DFRWS

Insider Threat Prediction Based on Unsupervised Anomaly Detection Scheme for Proactive Forensic Investigation

Yichen Wei, Kam-Pui Chow, Siu-Ming Yiu*

The University of Hong Kong, Hong Kong, China

Abstract

The complexity, concealment and infrequency of malicious internal actions make it difficult to detect insider threats. In the process of traditional reactive forensic investigation, analysis and interpretation of the digital evidence are performed after a crime has been committed. Even if insiders can be detected, they have already caused huge damage. In this paper, we propose a novel general unsupervised anomaly detection scheme based on cascaded autoencoders (CAEs) and joint optimization network. Our core idea is to utilize CAEs to do data purification among unlabeled digital evidence, then jointly optimize the dimension reduction and density estimation network to avoid sub-optimal problems. Basing on this scheme, we design an end-to-end insider threat prediction framework for proactive forensic investigation, through which we can make real time response to prevent the harmful influences of insider threats in advance. We extract the tractable and scalable feature representation automatically through the data driven Bidirectional Long Short-Term Memory (LSTM) feature extractor, waiving the time-consuming and customarily expert dependable feature engineering work. Additionally, a hypergraph correction module is applied to solve the commonly existed relatively high false positive rate problem in insider threat detection. We evaluate our scheme and framework on public benchmark datasets. The empirical experiments demonstrate that our models outperform state-of-the-art unsupervised methods.

Keywords: Insider threat prediction; Proactive forensic investigation; Unsupervised deep learning; Autoencoder; Hypergraph

1. Introduction

Insider threat detection is a challenging task in the field of digital forensic. As insiders have the privileges to access digital resources inside the organization, it is easy for them to hide their malicious behaviors. Moreover, insider threat actions are rare in the system, no matter the insiders commit the harmful actions unintentionally or with deliberate intentions. It is difficult to catch all insider threat actions with frequent tiny false alarms due to the complication of this problem and the large amount of information needed to be investigated through traditional statistic estimation [1] or rule-based methods. Insider threat prediction is more difficult.

Traditional digital forensic investigation adopts a reactive workflow, i.e., analyzing and interpreting the digital evidence are performed after a crime has been committed. Even if the insiders can be detected, they have already caused huge damage to the organization. Proactive forensic solutions, providing methods for supporting the automation of live investigation [2], have been raised to digitally investigate an incident while it appears and eliminate its harmfulness in advance. One of the popular techniques is deep learning. However, most existing applications that using deep learning for assisting investigation are either supervised or semi-supervised learning models. A large amount of labeled data is required in or before the training process of the supervised or semi-supervised deep learning networks. In reality, it is infeasible in many real-world cases to collect and label so much data.

Deep Autoencoding Gaussian Mixture Model (DAGMM) [3] can be recognized as the state-of-the-art anomaly detection model. The approach needs to use labels to select only the normal data for training, which is actually a semi-supervised approach or so-called one-class classification method. In this paper, we propose an unsupervised anomaly detection scheme which is fully unsupervised in the whole procedure.

Till now, almost all the insider threat detection methods rely on detecting anomalies among the collected information. The majority of them are based on supervised deep learning models. The lack of a large amount of labeled data [4] in the real cases makes these methods impractical. The state-of-the-art unsupervised insider threat detection framework which does not need any labels through the whole process was given in [5]. However, it uses the anomaly detection methods directly as most of existing insider threat methods, which suffers some accuracy losses due to the complexity and concealment of insider threat. Note that not all the anomalies are actual insider threat actions, and some

* Corresponding author. Tel.: +852-28578242; fax: +852-28578242.
E-mail address: smyiu@cs.hku.hk.

normal actions would have abnormal appearances, e.g., some behavior violations due to sudden role change. Insiders would also deliberately hide their malicious behaviors to make them look like normal ones. To overcome this dilemma, we propose a hypergraph correction module to make more accurate detection. After training on previous collected digital evidence, we can make real time predictions of the insider threat behaviors for assisting proactive investigation when malicious incidents appear and new operation data is generated.

Additionally, most unsupervised anomaly detection approaches rely on the steps of data reduction and density estimation, which is easily stuck in less attractive local optima [3] due to decoupled training in the two separate stages. The framework in [5] also encounters the same problem as well. Inspired by [3], we use a joint optimization network to avoid the problem of local optima in our unsupervised anomaly detection scheme. Besides, the representation capability of the input feature matrix plays a vital role in the training process. In the previous insider threat detection deep learning methods, a feature engineering process is performed to extract the features, which is quite time-consuming and customarily expert dependable. In this paper, we use Bidirectional LSTM (BiLSTM) to automatically learn the abstract features [41-45] among the behavior sequences. To our knowledge, this is the first usage of unsupervised BiLSTM feature extractor in insider threat prediction for proactive forensic investigation.

In this paper, our contributions include:

- We propose a novel unsupervised anomaly detection scheme based on cascaded autoencoders (CAEs) and joint optimization training. This is a general unsupervised anomaly detection scheme that can be utilized in various applications in many fields.
- We apply our anomaly detection scheme to solve the insider threat prediction problem. We propose an insider threat prediction framework, an end-to-end learning solution. Technically, we use Bidirectional LSTM to extract the features automatically from our pre-processed behavior sequences and use a hypergraph correction module to make a final prediction. No domain expert knowledge is needed for using this framework.
- Empirical evaluations demonstrate that our unsupervised anomaly detection scheme and insider threat prediction framework achieve superior performance over state-of-the-art models.

2. Related work

Insider threat detection problems are always framed [6] as the problem of anomaly detection, outlier detection and novelty detection [7]. Those work can be divided into three parts: statistical approaches [1], classic machine learning based methods and deep learning models [8]. The statistical approaches and supervised learning are mature research fields, tremendous of anomaly detection methods have been raised [9, 39]. In this paper, we focus on unsupervised anomaly detection. In the real-world case, high dimensional data is always encountered. The core of unsupervised anomaly detection for this kind of high dimensional data is dimension reduction and density estimation. Robust PCA [10] is a de-coupled approach to do the dimension reduction and density estimation separately, which would cause the sub-optimal problem. To overcome this problem, Zong et al. [3] proposed a model containing a compression network and an estimation network, and train them jointly. Additionally, autoencoder (AE) [11] is a widely used model in the research and industry field for anomaly detection and dimension reduction. It is a feed-forward symmetric deep neural network, and the middle layer can be recognized as the non-linear latent representation of the input layer. Although many anomaly detection applications related to AE and its variants (e.g., Sparse AE [35], Denoising AE [36], Contractive AE [37], Variational AE [38]) claim that they are unsupervised solutions, they firstly need the labels to pick up normal data for AE training, which are actually semi-supervised approaches in the whole process. In this paper, we utilize the property of AE in a truly unsupervised manner to do data purification. There are also many deep hybrid models [40] using AE as feature extractors, and the learned hidden representations are the input of latter classifiers. However, those decoupled two-stage methods also lead to sub-optimal performance. Inspired by DAGMM, this paper uses a joint optimization network to solve the sub-optimal problem.

We remark that insider threat detection is not the same as anomaly detection. Not all the anomalous records in the system are insider threats and not all the insider threat records have abnormal appearance. Other prediction or correction measures should be taken to detect insider threats precisely. [12] proposed a scenario-based time-series classification technique to detect insider threats and obtained acceptable performance. Nonetheless, it is a supervised learning method which cannot solve the practical cases without labels. [13] proposed to use a graph-based correlation model to do a second step decision making, its actual implementation is through hand crafted construction of the scenario feature trees and doing a simple matching on the recognized anomalous activities in the previous step, which is not applicable in the real case with large amount of data. In this paper, we devise a hypergraph correction module through hypergraph partition learning [14], which supports unsupervised end-to-end learning and decreases the false positive rate to an acceptable level.

3. Unsupervised anomaly detection scheme based on CAEs and joint optimization

In this section, we propose our novel unsupervised deep learning model for anomaly detection. The model is under the assumption that large amount of training data without any supervision has been collected, in which only a small portion of anomalous data exists. Our model can detect and predict the anomaly in the new coming data[2]. There are two steps in the overall pipeline: 1) unsupervised data purification; 2) joint optimization in dimension reduction and density estimation.

3.1. Overview

Figure 1 shows the whole framework of our proposed unsupervised and joint optimized anomaly detection scheme. This is a CAEs purification based joint optimization scheme (CPJOS). The training dataset is fed into CAEs to unsupervised filter out almost all the anomalous data. Given (K+1) concatenated autoencoders (AEs), after the unsupervised data purification using K AEs, the remaining dataset can be recognized as pure enough dataset with only normal data, containing none or negligible abnormal data. Then the purified normal data is the input of the subsequent joint optimization network, in which the reduction component for dimension reduction is the last AE, and the estimation component for density estimation is connected and optimized jointly with the last AE to overcome the suboptimal performance in the decoupled learning models.

3.2. Unsupervised data purification

The basic AE is a feed-forward symmetric deep neural network containing two parts, the encoder and decoder. In the middle layer of the network is a code layer, which is quite commonly used as the low non-linear latent representation of the input layer. The input and output layer consisting of same number of neurons, and the objective is to reconstruct the input in the output layer. The encoder and decoder have symmetric structure, samples in the input layer will be encoded into the middle code layer and then decode into the output layer to reconstruct the input samples. Given the unlabeled training input and their corresponding output as $X = \{x^{(1)}, x^{(2)}, x^{(3)}, \dots\}$ and $\hat{X} = \{\hat{x}^{(1)}, \hat{x}^{(2)}, \hat{x}^{(3)}, \dots\}$ respectively, where $x^{(i)}, \hat{x}^{(i)} \in \mathcal{R}^n$. The AE tries to learn the function $h_{W,b}(x) \approx x$, where W is the weight and b is the bias [15]. Although the training process of AE does not need any labels, most applications make use of AE to do anomaly detection in a semi-supervised manner. In fact, only normal data is picked to train the AE in the common practical usage, which is depending on the labels in the former step before training. While in many real-world cases, collecting the labeled data is quite difficult and labeling the data is also infeasible, so fully unsupervised methods are in need. Intuitively, based on the structure and property of AE, the model has the ability to learn the patterns of the normal data which accounts for most parts of the whole data, however, it is incapable of learning the patterns of rare abnormal data if it seldom or never observes the patterns during training. Therefore, normal data can be reconstructed well after training, while anomalous data would have a higher reconstruction error. We propose algorithm 1 to purify the original input samples in a real unsupervised manner based on CAEs shown in the upper part of Figure 1.

The original input training samples are feed into the first AE. After training, the samples with larger reconstruction errors are dropped. The remaining samples go continuously into the second AE and so on. After K continuous AEs, almost all the anomalous samples can be filtered out. Only normal samples with none or negligible anomaly data are remaining. The number of K is empirical numbers, depending on the quality of the features extracted and the portion of the anomaly data in the original dataset. We illustrate the filtering performance of our CAEs scheme through empirical evaluation in section 5.1.

3.3. Detection module with joint optimization

After data purification, the remaining purified normal samples are fed into a joint optimization network as shown in the right part of Figure 1, which combines the last AE in CAEs and a fully connected feed forward deep neural network (DNN) and trains them together with a joint objective function. Here we denote an input sample of the last AE as x, the encoding function as $e(\cdot)$, the decoding function as $d(\cdot)$, the latent representation in the code layer as z and the output as $\hat{x}$. Then the encoder and decoder in the last AE can be represented as $z = e(x)$ and $\hat{x} = d(z)$ respectively. This AE plays the role as a dimension reduction component and connects with the subsequent DNN which works as a density estimation component for estimating the parameters in Gaussian Mixture Model (GMM). The representation vector of the reconstruction error between x and $\hat{x}$ are concatenated with z [3] to form the input layer of the DNN, where the reconstruction error representation vector contains the cosine similarity, Pythagorean metric [16], Mahalanobis distance, Manhanttan distance [17], and Chebyshev distance [18] between x and $\hat{x}$.

[2] The detection of a large amount of injected fake data, which is also a kind of anomaly detection task, is out of the scope of our paper.

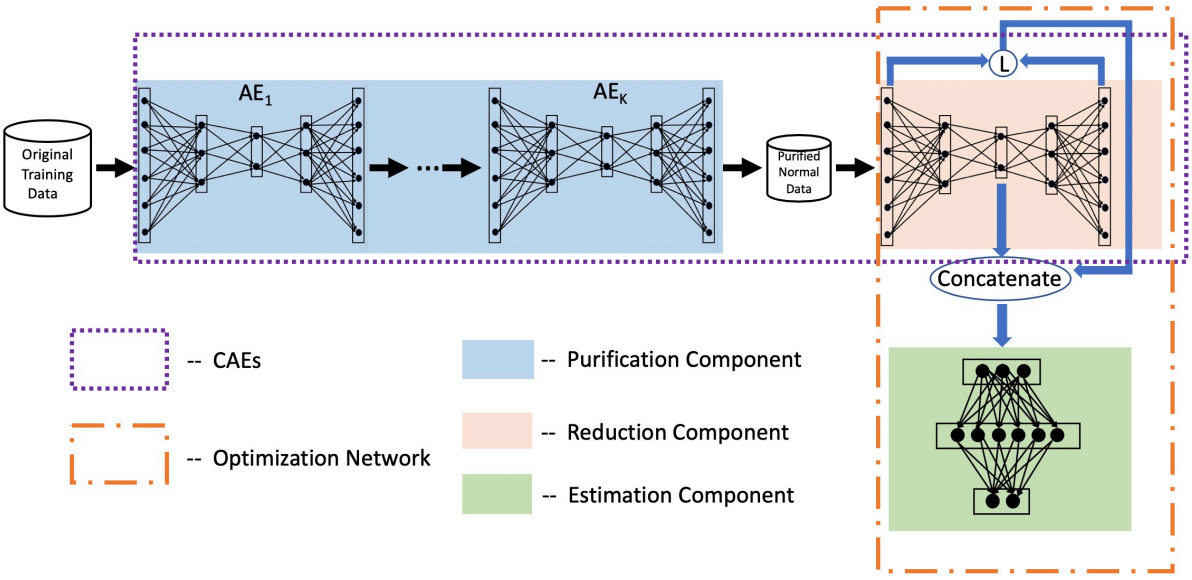

Figure 1. Pipeline of the proposed unsupervised anomaly detection scheme through joint optimization

Algorithm 1: Unsupervised data purification based on CAEs

Input: $\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \dots\}$ (unlabeled data), Dropping rate r

Output: $\boldsymbol{X}_{\mathbf{remain}} = \{\hat{\boldsymbol{x}}^{(1)}, \hat{\boldsymbol{x}}^{(2)}, \hat{\boldsymbol{x}}^{(3)}, \dots\}$ (purified normal data with none or negligible abnormal data)

for k = 1 to K:

  W, b ← Parameter Initialization

  Train the k-th AE with $\boldsymbol{X}$ by loss function $\zeta(\boldsymbol{X}, \hat{\boldsymbol{X}}) = \frac{-\sum_{i=1}^{N}[x^{(i)}\log\hat{x}^{(i)} + (1-x^{(i)})\log(1-\hat{x}^{(i)})]}{N}$, where $\hat{\boldsymbol{X}} = \{\hat{\boldsymbol{x}}^{(1)}, \hat{\boldsymbol{x}}^{(2)}, \hat{\boldsymbol{x}}^{(3)}, \dots\}$ is the output of the AE.

  Calculate the reconstruction error = $|\boldsymbol{X} - \hat{\boldsymbol{X}}|$

  Sort X according to reconstruction error values by ascending

  Drop the items with largest reconstruction errors with dropping rate of r

  X_{remain} is the remaining dataset

  $X \leftarrow X_{\text{remain}}$

We use the output of DNN to estimate the parameters in the GMM distribution with c Gaussian components. The DNN can be expressed as:

$$\hat{p} = \sigma(Wp + b)$$

where p is the input sample, W is the weight matrix, b is the bias vector, σ is the softmax activation function and $\hat{p}$ is the output which is a C-dimensional vector used for mixture membership estimation. Denote the mean and covariance matrix of the c-th Gaussian component as μ_c and Σ_c, and the probability that a sample belongs to the c-th Gaussian component as α_c, we estimate the parameters in GMM as:

$$\mu_c = \frac{\sum_{i=1}^{N'} \hat{p}_{ic}\, p_i}{\sum_{i=1}^{N'} \hat{p}_{ic}}, c = 1,2, \dots \mathrm{C}$$

$$\Sigma_c = \frac{\sum_{i=1}^{N'} \hat{p}_{ic}\,(p_i - \mu_c)(p_i - \mu_c)^T}{\sum_{i=1}^{N'} \hat{p}_{ic}}, c = 1,2, \dots \mathrm{C}$$

$$\alpha_c = \frac{\sum_{i=1}^{N'} \hat{p}_{ic}}{C}, c = 1,2, \dots \mathrm{C}$$

where N' is the number of input samples, $\hat{p}_{ic}$ is the probability that sample i comes from the c-th component. The objective function for joint optimization network containing the loss function of AE and the sample energy [20] of DNN is:

$$J = \frac{1}{N'}\Big(-\sum_{i=1}^{N'}[x_i \log\hat{x}_i + (1-x_i)\log(1-\hat{x}_i)] + \lambda_1 \sum_{i=1}^{N'} E(p_i)\Big) + \lambda_2 \sum_{c=1}^{C}\sum_{j=1}^{d} \frac{1}{\Sigma_{cjj}}$$

where coefficient λ_1 and λ_2 are hyperparameters, and $E(p_i)$ is the sample energy which is negative log probability of observing the input sample p_i:

$$E(p) = -log\left(\sum_{c=1}^{C} \alpha_c \frac{\exp\left(-\frac{1}{2}(p-\mu_c)^T {\Sigma_c}^{-1}(p-\mu_c)\right)}{\sqrt{\det(2\pi\Sigma_c)}}\right)$$

For testing set, we calculate their sample energies as the anomaly scores, all the samples with anomaly scores larger than a threshold ε will be predicted as anomalies. The threshold is set according to cross-validation.

4. Insider threat prediction framework based on CPJOS

In this section, we propose an insider threat prediction framework based on CPJOS as shown in Figure 2. As insider threat behaviors are quite stealthy and complex, it is very difficult to extract exact and precise features to profile the insiders and distinguish their threat actions from normal ones. Given the system log files in the real case, it is time-consuming to do the feature extraction in the field of feature engineering. Here we use natural language processing related methods to extract the features automatically. Then we use the extracted feature matrix to feed into our CPJOS. All the samples with anomaly scores larger than the pre-defined threshold are alarmed as suspicious anomalous behavior sequences. However, not all the anomalous sequences are indeed insider threat actions, and some benign actions would have anomalous appearances. Moreover, during the unsupervised data purification procedure of CPJOS, parts of normal data are also dropped. It would lead to high false positive rate in our framework. Thus, we use a hypergraph correction model to double-check the recognized suspicious sequences, and only the sequences with lower correction probabilities will be predicted as insider threat records.

4.1. Automatic feature extraction without feature engineering

The synthetic CMU insider threat test dataset (r6.2) [31] is used for illustration. Instead of taking much effort to make feature aggregation as our previous framework MAIDF [5], we combine the log lines from each log file on the per user per day basis and arrange them as behavior sequences according to timeline. First, we investigate the users' daily behaviors in the context of this dataset. We extract total 164 behaviors as listed in Table 1. Select one value in each tier to construct one user behavior encoded with a number, e.g., connecting a device in the office hour at weekday is encoded with 1. For each user in each day, there is one behavior sequence after this pre-processing. Then BiLSTM [29][32] network is used to learn the language of user behaviors and extract the features automatically. The structure of the BiLSTM feature extraction network is shown in Figure 3. Given an input sequence $\{b_1, b_2, \ldots, b_t\}$, we use one-hot encoding to embed [46] each behavior b_i to a vector v_i ($1<=i<=t$) and then feed into the BiLSTM as one of the input samples. The sequence lengths vary among all user-day pairs, so all the sequences shorter than the average length are padded with 0 to its end until reaching the average length, and the sequences longer than the average length are truncated. The BiLSTM model can predict the output of v_i as y_i, and the loss function is binary cross entropy between v_{i+1} and y_i.

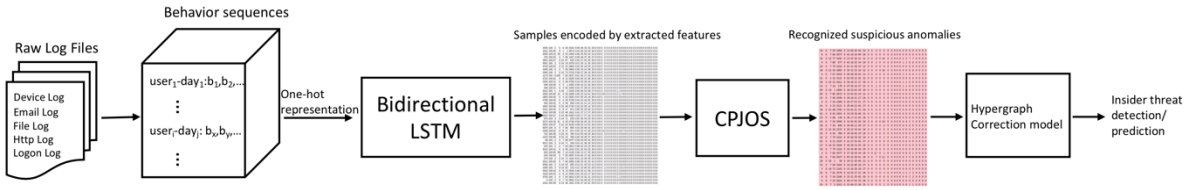

Figure 2. Overall insider threat prediction framework

Table 1. User behavior encoding

Tier 1	Weekdays/ Weekends								
Tier 2	In hour/ After hour								
Tier 3	Device op	Email op		File op		Http op		Logon op	
	Connect/ Disconnect	Send/ View	Competitor/ Mess/Home/ Routine Email	Open/ Copy/ Delete/ Write	From/To removable media/ Locally	Visit/ Upload/ Download	Wikileaks/ Jobsearch/ Keylogger/ Dropbox Websites	Logon/ Logoff	Usual/ Unusual PC
Encoded Number	1 -- 8	9 -- 40		41 -- 88		89 -- 148		149 -- 164	

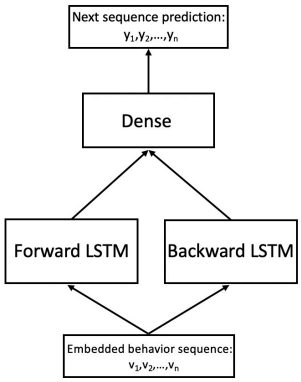

Figure 3. The structure of Bidirectional LSTM

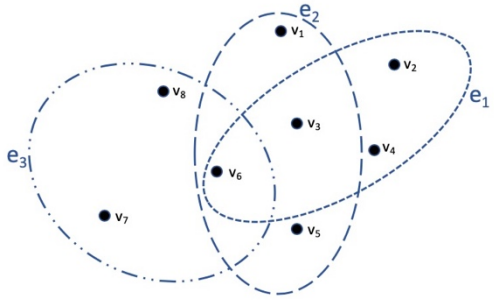

Figure 4. An example of hypergraph

After training, we use the hidden states of the last hidden layer as the learned features. Users' meta-data features, e.g., role, team, supervisor, psychometric test score etc., are concatenated with the learned features to form the final extracted features. Then all the samples, encoded with these extracted features, are fed into our CPJOS. The outputs of CPJOS contain all input behavior sequences' probabilities of them conforming to GMM distribution and the set of suspicious behavior sequences if their corresponding anomaly scores are larger than a threshold determined by cross-validation process. The output probability is considered as the appearance probability (β) of the corresponding samples.

4.2. Hypergraph correction module for insider threat prediction

After training in CPJOS, new log records will be encoded and fed into the trained model when they are generated. Then we can get the encoded suspicious anomalous behavior sequences. However, not all the suspicious anomalies should be predicted as insider threats because not all of them are actual insider threats. According to the demonstration in [5], this kind of model has the ability to detect and predict almost all the actual insider threat records, however, it would encounter relatively high false positive rate. There are mainly two reasons for this kind of false alarms: on one hand, our CPJOS would drop parts of normal data during the CAEs data purification procedure, thus, there is not enough normal data for GMM distribution estimation resulting in some normal behavior sequences been recognized abnormal ones; on the other hand, some behavior sequences would have abnormal appearance due to some unusual incidents. In order to decrease the false positive rate of our detection and prediction, we use a hypergraph [14][19] based correction method to determine the insider threat sequences.

Given a weighted hypergraph $G = (V,E,w)$, where V is a vertex set representing a set of samples, E is a set of hyperedges, and a hyperedge $e \in E$ is a subset of V, $\bigcup_{e \in E} = V$, $w(e)$ is the weight of e. Figure 4 is an example of a simple hypergraph. One of the differences between the hypergraph and the traditional graph is that one hyperedge contains more than two vertices. Denote the size of a hyperedge e as:

$$\delta(e) = |e|,$$

where $|\cdot|$ is the cardinality of a set. For a vertex $v \in V$, e is indent with v if $v \in e$, and the degree of v is $d(v) = \sum_{\{e \in E | v \in e\}} w(e)$. The incidence matrix of a hypergraph G has entries:

$$h(v,e) = \begin{cases} 1, \text{if } v \in e \\ 0, otherwise \end{cases}$$

with $|V| \times |E|$ dimensions, then

$$\delta(e) = \sum_{v \in V} h(v,e).$$

Given a vertex subset $S \subset V$, its complement is denoted as S^c. A hyperedge e is said to be a cut when it is incident with the vertices in the subset S and it complement S^c simultaneously. A hyperedge boundary of S is defined as a hyperedge set which comprise hyperedges that are cuts, which can be denoted as:

$$\partial S = \{e \in E | e \cap S \neq \emptyset, e \cap S^c \neq \emptyset\}.$$

A volume of S is defined as the sum of the degrees of the vertices in S, which can be denoted as:

$$vol\ S = \sum_{v \in S} d(v),$$

And the S's hyperedge boundary volume can be denoted as:

$$vol\ \partial S = \sum_{e \in \partial S} w(e) \frac{|e \cap S|\ |e \cap S^c|}{\delta(e)}.$$

Here we treat the insider threat determination problem as an unsupervised binary classification problem. The suspicious anomalous samples are treated as the vertex set V in the hypergraph, and the feature relationships are treated as hyperedges. It is straightforward to achieve two clusters in which the vertices in the same part have dense connection while the vertices from different parts have sparse connection. Hence, the objective is to get the following partition:

$$\underset{\emptyset \neq S \subset V}{\operatorname{argmin}}\ c(S) = vol\ \partial S \left(\frac{1}{vol\ S} + \frac{1}{vol\ S^c}\right). \quad (1)$$

Since formula 1 raise a NP-complete optimization problem [30], we relax it into a real-valued optimization problem as follows:

$$\underset{f \in R^{|V|}}{\operatorname{argmin}} \frac{1}{2} \sum_{e \in E} \sum_{\{u,v\} \subseteq e} \frac{w(e)}{\delta(e)} \left(\frac{f(u)}{\sqrt{d(u)}} - \frac{f(u)}{\sqrt{d(v)}}\right)^2 \quad (2)$$

s.t. $\sum_{v\in V} f^2(v) = 1, \sum_{v\in V} f(v)\sqrt{d(v)} = 0$.

After this spectral hypergraph partitioning, we can get two partitions, and we call the partition with smaller size as hyper-abnormal partition (HA) and the partition with larger size as hyper-normal partition (HN). The records in HA get the correction confidence with the value of φ multiplying corresponding appearance confidence, i.e., $\varphi \times \beta$; and those ones in HN get the correction confidence with the value of τ multiplying corresponding appearance confidence, i.e., $\tau \times \beta$, where $\varphi = \frac{|HA|}{|HA|+|HN|}, \tau = \frac{|HN|}{|HA|+|HN|}$. Hence, we can give out a belief confidence of each recognized suspicious anomalous sample being the actual insider threat. For users without domain knowledge [33], we set a threshold η for them to make a final decision, where η is set according to cross-validation procedure. They can predict the behavior sequences as insider threats if their corresponding correction confidences are less than η.

5. Experimental Evaluation

In this section, we evaluate the effectiveness of our CAEs, CPJOS and the insider threat prediction framework on some public benchmark datasets comparing with some baseline models.

5.1. Filtering performance of CAEs

Here we show the filtering performance of our CAEs data purification component through empirical evaluation[3]. In order to eliminate the effect of feature engineering quality, we obtain KDD CUP 99 dataset from the UCI repository [21], which already contains well extracted features. There are 4,898,431 samples with 120 features in the original dataset, among which 972,781 records are normal items and the remaining ones are attack records. We pre-process the dataset, make five updated versions with anomaly ratio 5%, 10%, 20%, 30% and 40% to verify the robustness of the CAEs data purification component as shown in Figure 5. Lines with different colors represent the model working on the datasets with different anomaly ratio. They are showing the decreasing tendency of remaining anomalies percentage in the remaining training set autoencoder-by-autoencoder. The horizontal axis represents the *K* value, and the vertical axis represents the percentage of remaining anomalies in the remaining training set. It shows the value of *K* required to filter out all the anomalies in the training set in an unsupervised manner. One AE (*K*=1) is enough to purify the training set when the anomaly ratio is less than 5%, two cascaded AEs (*K*=2) are needed to filter out the anomaly when less than 10% of the data is anomaly and three cascaded AEs (*K*=3) have the ability to drop the anomalous data when the anomaly ratio is less than 20%. Even if the anomaly ratio raises to 40%, our scheme still has the capability to purify the data with unsupervised manner. Although larger value of *K* can ensure dropping more anomaly, it would also lead to dropping more normal data with abnormal appearance, resulting in raising the risk of high false positive rate. The value of *K* should be pre-defined according to the data collected in real case. In the following part, we will show that our unsupervised data purification process can improve the performance of the following dimension reduction and density estimation.

[3] A theoretical demonstration of the filtering performance of AE has been shown in [5].

5.2. Evaluation of CPJOS

In order to demonstrate the effectiveness of our unsupervised anomaly detection scheme, we use some public benchmark datasets with the features already extracted by the domain experts to avoid the effect from the quality of feature engineering. For showing that our CAEs data purification process can improve the performance of the following dimension reduction and density estimation, we select the same benchmark datasets and compared with [3]. Besides KDD CUP 99 dataset, we also choose Thyroid and Arrhythmia dataset from the ODDS repository [22]. Thyroid dataset contains 6-dimensional 3,772 samples with anomaly ratio 0.025. On Arrhythmia dataset, there are 274-dimensional 452 samples with anomaly ratio 0.15. Table 2 shows the number of *K* needed to filter out the anomalies in CAEs data purification component. Table 3 and Table 4 shows the evaluation results of our scheme compared to Kmeans, One-class SVM [23], unsupervised DNN [6], One-class AE [34], DAGMM [3] and MAIDF [5]. We use the standard evaluation metrics for anomaly detection, i.e., Recall, Precision and F1 score. For each column, the best result is listed in bold. CPJOS almost achieve better performance than the state-of-the-art baselines. Although Kmeans and One-class SVM get slightly higher precision, they suffer quite low Recall and F1 as sacrifice, which are unacceptable in real cases.

5.3 Evaluation of the insider threat prediction framework

We use the synthetic CMU insider threat test dataset (r6.2) [24] for performance evaluation of our insider threat prediction framework. There are 120G log files about 4000 users' actions in 516 days, including device connection log file, email log file, file operation log file, network access log file and logon/logoff log file.

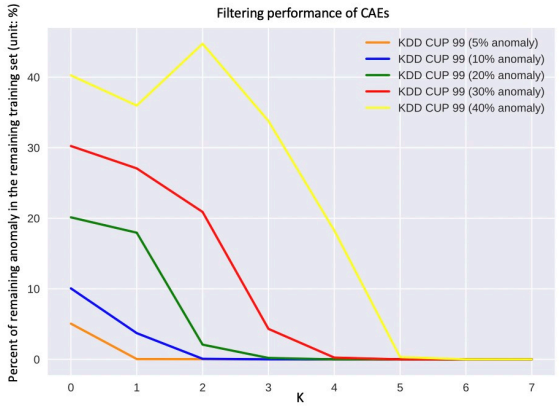

Figure 5. Filtering performance of CAEs

Table 2. The number of K in CAEs purification component for different benchmark dataset

Dataset	K
KDD CUP 99 (anomaly ratio: 0.1)	2
KDD CUP 99 (anomaly ratio: 0.2)	3
KDD CUP 99 (anomaly ratio: 0.3)	4
Thyroid	3
Arrhythmia	2

Table 3. Effectiveness comparison with baseline methods on the KDD CUP 99 dataset

Method	KDD CUP 99 (anomaly ratio: 0.1)			KDD CUP 99 (anomaly ratio: 0.2)			KDD CUP 99 (anomaly ratio: 0.3)		
	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1
Kmeans	0.054	0.102	0.071	0.279	0.549	0.370	0.221	**0.998**	0.362
One-class SVM	0.536	**0.998**	0.698	0.607	0.772	0.680	0.557	0.963	0.706
Unsupervised DNN	0.376	0.224	0.280	0.269	0.329	0.296	0.851	0.609	0.710
One-class AE	0.284	0.287	0.285	0.298	0.159	0.207	0.292	0.292	0.292
DAGMM	0.288	0.145	0.193	0.388	0.383	0.385	0.314	0.313	0.313
MAIDF	0.899	0.713	0.796	0.916	0.937	0.926	0.912	0.887	0.899
CPJOS	**0.918**	0.908	**0.913**	**0.926**	**0.944**	**0.935**	**0.938**	0.926	**0.932**

Table 4. Effectiveness comparison with baseline methods on the Thyroid and Arrhythmia dataset

Method	Thyroid			Arrhythmia		
	Recall	Precision	F1	Recall	Precision	F1
Kmeans	0	0.000	0.000	0.530	0.257	0.346
One-class SVM	0.481	**0.998**	0.649	0.402	**0.943**	0.564
Unsupervised DNN	0.545	0.480	0.511	0.636	0.500	0.560
One-class AE	0.833	0.385	0.526	0.500	0.357	0.417
DAGMM	0.818	0.119	0.208	0.273	0.158	0.200
MAIDF	0.833	0.556	0.667	0.786	0.688	0.734
CPJOS	**0.895**	0.739	**0.810**	**0.867**	0.722	**0.788**

There is another psychometric file indicating employees' psychometric test scores, and a LDAP directory recording employees' role-based information updated in each month. After pre-processing, we get 1, 394,010 behavior sequences including only 78 insider threat sequences. After one-hot embedding, all the sequences are fed into BiLSTM feature learner to automatically learn the appropriate features. After training, all the sequences are encoded with the last hidden layer of the BiLSTM network to get their abstract feature representation. Then all the abstract representations are randomly split into 80% training set, 10% cross-validation set and 10% testing set for the following deep learning models. We implement BiLSTM, CPJOS and other baseline deep learning models basing on Keras [25] running on the Tensorflow [26] backend. Scikit-learn's [27] implementations of Kmeans and one-class SVM are used in our experiments. All our experiments are run on a TITAN X GPU with 12G memory. We follow the hyperparameters setting of BiLSTM model as in [28]. The following illustrates the hyperparameters setting and the network structure of CAEs. We set the dropping rate in the data purification procedures as r=20%, the number of AEs for data purification as K=5. For each AE network, we tune the number of hidden layers (between 5 and 9) and the hidden layer dimension (between 8 and 500) as in [5].

Table 5. Effectiveness comparison with baseline methods on CMU insider threat dataset

Methods	Recall	FPR	AUC
Kmeans	0	0.319	NA
One-class SVM	0.501	0.308	0.338
Unsupervised DNN	0.615	0.200	0.625
One-class AE	0.364	0.200	0.588
DAGMM	0.421	0.202	0.691
MAIDF	0.909	0.200	0.928
Our new framework	**0.933**	**0.054**	**0.957**

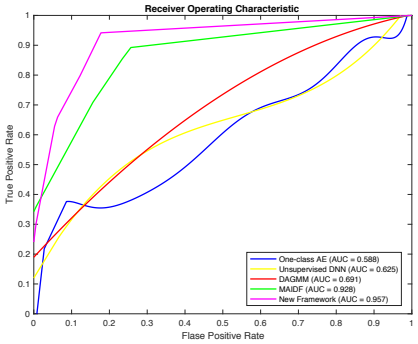

Figure 6. ROC curves of deep learning models

All AE filters are compiled by stochastic gradient descent (SGD) optimizer with learning rate 1*e*-4, training epoch number 50, and batch size 1024. About the hyperparameters in the joint optimization network, we set them referring to [3]. Other deep learning models' parameters are set according to the values suggested in their papers. Table 5 shows the effectiveness comparison between our proposed insider threat prediction framework and other baseline methods on CMU insider threat test dataset (r6.2). Figure 6 list the ROC curves of the deep learning models, smoothed by bionormal method [47,48].

For insider threat prediction, it is critical to discover all the insider threats in the organization, since even one overlooked insider action would bring huge damage. Therefore, Recall (the same as true positive rate, TPR) is the most important metric for framework performance evaluation. Meanwhile, too much false alarms will cause much workload for digital investigators and affect their efficiency, so false positive rate (FPR) is another vital metric to evaluate the methods. As TPR and FPR is a pair of trade-off, we also look highly upon the value of AUC score, which indicates the area under the ROC curve.

The best metric results have been labeled in bold in Table 5. Through empirical experiments, we demonstrate that our insider threat prediction framework outperforms other state-of-the-art baseline models. The proposed new framework achieves FPR over 4 times less than other baseline models. Figure 6 illustrates the ROC curves comparison for deep learning models, in which the abscissa is FPR and the ordinate is TPR. Our insider threat prediction framework acquires higher AUC score than the state-of-the-art deep learning methods.

6. Conclusions

In this paper, we propose a novel unsupervised anomaly detection scheme through joint optimization, which can be applied in many applications. Based on this scheme, we propose an end-to-end insider threat prediction framework for proactive forensic investigation. Our proposed scheme and framework are evaluated on the public benchmark datasets and show superior performance among state-of-the-art baseline models.

For automatic feature extraction, we use BiLSTM as the feature learning model which is capable of catching temporal relations between actions. However, the context of the actions is also important for feature extraction. So the next step, we will try to use BERT learning model as the feature extractor to see whether the performance can be raise or not. Additionally, our design here is for static and sequential data, our future direction would extend to detect anomaly among spatial data. Since AE has shown is strong capability in unsupervised learning and dimension reduction, and there are already many variations for AE, we will try to utilize its advanced variations for automatic feature extraction and dimension reduction for improving the performance and accelerating the training speed in our future work.

References

[1] Markou, M., & Singh, S. (2003). Novelty detection: a review—part 1: statistical approaches. Signal processing, 83(12), 2481-2497.

[2] Alharbi, S., Weber-Jahnke, J., & Traore, I. (2011, August). The proactive and reactive digital forensics investigation process: A systematic literature review. In International Conference on Information Security and Assurance (pp. 87-100). Springer, Berlin, Heidelberg.

[3] Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., & Chen, H. (2018). Deep autoencoding gaussian mixture model for unsupervised anomaly detection.

[4] Ge, W., & Yu, Y. (2017). Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1086-1095).

[5] Wei, Y., Chow, K. P., & Yiu, S. M. (2020). Insider threat detection using multi-autoencoder-filtered unsupervised learning. Sixteenth Annual IFIP WG 11.9 International Conference on Digital Forensics.

[6] Tuor, A., Kaplan, S., Hutchinson, B., Nichols, N., & Robinson, S. (2017, March). Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. In Workshops at the Thirty-First AAAI Conference on Artificial Intelligence.

[7] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM computing surveys (CSUR), 41(3), 1-58.

[8] Markou, M., & Singh, S. (2003). Novelty detection: a review—part 2: neural network based approaches. Signal processing, 83(12), 2499-2521.

[9] Görnitz, N., Kloft, M., Rieck, K., & Brefeld, U. (2013). Toward supervised anomaly detection. Journal of Artificial Intelligence Research, 46, 235-262.
[10] Candès, E. J., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis?. Journal of the ACM (JACM), 58(3), 1-37.
[11] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504-507.
[12] Chattopadhyay, P., Wang, L., & Tan, Y. P. (2018). Scenario-based insider threat detection from cyber activities. IEEE Transactions on Computational Social Systems, 5(3), 660-675.
[13] Jiang, J., Chen, J., Huang, W., & Mohapatra, P. (2019). Warder: Online Insider Threat Detection System Using Multi-Feature Modeling and Graph-Based Correlation. MILCOM.
[14] Zhou, D., Huang, J., & Schölkopf, B. (2007). Learning with hypergraphs: Clustering, classification, and embedding. In Advances in neural information processing systems (pp. 1601-1608).
[15] UFLDL Tutorial. http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/
[16] Anton, Howard (1994), Elementary Linear Algebra (7th ed.), John Wiley & Sons, pp. 170–171, ISBN 978-0-471-58742-2.
[17] Black, Paul E. "Manhattan distance". Dictionary of Algorithms and Data Structures. Retrieved October 6, 2019.
[18] Cyrus. D. Cantrell (2000). Modern Mathematical Methods for Physicists and Engineers. Cambridge University Press. ISBN 0-521-59827-3.
[19] Li, P., & Milenkovic, O. (2017). Inhomogeneous hypergraph clustering with applications. In Advances in Neural Information Processing Systems (pp. 2308-2318).
[20] Zhai, S., Cheng, Y., Lu, W., & Zhang, Z. (2016). Deep structured energy based models for anomaly detection. arXiv preprint arXiv:1605.07717.
[21] Lichman, M. (2013). UCI machine learning repository.
[22] Outlier Detection DataSets. http://odds.cs.stonybrook.edu/
[23] Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. Neural computation, 13(7), 1443-1471.
[24] CMU insider threat test datasets. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099
[25] https://keras.io
[26] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)* (pp. 265-283).
[27] https://scikit-learn.org/stable/
[28] Tuor, A. R., Baerwolf, R., Knowles, N., Hutchinson, B., Nichols, N., & Jasper, R. (2018, June). Recurrent neural network language models for open vocabulary event-level cyber anomaly detection. In Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence.
[29] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
[30] J. Shi and J. Malik. Normalized cuts and image segmentation. IEEE Tran. on Pattern Analysis and Machine Intelligence, 22(8):888–905, 2000.
[31] Glasser, J., & Lindauer, B. (2013, May). Bridging the gap: A pragmatic approach to generating insider threat data. In 2013 IEEE Security and Privacy Workshops (pp. 98-104). IEEE.
[32] Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. IEEE transactions on Signal Processing, 45(11), 2673-2681.
[33] Le, Q., Boydell, O., Mac Namee, B., & Scanlon, M. (2018). Deep learning at the shallow end: Malware classification for non-domain experts. Digital Investigation, 26, S118-S126.
[34] Williams, G., Baxter, R., He, H., Hawkins, S., & Gu, L. (2002, December). A comparative study of RNN for outlier detection in data mining. In 2002 IEEE International Conference on Data Mining, 2002. Proceedings. (pp. 709-712). IEEE.
[35] Makhzani, A., & Frey, B. (2013). K-sparse autoencoders. arXiv preprint arXiv:1312.5663.
[36] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P. A., & Bottou, L. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of machine learning research, 11(12).
[37] Rifai, S., Mesnil, G., Vincent, P., Muller, X., Bengio, Y., Dauphin, Y., & Glorot, X. (2011, September). Higher order contractive auto-encoder. In Joint European conference on machine learning and knowledge discovery in databases (pp. 645-660). Springer, Berlin, Heidelberg.
[38] Kingma, D. P., & Welling, M. (2019). An introduction to variational autoencoders. arXiv preprint arXiv:1906.02691.
[39] Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., ... & Wang, C. (2018). Machine learning and deep learning methods for cybersecurity. IEEE Access, 6, 35365-35381.
[40] Chalapathy, R., & Chawla, S. (2019). Deep learning for anomaly detection: A survey. arXiv preprint arXiv:1901.03407.
[41] Liu, L., De Vel, O., Han, Q. L., Zhang, J., & Xiang, Y. (2018). Detecting and preventing cyber insider threats: A survey. IEEE Communications Surveys & Tutorials, 20(2), 1397-1417.
[42] Diro, A., & Chilamkurti, N. (2018). Leveraging LSTM networks for attack detection in fog-to-things communications. IEEE Communications Magazine, 56(9), 124-130.
[43] Singh, M., Mehtre, B. M., & Sangeetha, S. (2019, January). User Behavior Profiling using Ensemble Approach for Insider Threat Detection. In 2019 IEEE 5th International Conference on Identity, Security, and Behavior Analysis (ISBA) (pp. 1-8). IEEE.
[44] Żołna, K., & Romański, B. (2017, February). User modeling using LSTM networks. In Thirty-First AAAI Conference on Artificial Intelligence.
[45] Zhu, Y., Li, H., Liao, Y., Wang, B., Guan, Z., Liu, H., & Cai, D. (2017, August). What to Do Next: Modeling User Behaviors by Time-LSTM. In IJCAI (Vol. 17, pp. 3602-3608).
[46] Yuan, F., Cao, Y., Shang, Y., Liu, Y., Tan, J., & Fang, B. (2018, June). Insider threat detection with deep neural network. In International Conference on Computational Science (pp. 43-54). Springer, Cham.
[47] Hanley, J. A. (1988). The robustness of the" binormal" assumptions used in fitting ROC curves. Medical decision making, 8(3), 197-203.
[48] Liu, F., Wen, Y., Zhang, D., Jiang, X., Xing, X., & Meng, D. (2019, November). Log2vec: A Heterogeneous Graph Embedding Based Approach for Detecting Cyber Threats within Enterprise. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (pp. 1777-1794).