Robust Fraud Detection via Supervised Contrastive Learning

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

2023 IEEE International Conference on Big Data (BigData) | 979-8-3503-2445-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/BigData59044.2023.10386925

***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 the 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.**

***Index Terms*—fraud detection; contrastive learning; open-set; augmentation.**

I. 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 an 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 [1]. 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 [1], [2]. 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 [1]. 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 [3], [4] or deviation loss based learning [5], [6]. 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 anomalies encountered during the testing stage deviate from the training set anomalies [7]. To address this challenge, Ding et al. [7] recently 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 some 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. [8] presented a supervised contrastive learning model which extends contrastive learning to the supervised setting. The main goal is to push 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) [3] 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 [9], UMD-Wikipedia [10], and Open-stack [11] in which, we show superior performance of our ConRo framework over state-of-the-art baselines.

II. Related Work

Anomaly Detection. Anomaly detection is to detect data that significantly deviate from the majority of data [12]. 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 [12], [13]. 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 [3], [4], [14]. Pang et al. [5], [6] 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 the closed-set anomaly detection task. In our empirical analysis study, we select some of these approaches [3]–[6] as baselines, and show that they fail to deliver noticeable results on the 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 [1] 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 insider threat detection approaches [2], [15]–[18] 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. [19] have extensively discussed about contemporary OSR approaches. Pang et al. [20] have addressed anomaly detection in the OSR scenario by employing unlabelled samples. Hendrycks et al. [21] 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 [20] or unseen [21] malicious sessions for learning. Ding et al. [7] 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 [7] as a baseline, and show that it under-performs on open-set fraud detection task. Recently, Pang et al. [22] 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. [23] presented an in-depth discussions on 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.

III. 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 representation space 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[1]. 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 [25]. 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.

A. 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 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 normal and seen malicious sessions. Then, to achieve the second goal, we leverage the DeepSVDD loss [3], 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. [8], which is given by:

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

Here, the set $A(\mathbf{x}_i)$ is defined as $\left(S \cup S^1\right) - \{\mathbf{x}_i\}$, 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 $l\left(\mathbf{z}_i, \mathbf{z}_p, A(\mathbf{x}_i)\right)$ between the pair $(\mathbf{x}_i, \mathbf{x}_p)$ is defined as:

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

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. [3]. 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:

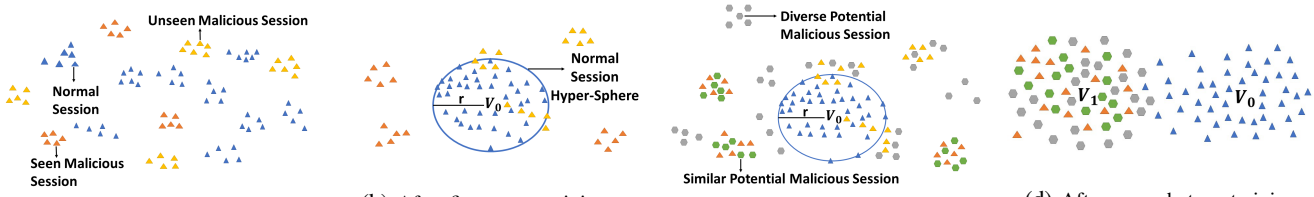
$$\mathcal{L}^{SV} = \frac{1}{R}\sum_{i=1}^{R}(1 - y_i)\left(||\mathbf{z}_i - \mathbf{v}_0||_2\right) \tag{3}$$

The loss function for the first stage is given by:

$$\mathcal{L}_1 = \mathcal{L}^{Sup} + \mathcal{L}^{SV} \tag{4}$$

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 the Section IV-B5, 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

[1]We can relate session representation learning to sentence representation learning wherein, an activity and a session correspond to a word and a sentence, respectively [15]. Any sequence model can be effective for learning sentence representations using contrastive learning [24]. Hence, we have adopted LSTM based session encoder to derive the encoded session representations.



(a) Before training (b) After first stage training (c) Potential malicious sessions (d) After second stage training

Fig. 1: Illustration of encoded representation space for ConRo.

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 the open-set fraud detection task.

B. Second Stage

In the second stage, for each seen malicious session $\mathbf{x}_i \in \mathcal{T}^1$, we achieve the first main objective (MO1) by generating potential/augmented malicious sessions closely resembling $\mathbf{x}_i$. The MO1 can be achieved by leveraging existing augmentation techniques [26]. However, the second main objective (MO2) cannot be achieved straightforwardly due to the lack of required augmentation strategies in the literature [23]. We achieve the MO2 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 the open-set fraud detection task. We do not individually train our encoder by considering normal sessions from $\mathcal{T}$ to avoid over-fitting.

Session augmentation strategy for achieving the MO1. 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. [26] 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. [26] for generating similar potential malicious sessions, which are similar to a seen malicious session $\mathbf{x}_i \in \mathcal{T}^1$.

Let $\widehat{G}^1(\mathbf{x}_i)$ denote this set of generated similar potential malicious sessions. The set $\widehat{G}^1(\mathbf{x}_i)$ is defined as $\widehat{G}^1(\mathbf{x}_i) = \{\widehat{\mathbf{z}} | \widehat{\mathbf{z}} = \lambda_1 \mathbf{z}_i + (1-\lambda_1)\mathbf{z}_j\}$. Here, $\widehat{\mathbf{z}}$ 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 $\mathbf{x}_i$.

Intuitions behind the design of $\widehat{G}^1(\mathbf{x}_i)$. 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 $\widehat{\mathbf{z}} = \lambda_1 \mathbf{z}_i + (1-\lambda_1)\mathbf{z}_j$, aids in achieving this goal. Additionally, performing mix-up with malicious session $\mathbf{x}_j$ will aid in learning separable representations.

Session augmentation strategy for achieving the MO2. We generate *diverse potential malicious sessions* which can diverge significantly from a seen malicious session $\mathbf{x}_i \in \mathcal{T}^1$. Let $\widetilde{G}^1(\mathbf{x}_i)$ denote this set of generated diverse potential malicious sessions. Our session augmentation strategy is inspired by the concept of *affine sets*. The set $\widetilde{G}^1(\mathbf{x}_i)$ is defined as $\widetilde{G}^1(\mathbf{x}_i) = \{\widetilde{\mathbf{z}} | \widetilde{\mathbf{z}} = \lambda_2 \mathbf{z}_i + (1-\lambda_2)\mathbf{z}_j, fp(\widetilde{\mathbf{z}}) = 0\}$. Here, $\widetilde{\mathbf{z}}$ 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:

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

Intuitions behind the design of $\widetilde{G}^1(\mathbf{x}_i)$. 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 seen 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 $\widetilde{G}^1(\mathbf{x}_i)$ 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 stage 2 loss function which is given by:

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

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

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

We show the ConRo training procedure in Algorithm 1.

Algorithm 1 Training procedure for ConRo.

Inputs: $\mathcal{T} = \mathcal{T}^1 \cup \mathcal{T}^0$, R, M, β_1, β_2, and our untrained encoder.
Output: well trained encoder.
generate raw representations for all sessions in $\mathcal{T}$;
[**First Stage**]
for each training batch $S = \{\mathbf{x}_i\}_{i=1}^{R}$ generated from $\mathcal{T}$ **do**
  create the auxiliary batch $S^1 = \{\mathbf{x}_i^1\}_{i=1}^{M}$ from $\mathcal{T}^1$;
  for each normal session $(\mathbf{x}_i, y_i = 0) \in S$ **do**
    construct $A(\mathbf{x}_i) = \left(S \cup S^1\right) - \{\mathbf{x}_i\}$;
    construct $B^0(\mathbf{x}_i) = \{\mathbf{x}_p \in A(\mathbf{x}_i) | y_p = 0\}$;
  calculate $\mathcal{L}^{Sup}$ by using Eq 1 and train the encoder;
  calculate $\mathcal{L}^{SV}$ by using Eq 3 and train the encoder;
[**Second Stage**]
for each training batch $S = \{\mathbf{x}_i\}_{i=1}^{R}$ generated from $\mathcal{T}$ **do**
  create the auxiliary batch $S^1 = \{\mathbf{x}_i^1\}_{i=1}^{M}$ from $\mathcal{T}^1$;
  for each malicious session $(\mathbf{x}_i, y_i = 1) \in S$ **do**
    construct $A(\mathbf{x}_i) = \left(S \cup S^1\right) - \{\mathbf{x}_i\}$;
    construct $B^1(\mathbf{x}_i) = \{\mathbf{x}_p \in A(\mathbf{x}_i) | y_p = 1\}$;
    construct $\widehat{G}^1(\mathbf{x}_i) = \{\widehat{\mathbf{z}} = \lambda_1 \mathbf{z}_i + (1-\lambda_1)\mathbf{z}_j\}$ where $\lambda_1 \sim U(\beta_1, 1.0)$ and $\mathbf{x}_j \sim B^1(\mathbf{x}_i)$;
    construct $\widetilde{G}^1(\mathbf{x}_i) = \{\widetilde{\mathbf{z}} = \lambda_2 \mathbf{z}_i + (1-\lambda_2)\mathbf{z}_j, fp(\widetilde{\mathbf{z}}) = 0\}$ where $\lambda_2 \sim U(-\beta_2, \beta_2)$ and $fp(\widetilde{\mathbf{z}})$ is shown in Eq 5;
    construct $C(\mathbf{x}_i) = A(\mathbf{x}_i) \cup \widehat{G}^1(\mathbf{x}_i) \cup \widetilde{G}^1(\mathbf{x}_i)$;
    construct $D(\mathbf{x}_i) = B^1(\mathbf{x}_i) \cup \widehat{G}^1(\mathbf{x}_i) \cup \widetilde{G}^1(\mathbf{x}_i)$;
    calculate $l\left(\mathbf{z}_i, \mathbf{z}_p, C(\mathbf{x}_i)\right)$ for each session $\mathbf{x}_p \in D(\mathbf{x}_i)$ by using Eq 7;
  calculate $\mathcal{L}_2$ by using Eq 6 and train the encoder;
return well trained encoder;

Time complexity analysis. We analyze the time complexity of our ConRo training procedure by considering the forward

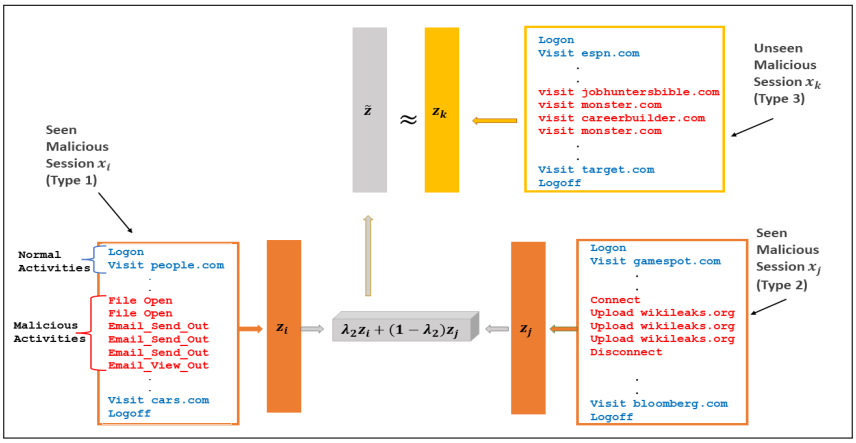


Fig. 2: Illustration of generating a diverse potential malicious session (CERT dataset).

pass and the number of times the individual loss $l(\cdot,\cdot,\cdot)$ is invoked in both stages. This time complexity is given by: $O\left(|\mathcal{T}^0|R + |\mathcal{T}^1|\left(M + |\widehat{G}^1(\mathbf{x}_i)| + |\widetilde{G}^1(\mathbf{x}_i)|\right)\right)$.

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:

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

C. 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 ϵ-$span(\mathbf{x}_j)$ for a session $\mathbf{x}_j$ which is defined as:

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

Here, $y = \{0,1\}$, ϵ-$span(\mathbf{x}_j)$ 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 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 ϵ-$span(\mathbf{x}_j)$ to a set of sessions $\mathcal{S}$ called ϵ-$span(\mathcal{S})$. This set definition is given by:

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

Here, ϵ-$span(\mathcal{S})$ 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 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 property:

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

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

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

We show our proof sketch of Theorem 1 in the Appendix A. 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 $|\widetilde{G}^1(\mathbf{x}_i)|$. Both these steps ensure that generated diverse potential malicious sessions can span diverse regions of the encoded representation space.

IV. EXPERIMENTS

We describe our experimental setup 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.

A. Experimental Setup

1) Datasets: We use three benchmark fraud detection datasets: CERT [9], UMD-Wikipedia [10], and OpenStack [11].

CERT [9]. 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 an open-set learning scenario corresponding to malicious sessions wherein, we include device, email, and file malicious session types in the training set and the remaining two types in the test set. Specifically, we include 30 and 18 malicious sessions in the training and test sets, respectively.

UMD-Wikipedia [10]. 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 et al. [27], which is described below. We calculate the appropriate number of malicious session clusters (K) in the available malicious sessions by using *silhouette coefficient analysis* [28]. 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 [11]. 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.

2) Training Details: By considering an user activity as a word and an activity session as a sentence, we train the word-to-vector model [29] to derive the session raw representation. 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 $\widehat{G}^1(\mathbf{x}_i)$ and $\widetilde{G}^1(\mathbf{x}_i)$ 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 the Section IV-B2. We use the Adam optimizer with a learning rate of 0.005 and we use 10 training epochs for both stages. We utilize three metrics to measure the fraud detection performance: F_1, False Positive Rate (FPR), and Area Under the Receiver Operating Characteristics Curve (AUC-ROC). We report the mean and standard deviation of performance scores after 5 times of running.

3) Baselines: We compare our ConRo framework with five state-of-the-art baselines: DeepSVDD [3], DeepSAD [4], DevNet [5], [6], CLDet [15], and Swan [7]. All these baselines operate in the setting where only a few anomalous samples is available for model training. DeepSVDD, DeepSAD, and DevNet have been designed for the closed-set anomaly detection whereas, Swan has been designed for the open-set anomaly detection. CLDet is a self-supervised contrastive learning based insider threat detection framework and is designed for the closed-set fraud detection task. Except CLDet, the remaining baselines originally operate on image datasets, and employ neural networks for image data such as CNN [4], ResNet-18 [7] 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. [26]. Swan learns in a residual representation space which is defined as $\mathbf{v}_0 - \mathbf{z}$.

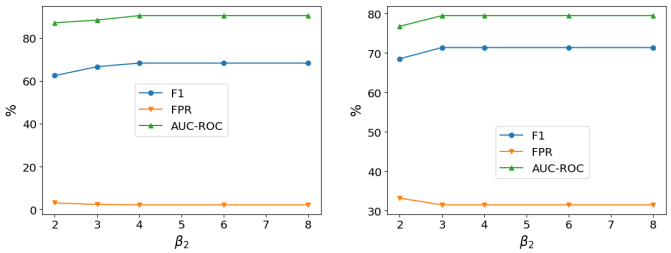


Fig. 3: Sensitivity analysis results w.r.t β_2 on CERT (left column) and UMD-Wikipedia (right column).

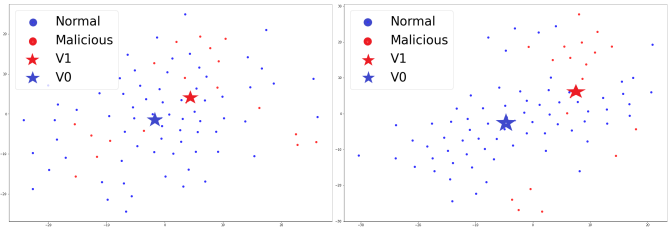


Fig. 4: Visualization of the encoded session representations generated by our encoder for the test set (CERT dataset). The left and right columns denote the encoded session representation spaces after first and second stage training, respectively. Blue and red dots denote normal and malicious sessions, respectively. Blue and red stars denote $\mathbf{v}_0$ and $\mathbf{v}_1$, respectively.

B. Experimental Results

1) Overall Comparison: The performance of our ConRo framework and baselines for all datasets are shown in Table I. Clearly, our ConRo outperforms all baselines[2] 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 challenges related to dataset imbalance and biased malicious training samples by generating a large amount of diverse potential malicious sessions.

For the UMD-Wikipedia dataset, Swan noticeably outperforms our ConRo w.r.t FPR score. Both ConRo and Swan employ different mechanisms for learning about diverse unseen malicious sessions. Specifically, Swan employs residual representation space $(\mathbf{v}_0 - \mathbf{z})$ learning whereas, ConRo learns by generating a large amount of diverse potential malicious sessions. In UMD-Wikipedia dataset, many normal sessions share close similarities with diverse 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. Hence, Swan under-performs against ConRo w.r.t F1 and AUC-ROC scores.

2) Sensitivity Analysis: For analyzing the sensitivity[3] of the hyper-parameter β_2, we first perform the first stage training of our encoder. Then, by employing this first stage trained encoder, we further perform the 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 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 set 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.

3) Visualization Analysis: We employ the tSNE technique [30] for visualization. Specifically, we separately visualize the encoded session representation spaces after first and second stage training. 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[4] are shown in Figure 4. After stage 1 training (refer to the left column), there are many malicious sessions that overlap the normal session cluster. Also, $\mathbf{v}_1$ and $\mathbf{v}_0$ 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 the right column), $\mathbf{v}_1$ and $\mathbf{v}_0$ get well separated. During stage 2 training, we train our encoder by employing both similar and diverse potential malicious sessions due to which, our encoder learns separable representations w.r.t the open-set fraud detection task.

4) Training Latency Analysis: Due to space constraints, we provide a brief summary on the training latency analysis results[4]. All experiments are executed on AMD EPYC (2.3 GHz) CPU server with 26 GB RAM and 226 GB hard disk. In general, ConRo incurs around 6 times more training cost than CLDet, and 10 times more training cost than remaining 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 encoder to learn class-specific shared features, and effectively address the session diversity challenge. Hence,

[2]Since 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 dataset imbalance challenge.

[3]We don't perform sensitivity analysis on β_1 because it is constrained to be set closer to 1 [26].

[4]Detailed visualization and training latency analysis results are available in the preprint version of this article [31].

TABLE I: Performances of our ConRo and baselines (mean±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±4.1	14.30±1.8	62.29±4.8	33.23±1.7	44.90 ±1.4	46.47±0.8	32.27±0.7	42.10±1.4	79.02±0.7
DeepSAD	24.71±7.5	20.53±7.1	84.17±3.6	56.88±2.9	13.30±0.2	68.35±1.7	67.43±3.1	9.70±1.3	94.12±0.1
CLDet	60.41±3.6	3.75±1.9	79.47±2.6	58.78±3.1	9.59±2.8	70.71±2.4	61.71±2.9	6.18±2.1	83.98±1.8
Swan	59.31±2.2	**0.0±0.0**	72.12±0.1	57.02±0.9	**0.0±0.0**	69.89±0.5	62.93±4.2	**0.0±0.0**	73.10±2.3
ConRo	**68.33±3.9**	2.20±0.5	**90.50±0.3**	**71.40±2.3**	31.50±2.1	**79.50±2.1**	**77.56±2.3**	5.80±0.8	**97.10±0.4**

ConRo is able to deliver noticeably better performance than other baselines.

5) Ablation Analysis: 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), $\widehat{G}^1(\cdot)$, and $\widetilde{G}^1(\cdot)$. The ablation analysis results are shown in Table II.

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.

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 $\mathcal{L}^{Sup}$ and $\mathcal{L}^{SV}$, and we do not specifically provide normal sessions having considerable shared features in the encoded representation space as inputs to the DeepSVDD objective.

W/o stage 2 . Mean F1 scores drop to 42.86 (CERT), 60.90 (UMD-Wikipedia), and 46.11 (Open-Stack). In stage 1, 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.

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 fails to achieve separable representations.

W/o $\widehat{\mathbf{G}}^{\mathbf{1}}(\cdot)$. 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 $\widetilde{\mathbf{G}}^{\mathbf{1}}(\cdot)$. 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.

V. 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 is 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*.

Acknowledgement

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

References

[1] S. Yuan and X. Wu, "Deep learning for insider threat detection: Review, challenges and opportunities," *Computers & Security*, 2021.

[2] S. Yuan, P. Zheng, X. Wu, and H. Tong, "Few-shot insider threat detection," in *Proc. of ACM CIKM*, 2020.

[3] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *Proc. of ICML*, 2018.

[4] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K. Müller, and M. Kloft, "Deep semi-supervised anomaly detection," in *Proc. of ICLR*, 2020.

TABLE II: Ablation analysis results (mean±std).

Models	CERT			UMD-Wikipedia			Open-Stack		
	F1	FPR	AUC-ROC	F1	FPR	AUC-ROC	F1	FPR	AUC-ROC
ConRo	**68.33±3.9**	2.20±0.5	**90.50±0.3**	**71.40±2.3**	31.50±2.1	**79.50±2.1**	**77.56±2.3**	5.80±0.8	**97.10±0.4**
w/o stage 1	18.33±1.2	25.10±0.3	78.56±1.1	40.23±1.3	28.37±1.9	55.55±1.1	14.92±1.7	47.14±2.3	49.43±2.9
w/o $\mathcal{L}^{Sup}$	5.28±0.2	48.10±1.8	45.44±0.9	53.16±2.3	25.10±1.9	64.65±1.9	15.12±1.4	46.90±2.7	49.78±2.4
w/o $\mathcal{L}^{SV}$	20.10±1.1	27.05±1.3	83.72±0.7	64.95±0.8	**18.85±6.1**	73.72±0.5	38.76±0.8	31.60±1.1	84.20±0.4
w/o AO	8.99±0.4	68.46±3.1	62.98±1.6	52.04±1.4	80.30±1.4	55.68±1.9	14.08±2.1	26.16±1.4	50.59±2.3
w/o stage 2	42.86±1.1	**0.0±0.0**	63.84±0.1	60.90±1.1	23.85±1.6	70.42±0.6	46.11±3.4	**0.0±0.0**	65.40±1.6
w/o $fp(\cdot)$	31.32±5.1	26.66±6.3	72.77±3.1	59.38±1.8	65.52±4.2	65.95±2.6	37.50±3.7	49.60±3.6	48.16±3.5
w/o $\widehat{\mathbf{G}}^{\mathbf{1}}(\cdot)$	55.92±1.2	4.13±0.3	89.49±0.2	65.58±2.6	31.20±0.3	74.04±2.3	67.57±1.1	9.60±0.4	95.20±0.2
w/o $\widetilde{\mathbf{G}}^{\mathbf{1}}(\cdot)$	44.17±1.9	7.10±0.6	88.16±0.3	63.40±0.8	31.70±4.7	72.10±0.6	52.26±1.6	18.10±1.4	90.61±0.2

[5] G. Pang, C. Ding, C. Shen, and A. van den Hengel, "Explainable deep few-shot anomaly detection with deviation networks," *CoRR*, vol. abs/2108.00462, 2021.

[6] G. Pang, C. Shen, and A. van den Hengel, "Deep anomaly detection with deviation networks," in *Proc. of ACM KDD Conf.*, 2019.

[7] C. Ding, G. Pang, and C. Shen, "Catching both gray and black swans: Open-set supervised anomaly detection," in *Proc. of IEEE CVPR*, 2022.

[8] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," in *Proc. of NeurIPS Conf.*, 2020.

[9] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data," in *Proc. of IEEE SPW*, 2013.

[10] S. Kumar, F. Spezzano, and V. Subrahmanian, "Vews: A wikipedia vandal early warning system," in *Proc. of ACM KDD Conf.*, 2015.

[11] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. of ACM CCS*, 2017.

[12] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surv.*, 2021.

[13] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller, "A unifying review of deep and shallow anomaly detection," *Proc. of IEEE*, 2021.

[14] G. Pang, L. Cao, L. Chen, and H. Liu, "Learning representations of ultrahigh-dimensional data for random distance-based outlier detection," in *Proc. of ACM KDD Conf.*, 2018.

[15] M. S. Vinay, S. Yuan, and X. Wu, "Contrastive learning for insider threat detection," in *Proc. of DASFAA Conf.*, 2022.

[16] S. Zhou, L. Wang, J. Yang, and P. Zhan, "SITD: insider threat detection using siamese architecture on imbalanced data," in *IEEE CSCWD*, 2022.

[17] H. Guo, S. Yuan, and X. Wu, "Logbert: Log anomaly detection via BERT," *CoRR*, vol. abs/2103.04475, 2021.

[18] J. Qi, Z. Luan, S. Huang, Y. Wang, C. Fung, H. Yang, and D. Qian, "Adanomaly: Adaptive anomaly detection for system logs with adversarial learning," in *Proc. of IEEE/IFIP NOMS*, 2022.

[19] M. Salehi, H. Mirzaei, D. Hendrycks, Y. Li, M. H. Rohban, and M. Sabokrou, "A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges," *CoRR*, vol. abs/2110.14051, 2021.

[20] G. Pang, A. van den Hengel, C. Shen, and L. Cao, "Toward deep supervised anomaly detection: Reinforcement learning from partially labeled anomaly data," in *Proc. of ACM KDD Conf.*, 2021.

[21] D. Hendrycks, M. Mazeika, and T. G. Dietterich, "Deep anomaly detection with outlier exposure," in *Proc. of ICLR*, 2019.

[22] G. Pang, C. Shen, H. Jin, and A. van den Hengel, "Deep weakly-supervised anomaly detection," in *Proc. of ACM KDD Conf.*, 2023.

[23] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, "A survey on contrastive self-supervised learning," *CoRR*, vol. abs/2011.00362, 2020.

[24] Z. Wu, S. Wang, J. Gu, M. Khabsa, F. Sun, and H. Ma, "CLEAR: contrastive learning for sentence representation," *CoRR*, vol. abs/2012.15466, 2020.

[25] H. Ju, D. Lee, J. Hwang, J. Namkung, and H. Yu, "Pumad: Pu metric learning for anomaly detection," *Information Sciences*, 2020.

[26] V. Verma, T. Luong, K. Kawaguchi, H. Pham, and Q. V. Le, "Towards domain-agnostic contrastive learning," in *Proc. of ICML*, 2021.

[27] W. Du and X. Wu, "Fair and robust classification under sample selection bias," in *Proc. of ACM CIKM*, 2021.

[28] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, 1987.

[29] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint*, 2013.

[30] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, 2008.

[31] M. S. Vinay, S. Yuan, and X. Wu, "Robust fraud detection via supervised contrastive learning," *arXiv:2308.10055v1 [cs.CR]*, 2023.

APPENDIX

A. Proof Sketch of Theorem 1

There are two cases when the label $y = 1$ and $y = 0$. We will consider both these cases separately. Case when $y = 1$. In this case: $P\left(||\mathbf{v}_1^o - \mathbf{z}||_2 < ||\mathbf{v}_0^o - \mathbf{z}||_2\right) = 1$. 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: $\mathbf{x} \in \bigcup_{\mathbf{x}_i \in \mathcal{T}^1} \epsilon\text{-}span\left(\widehat{G}^1(\mathbf{x}_i) \cup \mathbf{x}_i\right) \cup \epsilon\text{-}span\left(\widetilde{G}^1(\mathbf{x}_i)\right)$. By only considering the first scenario and by the definition of $\epsilon\text{-}span\left(\mathcal{S}\right)$, we have the result:

$$P\left(||\mathbf{v}_1 - \mathbf{z}||_2 < ||\mathbf{v}_0 - \mathbf{z}||_2 \middle| ||\mathbf{v}_1^o - \mathbf{z}||_2 < ||\mathbf{v}_0^o - \mathbf{z}||_2\right)$$

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

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:

$$P\left(||\mathbf{v}_1 - \mathbf{z}||_2 < ||\mathbf{v}_0 - \mathbf{z}||_2 \middle| ||\mathbf{v}_1^o - \mathbf{z}||_2 < ||\mathbf{v}_0^o - \mathbf{z}||_2\right)$$

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

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