad (8)$$

Here, $y = \{0, 1\}$, $\epsilon\text{-span}(\mathbf{x}_j)$ contains those sessions \mathbf{x}_k such that $\|\mathbf{z}_k - \mathbf{z}_j\|_2 \leq \epsilon$ for some $\epsilon \geq 0$, and our encoder has similar session projection action w.r.t to proximities between \mathbf{v}_0 and \mathbf{v}_1 for both \mathbf{x}_k and \mathbf{x}_j , which is as shown in the right hand side second term of Equation 8. We extend the definition of $\epsilon\text{-span}(\mathbf{x}_j)$ to a set of sessions \mathcal{S} called $\epsilon\text{-span}(\mathcal{S})$. This set definition is given by:

$$\epsilon\text{-span}(\mathcal{S}) = \left\{ \mathbf{x}_k \left| \exists \mathbf{x}_j \in \mathcal{S}, \|\mathbf{z}_k - \mathbf{z}_j\|_2 \leq \epsilon, P \left(\|\mathbf{v}_y - \mathbf{z}_k\|_2 < \|\mathbf{v}_{1-y} - \mathbf{z}_k\|_2 \mid \|\mathbf{v}_y - \mathbf{z}_j\|_2 < \|\mathbf{v}_{1-y} - \mathbf{z}_j\|_2 \right) = 1 \right. \right\}$$

Here, $\epsilon\text{-span}(\mathcal{S})$ contains those sessions \mathbf{x}_k such that there exists some session $\mathbf{x}_j \in \mathcal{S}$ where $\|\mathbf{z}_k - \mathbf{z}_j\|_2 \leq \epsilon$ for some $\epsilon \geq 0$, and our encoder has similar session projection action w.r.t to proximities between \mathbf{v}_0 and \mathbf{v}_1 for both \mathbf{x}_k and \mathbf{x}_j . For our theoretical analysis, we assume an existence of a hypothetical oracle version of our encoder which is trained by using the labeled sessions sampled from the test distribution \mathcal{D} and supervised contrastive loss function.

Let \mathbf{v}_0^o and \mathbf{v}_1^o denote the centers of normal and malicious sessions in the encoded representation space corresponding to this oracle encoder, respectively. For a test case session \mathbf{x} , the oracle encoder has the following properties:

$$P(\|\mathbf{v}_1^o - \mathbf{z}\|_2 < \|\mathbf{v}_0^o - \mathbf{z}\|_2) = \begin{cases} 1, & \text{if } y = 1 \\ 0, & \text{otherwise} \end{cases}$$

Theorem 1. For any test case session \mathbf{x} which is sampled from \mathcal{D} , the following bound holds:

$$P\left(\|\mathbf{v}_y - \mathbf{z}\|_2 < \|\mathbf{v}_{1-y} - \mathbf{z}\|_2 \mid \|\mathbf{v}_y^o - \mathbf{z}\|_2 < \|\mathbf{v}_{1-y}^o - \mathbf{z}\|_2\right) \geq P\left(\mathbf{x} \in \bigcup_{\mathbf{x}_i \in \mathcal{T}^1} \epsilon\text{-span}\left(\widehat{G}^1(\mathbf{x}_i) \cup \mathbf{x}_i\right) \cup \epsilon\text{-span}\left(\widetilde{G}^1(\mathbf{x}_i)\right)\right)$$

Proof. Case when $y = 1$. In this case: $P(\|\mathbf{v}_1^o - \mathbf{z}\|_2 < \|\mathbf{v}_0^o - \mathbf{z}\|_2) = 1$. Now our encoder can either project \mathbf{x} closer to \mathbf{v}_1 or \mathbf{v}_0 . If it projects closer to \mathbf{v}_1 which means that $\|\mathbf{v}_1 - \mathbf{z}\|_2 < \|\mathbf{v}_0 - \mathbf{z}\|_2$. Then, there are two scenarios. In the first scenario, we have that:

$$\mathbf{x} \in \bigcup_{\mathbf{x}_i \in \mathcal{T}^1} \epsilon\text{-span}\left(\widehat{G}^1(\mathbf{x}_i) \cup \mathbf{x}_i\right) \cup \epsilon\text{-span}\left(\widetilde{G}^1(\mathbf{x}_i)\right)$$

For a set of sessions \mathcal{S} , we have defined $\epsilon\text{-span}(\mathcal{S})$ as:

$$\epsilon\text{-span}(\mathcal{S}) = \left\{ \mathbf{x}_k \mid \exists \mathbf{x}_j \in \mathcal{S}, \|\mathbf{z}_k - \mathbf{z}_j\|_2 \leq \epsilon, P\left(\|\mathbf{v}_y - \mathbf{z}_k\|_2 < \|\mathbf{v}_{1-y} - \mathbf{z}_k\|_2 \mid \|\mathbf{v}_y - \mathbf{z}_j\|_2 < \|\mathbf{v}_{1-y} - \mathbf{z}_j\|_2\right) = 1 \right\}$$

By only considering the first scenario and by the definition of $\epsilon\text{-span}(\mathcal{S})$, we have the result:

$$P\left(\|\mathbf{v}_1 - \mathbf{z}\|_2 < \|\mathbf{v}_0 - \mathbf{z}\|_2 \mid \|\mathbf{v}_1^o - \mathbf{z}\|_2 < \|\mathbf{v}_0^o - \mathbf{z}\|_2\right) = P\left(\mathbf{x} \in \bigcup_{\mathbf{x}_i \in \mathcal{T}^1} \epsilon\text{-span}\left(\widehat{G}^1(\mathbf{x}_i) \cup \mathbf{x}_i\right) \cup \epsilon\text{-span}\left(\widetilde{G}^1(\mathbf{x}_i)\right)\right)$$

In the alternate scenario, we have that:

$$\mathbf{x} \notin \bigcup_{\mathbf{x}_i \in \mathcal{T}^1} \epsilon\text{-span}\left(\widehat{G}^1(\mathbf{x}_i) \cup \mathbf{x}_i\right) \cup \epsilon\text{-span}\left(\widetilde{G}^1(\mathbf{x}_i)\right)$$

By considering both these scenarios, we have the result:

$$P\left(\|\mathbf{v}_1 - \mathbf{z}\|_2 < \|\mathbf{v}_0 - \mathbf{z}\|_2 \mid \|\mathbf{v}_1^o - \mathbf{z}\|_2 < \|\mathbf{v}_0^o - \mathbf{z}\|_2\right) \geq P\left(\mathbf{x} \in \bigcup_{\mathbf{x}_i \in \mathcal{T}^1} \epsilon\text{-span}\left(\widehat{G}^1(\mathbf{x}_i) \cup \mathbf{x}_i\right) \cup \epsilon\text{-span}\left(\widetilde{G}^1(\mathbf{x}_i)\right)\right)$$

Case when $y = 0$. In this case, $P(\|\mathbf{v}_0^o - \mathbf{z}\|_2 \leq \|\mathbf{v}_1^o - \mathbf{z}\|_2) = 1$. Since our encoder is trained by using a large number of normal sessions in \mathcal{T} and due to which, we can sufficiently cover the diversity in normal sessions, we have that: $P\left(\|\mathbf{v}_0 - \mathbf{z}\|_2 < \|\mathbf{v}_1 - \mathbf{z}\|_2 \mid \|\mathbf{v}_0^o - \mathbf{z}\|_2 < \|\mathbf{v}_1^o - \mathbf{z}\|_2\right) \approx 1$. Thus, the theorem immediately follows. \square

Theorem 1 outlines a formal explanation on the generalization performance of ConRo. Clearly, this performance is influenced by $P\left(\mathbf{x} \in \bigcup_{\mathbf{x}_i \in \mathcal{T}^1} \epsilon\text{-span}\left(\widehat{G}^1(\mathbf{x}_i) \cup \mathbf{x}_i\right) \cup \epsilon\text{-span}\left(\widetilde{G}^1(\mathbf{x}_i)\right)\right)$. This probability value can be increased by: (1) setting a suitable value for β_2 based on empirical analysis, and (2) setting a large value for $|\widetilde{G}^1(\mathbf{x}_i)|$ ensures that generated diverse potential malicious sessions can span diverse regions of the encoded representation space.

4 Experiments

We describe our experimental setup¹ including datasets and baselines used in this paper and then discuss our experimental results including hyper-parameter sensitivity, visualization, training latency, and ablation analysis results.

4.1 Experimental Setup

4.1.1 Datasets

We use three benchmark fraud detection datasets: CERT [Glasser and Lindauer, 2013], UMD-Wikipedia [Kumar et al., 2015], and OpenStack [Du et al., 2017].

CERT [Glasser and Lindauer, 2013]. The CERT dataset is a comprehensive dataset for insider threat detection. There are 48 malicious and 1,581,358 normal sessions. The insider sessions are chronologically recorded over 516 days. To avoid extreme training latency, we randomly sample 10,000 normal sessions from the first 460 days, and include them in the training set \mathcal{T} . Similarly, we randomly sample 500 normal sessions from 461 to 516 days to construct our test set. There are 5 types of malicious sessions. (1) *Logon*: The insider logs on a computer during weekends or on a weekday after work hours. (2) *Email*: The insider sends/views unexpected emails to/from external sources. (3) *HTTP*: The insider uploads/downloads organizational information to/from external malicious websites. (4) *Device*: The insider connects a device such as removable drives during weekends or on a weekday after work hours. (5) *File*: The insider manipulates organizational files with malicious intentions. We construct a biased training set corresponding to malicious sessions wherein, we include device, email, and file malicious session types in the training set and remaining two types in the test set. Specifically, we include 30 and 18 malicious sessions in the training and test sets, respectively.

UMD-Wikipedia [Kumar et al., 2015]. This dataset consists of activity sessions of a set of users who have edited the Wikipedia website. In this dataset, there are 5486 normal and 4627 malicious sessions. We randomly sample 1000 normal sessions to construct the test set and include all the remaining 4486 normal sessions in the training set. For the malicious sessions, in-order to simulate open-set and imbalanced dataset scenario, we construct the training set by leveraging and suitably adapting the procedure utilized by Du and Wu [2021], which is described below. We calculate the appropriate number of malicious session clusters (K) in the available malicious sessions by using *silhouette coefficient analysis* [Rousseeuw, 1987]. From our empirical study, we get $K = 3$. Then, we randomly sample 70 and 10 malicious sessions from the first and second malicious session clusters, respectively, and include them in the training set. From the remaining malicious sessions, we randomly sample similar number of malicious sessions from each of the 3 clusters to construct the test set which contains 500 malicious sessions.

OpenStack [Du et al., 2017]. This dataset records the activity sessions of users who have used the OpenStack cloud services. In this dataset, there are 244,908 normal and 18,434 malicious sessions. We randomly sample 10,000 and 1000 normal sessions and include them in our training and test sets, respectively. For the malicious sessions, through silhouette coefficient analysis we get $k = 12$ malicious session clusters. We randomly sample 50 and 10 malicious sessions from the first and second malicious session clusters, respectively, and include them in the training set. From the remaining malicious sessions, we construct a test set having 120 malicious sessions by randomly sampling equal number of malicious sessions from each of the 12 malicious session clusters.

4.1.2 Training Details

By considering a user activity session as a sentence, we train the word-to-vector model [Mikolov et al., 2013] to derive the activity representation. The minimum activity frequency is set as 1 since every activity is given importance in the design. To effectively train our session encoder, we set the number of dimensions of the activity and session representations as $d = 50$. Since we generate encoded session representation by averaging the output sequence of the LSTM model, we set the hidden layer size of LSTM to 50. The temperature parameter α shown in Equations 2 and 7 is set to its default value 1. We opt for medium sized training batches in order to avoid extreme memory requirements during encoder training. Specifically, we use 100 sessions (R) in each training batch. We set the size of the malicious session auxiliary batch (M) as 20. The sizes of potential malicious session batches $|\hat{G}^1(\mathbf{x}_i)|$ and $|\tilde{G}^1(\mathbf{x}_i)|$ are set as 20 and 200, respectively. For β_1 , we set its value as 0.92 because it is supposed to be closer to 1. For β_2 , we set its value as 4 in order to generate potential malicious sessions which are sufficiently diverse. Additionally, we perform a sensitivity analysis study on β_2 which is described in Section 4.2.2. We use the Adam optimizer [Kingma and Ba, 2015] with a learning rate of 0.005 and we use 10 training epochs for both stages. We utilize three metrics to measure the anomaly detection performance: F_1 , False Positive Rate (FPR), and Area Under the Receiver Operating

¹Code is available through the hyperlink: [Code Link](#).

Characteristics Curve (AUC-ROC). We report the mean and standard deviation of performance scores after 5 times of running.

4.1.3 Baselines

We compare our ConRo framework with five state-of-the-art baselines which were specifically designed for anomaly detection: DeepSVDD [Ruff et al., 2018], DeepSAD [Ruff et al., 2020], DevNet [Pang et al., 2019, 2021a], CLDet [Vinay et al., 2022], and Swan [Ding et al., 2022]. All these baselines operate in the setting where only a few anomalous samples are available. DeepSVDD, DeepSAD, DevNet, and CLDet have been designed for closed set anomaly detection whereas, Swan has been designed for open-set anomaly detection. Specifically, CLDet is a self-supervised contrastive learning based insider threat detection framework. Except CLDet, the remaining baselines originally operate on image datasets, and employ neural networks for image data such as CNN [Ruff et al., 2020], ResNet-18 [Ding et al., 2022] etc. Hence, they cannot be directly applied for our fraud detection task which operates on sequential data. We replace their neural networks with our LSTM-based session encoder and adapt these baselines to our fraud detection task. We employ the same training set used for our ConRo to train all these baselines. For Swan, the original augmentation technique is image-specific. Hence, we replace it with the augmentation technique proposed by Verma et al. [2021]. Additionally, the unseen malicious sessions are detected in a residual representation space which is defined as: $\mathbf{v}_0 - \mathbf{z}$.

4.2 Experimental Results

4.2.1 Overall Comparison

Table 1: Performances of our ConRo and baselines (mean \pm std). The higher the better for F1 and AUC-ROC. The lower the better for FPR. The best values are bold highlighted.

Models	CERT			UMD-Wikipedia			Open-Stack		
	F1	FPR	AUC-ROC	F1	FPR	AUC-ROC	F1	FPR	AUC-ROC
DeepSVDD	14.67 \pm 4.1	14.30 \pm 1.8	62.29 \pm 4.8	33.23 \pm 1.7	44.90 \pm 1.4	46.47 \pm 0.8	32.27 \pm 0.7	42.10 \pm 1.4	79.02 \pm 0.7
DeepSAD	24.71 \pm 7.5	20.53 \pm 7.1	84.17 \pm 3.6	56.88 \pm 2.9	13.30 \pm 0.2	68.35 \pm 1.7	67.43 \pm 3.1	9.70 \pm 1.3	94.12 \pm 0.1
CLDet	60.41 \pm 3.6	3.75 \pm 1.9	79.47 \pm 2.6	58.78 \pm 3.1	9.59 \pm 2.8	70.71 \pm 2.4	61.71 \pm 2.9	6.18 \pm 2.1	83.98 \pm 1.8
Swan	59.31 \pm 2.2	0.0\pm0.0	72.12 \pm 0.1	57.02 \pm 0.9	0.0\pm0.0	69.89 \pm 0.5	62.93 \pm 4.2	0.0\pm0.0	73.10 \pm 2.3
ConRo	68.33\pm3.9	2.20 \pm 0.5	90.50\pm0.3	71.40\pm2.3	31.50 \pm 2.1	79.50\pm2.1	77.56\pm2.3	5.80 \pm 0.8	97.10\pm0.4

The performance of our ConRo framework and baselines for all datasets are shown in Table 1. Clearly, our ConRo outperforms all baselines² w.r.t most of the performance metrics. These baselines do not learn effective class-specific shared features in the encoded representation space. Thus, due to the combined challenges of session diversity, dataset imbalance, and biased malicious training samples, they fail to provide noticeable results. However, ConRo addresses all these challenges effectively. It addresses the session diversity challenge through supervised contrastive learning. It addresses the dataset imbalance challenge by generating a large amount of augmented/potential malicious sessions. Finally, it addresses the challenge of biased malicious training samples by generating diverse potential malicious sessions.

For the UMD-Wikipedia dataset, Swan noticeably outperforms our ConRo w.r.t FPR score. The mechanisms of ConRo and Swan are different. Specifically, Swan learns to identify unseen malicious sessions in a residual representation space ($\mathbf{v}_0 - \mathbf{z}$) whereas, ConRo learns to identify unseen malicious sessions by generating a large amount of diverse potential malicious sessions. In UMD-Wikipedia dataset, many normal sessions share close similarities with unseen malicious sessions. Therefore, ConRo identifies some of the test normal sessions sharing close similarities with test malicious sessions as malicious (false positive) which negatively impacts the FPR scores of ConRo. However, Swan does not specifically address the session diversity challenge in malicious sessions due to which, Swan under-performs against ConRo w.r.t F1 and AUCROC scores.

4.2.2 Sensitivity Analysis

For analyzing the sensitivity³ of the hyper-parameter β_2 , we first perform first stage training of our encoder. Then by employing this first stage trained encoder, we further perform second stage training of our encoder separately corresponding to different β_2 values. Our sensitivity analysis results are shown in Figure 3. Clearly, ConRo is not

²Since DevNet classifies all test sessions as normal, we have not shown its performance scores. Devnet does not employ any augmented malicious sessions for its training, so it cannot effectively address the dataset imbalance challenge.

³We don't perform sensitivity analysis on β_1 because it is constrained to be set closer to 1 [Verma et al., 2021].

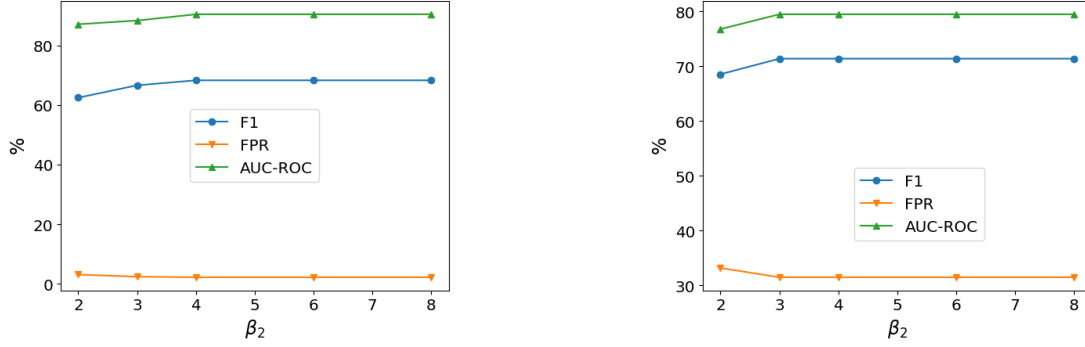


Figure 3: Sensitivity analysis results w.r.t β_2 on CERT (left column) and UMD-Wikipedia (right column).

highly sensitive to hyper-parameter β_2 . It only suffers slightly when β_2 is low and performance values converge for higher values. For example in the CERT dataset, for $\beta_2 \geq 4$, the performance values converge. β_2 controls the diversity aspect of potential malicious sessions. The test set malicious sessions are not extremely diverse from their training set counterparts. Hence, generating extremely diverse potential malicious sessions does not aid in improving the generalization performance on the test set. In certain datasets, where the test case malicious sessions are extremely diverse when compared to their training set counterparts, we expect that higher values of β_2 can aid in improving the generalization performance.

4.2.3 Visualization Analysis

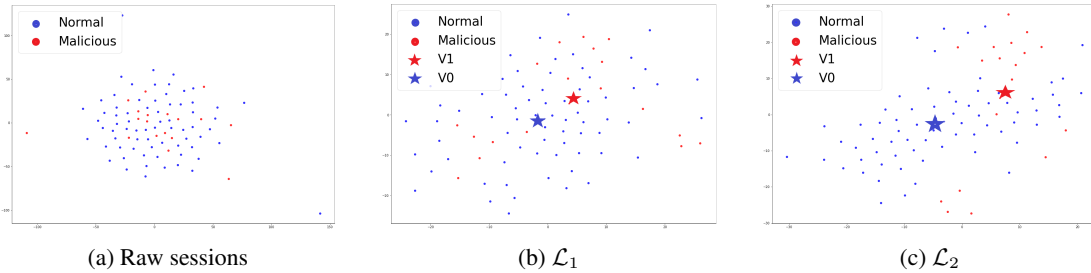


Figure 4: Visualization of test session representations for CERT dataset. Blue and red dots denote normal and malicious sessions, respectively. Blue and red stars denote v_0 and v_1 , respectively.

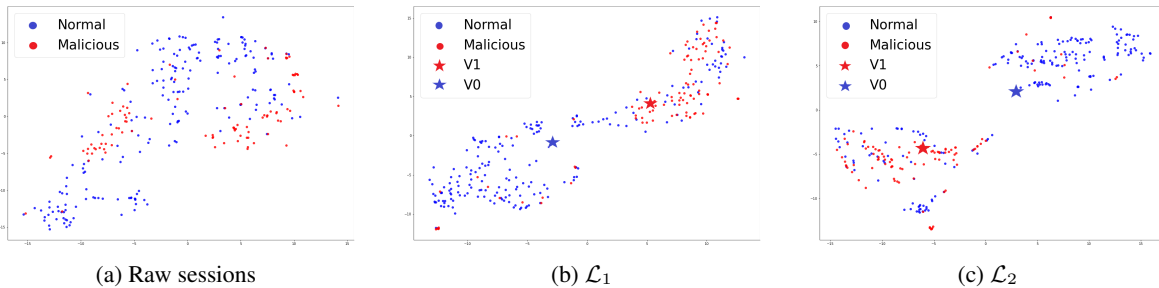


Figure 5: Visualization of test session representations for UMD-Wikipedia dataset.

We employ the t-SNE technique [van der Maaten and Hinton, 2008] for visualization. We consider two different encoded session representations based on whether our encoder is trained by \mathcal{L}_1 (stage 1) or \mathcal{L}_2 (stage 2). For the CERT dataset, we randomly sample 70 normal sessions from the test set and utilize all test malicious sessions for visualization. The visualization results for the CERT dataset are shown in Figure 4. After stage 1 training (refer to Figure 4b), there are many malicious sessions that overlap the normal session cluster. Also, v_1 and v_0 are closer to each other. The reason is that during stage 1 training, our encoder does not get an opportunity to contrast between unseen malicious and normal sessions. After stage 2 training (refer to Figure 4c), v_1 and v_0 get well separated. During

stage 2 training, we train our encoder by employing both similar and diverse potential malicious sessions. Therefore, our encoder learns separable representations w.r.t the open-set fraud detection task.

For the UMD-Wikipedia dataset, we randomly sample 200 normal and 100 malicious sessions from the test set. The visualization results for UMD-Wikipedia dataset are shown in Figure 5. After stage 1 training, as seen in Figure 5b, even though \mathbf{v}_1 and \mathbf{v}_0 are not closer, the malicious and normal session clusters are not well separated. This visualization result demonstrates that stage 1 is not sufficient enough to obtain a good separation between normal and malicious sessions in the encoded representation space. After stage 2 training, as seen in Figure 5c, both malicious and normal session clusters get well separated.

4.3 Training Latency Analysis

Table 2: Training latencies of our ConRo and baselines.

Models	Training Latency (seconds)	
	CERT	UMD-Wikipedia
DeepSVDD	2463	1322
DeepSAD	3842	2164
DevNet	1741	844
CLDet	6548	3864
Swan	3920	2228
ConRo	40143	25080

All experiments are executed on AMD EPYC (2.3 GHz) CPU server with 26 GB RAM and 226 GB hard disk. We use 150 epochs to train all baselines except CLDet. Specifically, CLDet has pre-training and fine tuning components which are trained by employing 10 and 150 epochs, respectively. The training latencies of our ConRo framework and baselines for both CERT and UMD-Wikipedia datasets are shown in Table 2. ConRo incurs substantially more training cost than other baselines. The reason being that ConRo employs supervised contrastive loss which is the primary factor for this observed high training costs. However, supervised contrastive learning enables our session encoder to learn class-specific shared features, and effectively address the session diversity challenge. Hence, ConRo is able to deliver better performance than other baselines.

4.3.1 Ablation Analysis

Table 3: Ablation analysis results (mean \pm std).

Models	CERT			UMD-Wikipedia			Open-Stack		
	F1	FPR	AUC-ROC	F1	FPR	AUC-ROC	F1	FPR	AUC-ROC
w/o stage 1	18.33 \pm 1.2	25.10 \pm 0.3	78.56 \pm 1.1	40.23 \pm 1.3	28.37 \pm 1.9	55.55 \pm 1.1	14.92 \pm 1.7	47.14 \pm 2.3	49.43 \pm 2.9
w/o \mathcal{L}^{Sup}	5.28 \pm 0.2	48.10 \pm 1.8	45.44 \pm 0.9	53.16 \pm 2.3	25.10 \pm 1.9	64.65 \pm 1.9	15.12 \pm 1.4	46.90 \pm 2.7	49.78 \pm 2.4
w/o \mathcal{L}^{SV}	20.10 \pm 1.1	27.05 \pm 1.3	83.72 \pm 0.7	64.95 \pm 0.8	18.85 \pm 6.1	73.72 \pm 0.5	38.76 \pm 0.8	31.60 \pm 1.1	84.20 \pm 0.4
w/o AO	8.99 \pm 0.4	68.46 \pm 3.1	62.98 \pm 1.6	52.04 \pm 1.4	80.30 \pm 1.4	55.68 \pm 1.9	14.08 \pm 2.1	26.16 \pm 1.4	50.59 \pm 2.3
w/o stage 2	42.86 \pm 1.1	0.0 \pm 0.0	63.84 \pm 0.1	60.90 \pm 1.1	23.85 \pm 1.6	70.42 \pm 0.6	46.11 \pm 3.4	0.0 \pm 0.0	65.40 \pm 1.6
w/o $fp(\cdot)$	31.32 \pm 5.1	26.66 \pm 6.3	72.77 \pm 3.1	59.38 \pm 1.8	65.52 \pm 4.2	65.95 \pm 2.6	37.50 \pm 3.7	49.60 \pm 3.6	48.16 \pm 3.5
w/o $\hat{G}^1(\cdot)$	55.92 \pm 1.2	4.13 \pm 0.3	89.49 \pm 0.2	65.58 \pm 2.6	31.20 \pm 0.3	74.04 \pm 2.3	67.57 \pm 1.1	9.60 \pm 0.4	95.20 \pm 0.2
w/o $\tilde{G}^1(\cdot)$	44.17 \pm 1.9	7.10 \pm 0.6	88.16 \pm 0.3	63.40 \pm 0.8	31.70 \pm 4.7	72.10 \pm 0.6	52.26 \pm 1.6	18.10 \pm 1.4	90.61 \pm 0.2

We conduct the ablation analysis study on our ConRo framework by ablating the following main components: stage 1, \mathcal{L}^{Sup} , \mathcal{L}^{SV} , Alternating Optimization (AO), stage 2, $fp(\cdot)$ (optimistic choice), $\hat{G}^1(\cdot)$, and $\tilde{G}^1(\cdot)$. The ablation analysis results are shown in Table 3.

W/o stage 1. Mean F1 scores drop to 18.33 (CERT), 40.23 (UMD-Wikipedia), and 14.92 (Open-Stack). Stage 1 ensures that the encoder learns shared features for normal sessions. Without learning these shared features, the encoder fails to achieve tight class-specific clusters in the encoded representation space.

W/o \mathcal{L}^{Sup} . Mean F1 scores drop to 5.28 (CERT), 53.16 (UMD-Wikipedia), and 15.12 (Open-Stack). Both normal and malicious sessions typically exhibit large diversity and \mathcal{L}^{Sup} is essential to address this session diversity challenge. We can see that there is a significant drop in F1 scores on CERT and Open-Stack datasets but not in the case for UMD-Wikipedia dataset. We can attribute the reason to the different characteristics of these datasets. Addressing the session diversity challenge for the normal sessions is much more critical in both CERT and Open-Stack datasets than in the UMD-Wikipedia dataset.

W/o \mathcal{L}^{SV} . Mean F1 scores drop to 20.10 (CERT), 64.95 (UMD-Wikipedia), and 38.76 (Open-Stack). The DeepSVDD loss \mathcal{L}^{SV} enables the encoder to push normal sessions in a minimum volume hyper-sphere in the encoded representation space. Without this topological effect, the efficacy of stage 2 reduces because the generated diverse potential malicious sessions do not effectively cover unseen malicious sessions.

W/o AO . By employing the joint optimization approach, mean F1 scores drop to 8.99 (CERT), 52.04 (UMD-Wikipedia), and 14.08 (Open-Stack). Optimizing DeepSVDD objective (\mathcal{L}^{SV}) can yield maximum benefits only when the input normal sessions have considerable shared features in the encoded representation space. Here, we jointly optimize both supervised contrastive (\mathcal{L}^{Sup}) and DeepSVDD objectives, and we do not specifically provide normal sessions having considerable shared features in the encoded representation space as inputs to the DeepSVDD objective. As a consequence, we can observe a significant drop in the performance.

W/o stage 2 . Mean F1 scores drop to 42.86 (CERT), 60.90 (UMD-Wikipedia), and 46.11 (Open-Stack). In stage 1 training, our encoder learns to contrast normal sessions with few available malicious sessions having limited diversity. Stage 2 generates diverse potential malicious sessions which can be similar to unseen malicious sessions w.r.t their encoded representations. As a consequence, our encoder can learn effective separable encoded representations.

W/o $fp(\cdot)$. By employing the pessimistic choice, mean F1 scores drop to 31.32 (CERT), 59.38 (UMD-Wikipedia), and 37.50 (Open-Stack). Without employing $fp(\cdot)$, the encoder learns to push malicious sessions and those potential malicious sessions which are false positives, closer in the encoded representation space. Due to this improper learning effect, the encoder does not achieve effective separable encoded representations.

W/o $\hat{G}^1(\cdot)$. Mean F1 scores drop to 55.92 (CERT), 65.58 (UMD-Wikipedia), and 67.57 (Open-Stack). Generating similar potential malicious sessions which are similar to a seen malicious session in the encoded representation space, aids the encoder to learn more effective separable representations.

W/o $\tilde{G}^1(\cdot)$. Mean F1 scores drop to 44.17 (CERT), 63.40 (UMD-Wikipedia), and 52.26 (Open-Stack). Generating diverse potential malicious sessions which can be similar to unseen malicious sessions in the encoded representation space, aids the encoder to effectively contrast normal sessions with unseen malicious sessions.

5 Conclusion

In this work, we have developed a robust and open-set fraud detection framework called ConRo, which is specifically designed to operate in the scenario where only **a few malicious sessions having limited diversity are available for training**. We developed a training procedure for ConRo to learn separable session representations by employing effective **data augmentation strategies** and by the combined effect of **supervised contrastive and DeepSVDD losses**. We presented a theoretical analysis study to analyze the **main factors** influencing the **generalization performance** of ConRo. The empirical study on three benchmark datasets demonstrated that our ConRo can outperform state-of-the-art baselines. **In our future work, we plan to extend ConRo to address specific distribution shift scenarios such as sample selection bias. We will study how to integrate bias correction approaches with supervised contrastive learning.**

Acknowledgement

This work was supported in part by NSF grants 1920920, 1946391 and 2103829.

References

- Shuhan Yuan and Xintao Wu. Deep learning for insider threat detection: Review, challenges and opportunities. *Computers & Security*, 2021.
- Shuhan Yuan, Panpan Zheng, Xintao Wu, and Hanghang Tong. Few-shot insider threat detection. In *The 29th ACM International Conference on Information and Knowledge Management*, 2020.
- Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.
- Lukas Ruff, Robert A. Vandermeulen, Nico Görnitz, Alexander Binder, Emmanuel Müller, Klaus-Robert Müller, and Marius Kloft. Deep semi-supervised anomaly detection. In *8th International Conference on Learning Representations*, 2020.
- Guansong Pang, Choubo Ding, Chunhua Shen, and Anton van den Hengel. Explainable deep few-shot anomaly detection with deviation networks. *CoRR*, abs/2108.00462, 2021a.

- Guansong Pang, Chunhua Shen, and Anton van den Hengel. Deep anomaly detection with deviation networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.
- Choubo Ding, Guansong Pang, and Chunhua Shen. Catching both gray and black swans: Open-set supervised anomaly detection. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems*, 2020.
- Joshua Glasser and Brian Lindauer. Bridging the gap: A pragmatic approach to generating insider threat data. In *IEEE Symposium on Security and Privacy Workshops*, 2013.
- Srikanth Kumar, Francesca Spezzano, and V.S. Subrahmanian. Vews: A wikipedia vandal early warning system. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.
- Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, CCS*, 2017.
- Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for anomaly detection: A review. *ACM Comput. Surv.*, 2021b.
- Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, 2021.
- Guansong Pang, Longbing Cao, Ling Chen, and Huan Liu. Learning representations of ultrahigh-dimensional data for random distance-based outlier detection. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.
- M. S. Vinay, Shuhan Yuan, and Xintao Wu. Contrastive learning for insider threat detection. In *Database Systems for Advanced Applications - 27th International Conference*, 2022.
- Shaolei Zhou, Liming Wang, Jing Yang, and Pengwei Zhan. SITD: insider threat detection using siamese architecture on imbalanced data. In *IEEE 25th International Conference on Computer Supported Cooperative Work in Design*, 2022.
- Haixuan Guo, Shuhan Yuan, and Xintao Wu. Logbert: Log anomaly detection via BERT. *CoRR*, abs/2103.04475, 2021.
- Jiaying Qi, Zhongzhi Luan, Shaohan Huang, Yukun Wang, Carol Fung, Hailong Yang, and Depei Qian. Adanomaly: Adaptive anomaly detection for system logs with adversarial learning. In *IEEE/IFIP Network Operations and Management Symposium*, 2022.
- Mohammadreza Salehi, Hossein Mirzaei, Dan Hendrycks, Yixuan Li, Mohammad Hossein Rohban, and Mohammad Sabokrou. A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges. *CoRR*, abs/2110.14051, 2021.
- Guansong Pang, Anton van den Hengel, Chunhua Shen, and Longbing Cao. Toward deep supervised anomaly detection: Reinforcement learning from partially labeled anomaly data. In *The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2021c.
- Dan Hendrycks, Mantas Mazeika, and Thomas G. Dietterich. Deep anomaly detection with outlier exposure. In *7th International Conference on Learning Representations*, 2019.
- Guansong Pang, Chunhua Shen, Huidong Jin, and Anton van den Hengel. Deep weakly-supervised anomaly detection. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023.
- Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *CoRR*, abs/2011.00362, 2020.
- Hyunjun Ju, Dongha Lee, Junyoung Hwang, Junghyun Namkung, and Hwanjo Yu. Pumat: Pu metric learning for anomaly detection. *Information Sciences*, 2020.
- Vikas Verma, Thang Luong, Kenji Kawaguchi, Hieu Pham, and Quoc V. Le. Towards domain-agnostic contrastive learning. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.
- Wei Du and Xintao Wu. Fair and robust classification under sample selection bias. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021.
- Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 1987.

- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*, 2015.
- Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 2008.