AB-TCAD: An Access Behavior-Based Two-stage Compromised Account Detection Framework

Kunling He*, Fenghua Li*, Jessie Hui Wang*, Han Zhang*, Yiren Zhao‡
*Institute of Network Sciences and Cyberspace, BNRist, Tsinghua University, Beijing, China
‡University of Toronto, Toronto, Canada
Email: hkl21@mails.tsinghua.edu.cn, lifh@cernet.edu.cn, jessiewang@tsinghua.edu.cn,
zhhan@tsinghua.edu.cn, yiren.zhao@mail.utoronto.ca

***Abstract*—In the fast-growing Internet, legitimate user accounts can be stolen by attackers, posing a serious threat to network security. Therefore, compromised account detection is an urgent problem. Most existing work is based on the behavior and content associated with posts. However, these approaches cannot detect anomalous behavior that attackers only collect information in the early stage, and often fail to consider temporal features. In this paper, we propose an access behavior-based two-stage compromised account detection framework, called AB-TCAD. In the first stage, we propose a novel feature called URL graph that depicts a user's website access pattern. To analyze abnormal changes in user access patterns, we design an AddEdge-GNN algorithm that detects similarities between URL graphs and obtains suspicious accounts. AddEdge-GNN predicts potential new user access behavior and reduces misjudgments that treat normal behavioral changes as anomalies. In the second stage, we propose an RVAE-based temporal detection. We construct temporal features of access behavior in multiple dimensions and use RVAE to detect anomalies, thereby identifying compromised accounts. We perform a real-world evaluation using data from a production network. The results show that AB-TCAD outperforms existing solutions in terms of both precision and recall metrics.**

Index Terms-compromised accounts, access behavior, temporal features, RVAE

I. Introduction

With the increasing number of Internet users, the importance of user account protection is becoming more and more prominent. User accounts contain important data, including personal information and online activities. Protecting account security is not only a matter of personal privacy and property security, but also a matter of healthy development of the entire Internet ecosystem. However, account compromise incidents are occurring on a large scale. Reports reveal that thirty-three million Twitter accounts have been compromised [1], while Yahoo announced a breach affecting one billion user accounts [2]. Recently, a considerable number of LinkedIn accounts have been hacked and held for ransom [3]. Compromised accounts can be used by attackers for malicious activities such as privacy breaches, malware propagation, phishing, and spreading disinformation. These activities not only pose severe risks to individual users but also have the potential to catalyze more extensive security threats. Therefore, detecting compromised accounts is crucial to protecting network and information security.

A compromised account is a legitimate account that has been stolen by attackers through various methods (e.g., weak password cracking, malware, spam, social engineering, etc.) and then used for malicious activities [4]. The detection of compromised accounts can be a difficult task, as the accounts blend the behavioral patterns of both legitimate users and attackers, maintaining a semblance of normalcy [5]. This distinction sets compromised accounts apart from bots and fake accounts, rendering conventional detection methods [6] [7] [8] designed for the latter ineffective for identifying compromised accounts. Existing scenarios for compromised account detection focus on social networks, and the schemes are mainly categorized into two types. The first is behavior-based detection, which usually analyzes the behavioral changes related to user posts, focusing on feature selection and classification algorithms. In terms of feature selection, existing schemes select different features for detection [9] [5] [10], such as the time, source and topic of posts. In terms of classification algorithms, existing schemes propose to use machine learning-based methods for user classification [11], e.g., k-NN and random forest. The behavior-based schemes are fast and efficient, but they need to construct effective features and collect a certain amount of data, and are easily affected by changes in user behavior. The second is content-based detection, which detects anomalies by analyzing the content of user posts. Existing schemes include those based on detecting the user's writing style [12] [13], as well as those based on textual features and semantic features [14] [15]. The content-based schemes can effectively detect anomalies in post content. However, their success depends on the frequency and quality of user posts.

Compromised account detection still faces two challenges. The mixture of attacker and normal user behavior in the compromised account presents a major challenge. Current schemes, whether behavior-based or content-based, often analyze characteristics associated with user posts. However, in the early stage after an account being stolen, attackers tend to only collect large amounts of information and do not post or engage in other malicious activities. For example, Onaolapo et al. [4] find that a number of attackers do not engage in malicious activities during the six months after the stealing, which accounts for 45% of all observed behavior. For this reason, detection based on user posts is inappropriate which fails to detect the above behavior. Industry methods (e.g., Facebook and Google) alert users to logins from unknown IPs or clients but suffer from a high false positive rate. We note that whether attackers collect information or perform malicious activities, the website access pattern of the account changes. As attackers know little about users, it is difficult

ISBN 978-3-903176-63-8 © 2024 IFIP

for them to mimic user access patterns. Therefore, we detect the website access pattern. A scheme close to ours [5] only uses statistical features (such as the top webpage) to describe access behavior, which is insufficient. We present a new feature, URL graph, which uses topological relationships between website visits to represent the access pattern. When an account exhibits abnormal behavior, the URL graph will change. As a result, we can identify suspicious accounts for further detection.

Another challenge is the time dependence of user behavior characteristics. In reality, behavior of both normal and compromised accounts changes over time. Most of the existing schemes only take account of the static behavior characteristics of the user and fail to capture the temporal evolution of attacker behavior. Instead, we need to monitor user activities and detect accounts over a period of time. Thus, to further improve the detection performance, we detect temporal features of access behavior from multiple dimensions to determine whether suspicious accounts are compromised. We design a method based on Recurrent Variational Autoencoder (RVAE), which combines seq2seq [16] and VAE [17] to analyze temporal data efficiently. Specifically, we use the RVAE model to detect the anomalies over time in the three dimensions of access time, access device, and access type.

In this paper, we propose AB-TCAD, an access behavior-based two-stage compromised account detection framework. AB-TCAD consists of two stages. In the first stage, we propose a new feature, URL graph, to capture the pattern of user website access. In order to detect anomalous changes in the access pattern, we propose the AddEdge-GNN algorithm. AddEdge-GNN analyzes similarities between URL graphs based on Graph Neural Network (GNN), which processes graph-structured data, to obtain suspicious accounts. It predicts potential new user access behavior and reduces the misjudgment of treating changes in normal user behavior as anomalies. In the second stage, we design an RVAE-based temporal detection. We identify anomalies in temporal features of access behavior across multiple dimensions. For each feature, we use an RVAE model to detect its anomaly. After reducing the results for all features, we obtain an overall anomaly score of the account to determine whether it is compromised or not.

The main contributions of this paper are as follows:

- We propose a new feature called URL graph that represents a user's website access pattern, based on which we can detect attacker behavior including both information gathering and malicious activities.
- We propose the AddEdge-GNN algorithm to detect anomalous changes in the user access pattern by analyzing the similarity between URL graphs. AddEdge-GNN reduces the misjudgment by predicting newly user-generated access behavior.
- We propose the RVAE-based temporal detection, which combines RVAE to effectively detect multidimensional temporal features of access behavior, further improving the detection performance.

We implement experiments in a real production network. We collect data such as access logs from January 2023 to December 2023 for 1200 accounts, including 130 compromised accounts. We perform tests on workdays and holidays. During workdays, the precision, recall, and F1-score of AB-TCAD are 85%, 66%, and 74%, respectively, which are 10%, 12%, and 12% higher compared to the best-performing existing scheme. During holidays, the precision, recall, and F1-score of AB-TCAD are 81%, 62%, and 70%, respectively, which are 10%, 11%, and 11% higher compared to the best-performing existing scheme.

II. Related Work

Existing approaches for detecting compromised accounts are mainly divided into two types: behavior-based detection and content-based detection.

A. Behavior-based detection scheme

The behavior-based detection scheme usually analyzes the behavioral changes related to user posts to determine whether the account is abnormal or not, focusing on the selection of behavioral features and the choice of classification algorithms. There are many behavior-based schemes.

From the perspective of feature selection, Egele et al. [9] [18] proposed COMPA, which uses a statistical model to build a user's behavioral profile (the time, source and topic of posts, etc.) and to detect statistical anomalies in the user's characteristics. Similarly, Viswanath et al. [19] analyzed liked posts to detect anomalies. VanDam et al. [10] carried out a systematic study on the features of compromised accounts on Twitter and found that the term and source of posts are the most predictive features for detecting compromised accounts. Later, VanDam et al. [20] proposed CADET and utilized the time, source and location of posts for detection. Furthermore, Pv et al. [11] proposed UbCadet, which uses several derived attributes to characterize Twitter users' posts, including similarity of topic tags, time and geographic location of tweets, etc. In addition, there is a scheme [5] that used some statistical features of website visits such as browsing preference and top webpage to detect compromised accounts. From the perspective of classification algorithms, Viswanath et al. [19] used a Principal Component Analysis (PCA) based approach to detect anomalies. CADET proposed by VanDam et al. [20] learns feature embeddings of multiple views and then projects them onto a common latent space for detection. In recent work, UbCadet proposed by Pv et al. [11] uses k-NN and random forest classifiers with different distance metrics for classification.

The behavior-based detection scheme is recognized for its speed, efficiency, and ease of integration, making it the most widely adopted detection method. However, such schemes usually require a certain amount of behavior data about posts, which makes it difficult to detect behavior that attackers only collect information in the early stage.

B. Content-based detection scheme

The content-based detection scheme detects anomalies by analyzing the content of user posts, focusing on writing style [12] [13], textual features [14] [21], semantic features [15], and language modeling [22].

Barbon et al. [12] and Igawa et al. [13] detected compromised accounts based on the user's writing style. When the

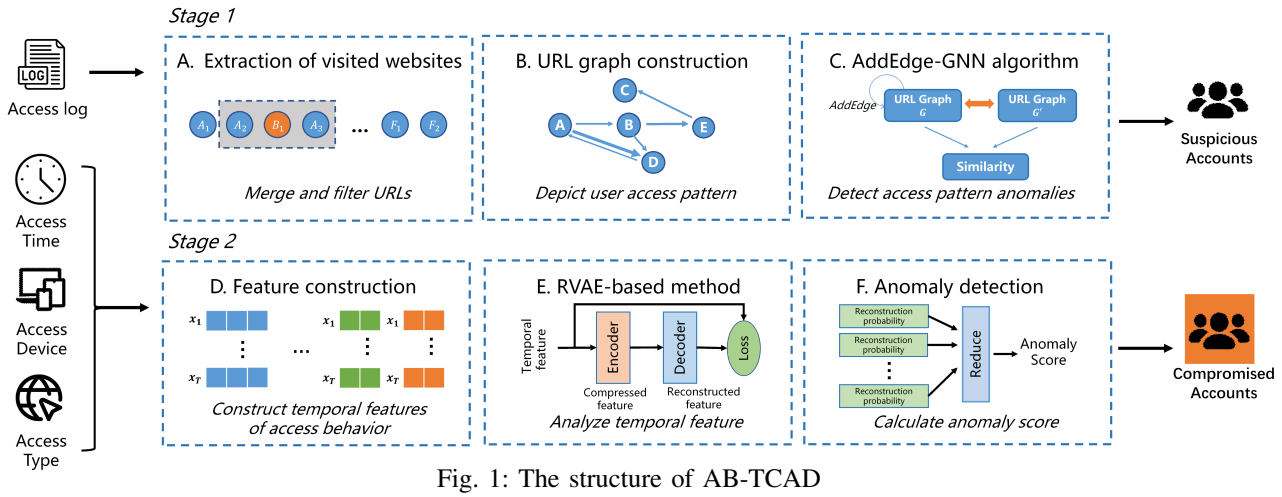


Fig. 1: The structure of AB-TCAD

posted content is inconsistent with the user's writing style, the account is considered compromised. VanDam et al. [14] proposed a detection framework based on the textual feature, CAUTE, which first learns a tweet-to-user encoder (used to infer user features from tweets), and a user-to-tweet encoder (used to predict the content of tweets from the user features and tweet meta-features). The account is then determined to be compromised based on the residuals of the two encoders. Karimi et al. [21] also proposed an end-to-end compromised account detection framework based on the textual feature, E2ECAD, that exploits a contextual representation of user posts to address data sparsity. Seyler et al. [15] defined a new semantic feature to detect compromised accounts by measuring semantic incoherence in the message stream. Seyler et al. [22] proposed a language modeling-based algorithm that identifies compromised accounts by calculating the similarity between the language models of users and attackers.

The content-based detection scheme can effectively detect compromised accounts that post messages, but its effectiveness is related to the frequency and quality of the posting content, and it suffers from privacy and data security issues.

III. Problem Statement

A compromised account refers to a legitimate account that has been accessed or taken over by someone without authorization. Before an account is compromised, the legitimate user typically engages in normal activities, such as browsing news. Once compromised, an unauthorized person may use the account for illegal activities, such as stealing information or posting advertisements. Therefore, the change in account behavior before and after compromise can be used to determine whether the account is compromised or not.

Formally, let $A = \{a_1, ..., a_N\}$ denote a set of user accounts, where N is the number of accounts. Each account a_i is associated with a set of feature vectors $X_i = \{\mathbf{x}_i^1, ..., \mathbf{x}_i^M\}$, where M is the number of features and $\mathbf{x}_i^j$ is the jth feature.

Definition 1. We define compromised account detection as follows. For any account $a_i \in A$ and its corresponding features X_i, we learn a target function $f_1 : a_i \rightarrow S_i$, $S_i \in \mathbb{R}_0^+$, where S_i is a non-negative score representing the probability of account a_i being compromised.

In this paper, we divide compromised account detection into two stages. In the first stage, we initially detect suspicious accounts based on the proposed new feature URL graph G. For each account a_i, we construct its baseline URL graph $G_i \in X_i$ and test URL graph $G_i^{'} \in X_i$. We then propose the AddEdge-GNN algorithm to learn a target function $f_2 : (G_i, G_i^{'}) \rightarrow s_i$, $s_i \in \mathbb{R}_0^+$, where s_i represents the similarity between the baseline URL graph G_i and the test URL graph $G_i^{'}$. If s_i is greater than a threshold, a_i is considered to be a suspicious account. Finally, we get a set of suspicious accounts B. In the second stage, for each suspicious account $b_i \in B$, we construct a set of multidimensional features $\tilde{X}_i \subset X_i$, where $\tilde{X}_i = \{\mathbf{x}_i^1, ..., \mathbf{x}_i^m\}$, with $\mathbf{x}_i^j$ representing the jth temporal feature. Then we use the RVAE-based anomaly detection to learn a target function $f_3 : b_i \rightarrow S_i$, where S_i represents the probability of account b_i being compromised.

IV. The Proposed Approach

In this section, we first present the structure of AB-TCAD. Then, we introduce the details of AB-TCAD in two stages.

A. The Structure of AB-TCAD

AB-TCAD is a two-stage framework, as shown in Fig. 1.

In order to detect attackers' behavior in collecting information in the early stage and malicious activities later, the first stage analyzes the website access pattern to identify the set of suspicious accounts. Specifically, to extract the websites visited by the user, we design a sliding window algorithm to merge and filter URL requests in access logs. Then, the URL graph is proposed and defined. Finally, we introduce the AddEdge-GNN algorithm to analyze anomalous changes in access patterns to identify suspicious accounts.

The second stage utilizes multi-dimensional temporal information on access behavior to further detect compromised accounts from suspicious accounts. Initially, we construct temporal features from dimensions of access time, access device, and access type. Next, we propose a method based on RVAE to analyze each feature. Eventually, by reducing the results of different features, we can obtain the overall anomaly score and determine whether the suspicious account is compromised or not.

B. Stage 1: Suspicious account detection based on access pattern

In the first stage, we first design a sliding window algorithm to extract visited websites from the access logs. Then,

we propose a new feature URL graph to represent the pattern of user website access. Finally, to determine whether there are abnormal changes in the access pattern, we propose the AddEdge-GNN algorithm, which analyzes the similarity of URL graphs to filter out suspicious accounts.

1) Extraction of visited websites: To construct the URL graph, we first detail the process of extracting user-accessed website data from access logs, which are widely available in network systems. Access logs in different networks record the various websites visited by users, such as user home pages and tweet pages in social networks, as well as office and study websites in campus networks, and so on. Specifically, the access logs record the various requests (e.g., common GET and POST, etc.) made by users from clients in time order. Network administrators can use these URL requests in logs to obtain information about the websites visited by users. It is worth noting that, on the one hand, during the process of loading a website, the client requests multiple resources from the backend, such as CSS, JavaScript, and images. This can result in multiple URL requests from a particular website in the logs, which we need to merge. On the other hand, some requests are delayed due to network issues, which are recorded in the logs as arriving after requests from other websites, which we call "disordered URLs". Therefore, we design a sliding window algorithm to merge the URL requests and remove the disordered URLs.

Fig. 2 shows our sliding window algorithm. The access logs record the URL requests from a particular user. We can see that the user visits websites A, C, and F, etc. Multiple URL requests are recorded for each website. Among them, B_1 is a disordered URL. We use a sliding window of length N_{window} with a sliding step of 1. We select the URL with the highest number of URL requests within the window as the result. If the current result is different from the end-of-queue result of the visited website queue, it is added to the queue. Using this sliding window algorithm, we extract the websites visited by the user.

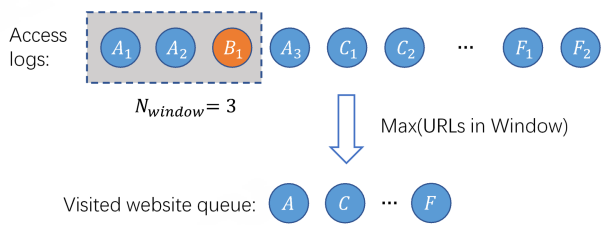


Fig. 2: Extraction of visited websites based on sliding window

2) URL graph construction: We propose the URL graph to represent the pattern of the user's website access behavior, which is defined as follows:

Definition 2. The URL graph for each user is defined as $G(V, E, W)$, where V represents the set of nodes, each node corresponding to a distinct website visited by the user, i.e., $V = \{v_1, v_2, \ldots, v_n\}$. E is the set of directed edges, where each edge $< v_i, v_j >$ represents that the user visits the source website v_i at the current moment and visits the destination website v_j at the next moment, i.e., $E = \{< v_i, v_j > | \ v_i, v_j \in V\}$. W is the weight matrix of the directed edges. $W[v_i][v_j]$ denotes the weight of the directed edge $< v_i, v_j >$, which represents the frequency from the source node v_i to the destination node v_j.

After preprocessing the data for each user, we construct a baseline URL graph G using historical website access data, and a test URL graph G' using recent website access data. The baseline URL graph represents the user's past website access pattern, while the test URL graph represents the user's current website access pattern.

3) AddEdge-GNN algorithm: Then, we analyze the similarity of these two URL graphs for each user to determine whether the user's access pattern changes abnormally and whether the account is suspicious.

It is worth noting that directly analyzing the similarity of these two URL graphs is prone to misjudgment. This is because the normal user's website access behavior changes dynamically, making their test URL graph somewhat different from the baseline URL graph. Therefore, it is easy to judge normal behavioral changes as abnormal, which affects the similarity calculation. For example, a social network user indirectly visits another user through a mutual friend (represented in the URL graph that the user visits website E from website B and jumps to the website of interest C, as shown on the left side of Fig. 3). Over time, the user's access pattern may change, e.g., by directly accessing the other user (represented in the URL graph as a direct access to the target website C from B, as shown on the right side of Fig. 3). Such changes can cause the problem described above.

To address this problem, we propose an algorithm called AddEdge-GNN to compute the similarity. AddEdge-GNN predicts new website access behavior and adds the predictions as directed edges to the baseline URL graph G for model training. This reduces the difference between the baseline URL graph and the test URL graph, thereby reducing the misjudgment mentioned above and improving the detection performance.

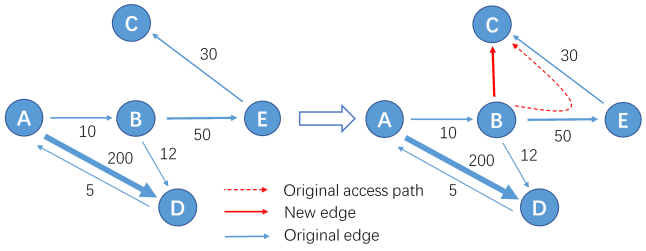


Fig. 3: Variations in the user's URL graph

The AddEdge-GNN algorithm is described in Algorithm 1. First, we predict the user's newly generated website visit behavior. We compute the embedding of each node of the baseline URL graph G, which stores information about the relationship of its corresponding website to other websites in a vector. To compute the node embeddings, we perform some walks that traverse the nodes on the URL graph G to obtain the sequences of nodes. It is worth noting that to avoid the lack of information about other nodes due to trapping into the loop of the URL graph, we design a Reweighted Walk algorithm. For example, in Fig. 3, the user heavily accesses D from A, and D may be a high-frequency website, such as an email service or an online service. It is easy for a walk

to fall into the loop of $A \rightarrow D \rightarrow A$. Therefore, as shown in Algorithm 2, during the walk, we select the most probable node v^{l+1} from the neighbors of the current node v^l and calculate its probability $prob$. If the probability is greater than a threshold Θ, the weight of the edge $< v^l, v^{l+1} >$ is halved, reducing the probability of selecting this edge next time. After obtaining the walking sequences, we take each node as a word and use the context prediction model skip-gram [23] in natural language processing, to obtain the embeddings of all nodes.

Then, we use a binary operation (Average, Weighted-L1, Weighted-L2, etc) on the node embeddings to obtain the weights $edge\_weight$ of all edges. The edge weight represents the distance between the corresponding two websites in the space. The smaller the distance, the higher the homogeneity of the two websites, i.e., the higher the likelihood that the user will visit these two websites directly. If there is an edge with a weight greater than a threshold γ, we consider that the user will visit directly between these two websites in the future, and add the edge to the baseline URL graph G. Next, we use a weight-sharing-based GNN to obtain embedding vectors $graph\_vector$ of the baseline URL graph and the test URL graph after adding edges, which preserve the topological patterns of the two URL graphs. Finally, we compute the similarity score based on the embedding vectors of the URL graphs using the fully connected layer. The similarity represents the degree of difference between the user's previous website access pattern and the current website access pattern. If the difference exceeds a threshold, we consider the account as suspicious.

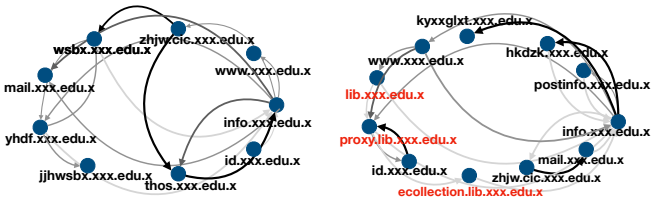
Algorithm 1 The AddEdge-GNN algorithm

Input: Baseline URL graph $G = (V, E, W)$, Test URL graph $G^{'} = (V^{'}, E^{'}, W^{'})$, Walk length L, Context size K, Reweighted threshold Θ, Threshold γ, Embedding dimension D.
for v in V **do**
  $walk$ = ReweightWalk(G, v, L, Θ)
  Add $walk$ to $walks$
end for
$node\_embedding[V]$ = Skip-gram$(walks, K, D)$
$edge\_weight[E]$ = BinaryOperators$(node_embedding)$
for v in V **do**
  for u in V **do**
    if $edge\_weight[< v, u >] > \gamma$ **then**
      Add edge $< v, u >$ to E
    end if
  end for
end for
$graph\_vector$ = GnnWithWeightSharing$(G, G^{'})$
s_i = FC$(graph_vector[G], graph_vector[G^{'}])$

Fig. 4 shows the URL graphs for a suspicious user. The left side is the baseline URL graph and the right side is the test URL graph. The thickness of the edges represents their weight. We can see that the two URL graphs are less similar and the user's website access patterns are quite different. In particular, in the test URL graph, the user visits the academic websites heavily in the current period, which we highlight in red. Therefore, we have reason to suspect that this is a suspicious account.

Algorithm 2 Reweight Walk

Input: Baseline URL graph $G = (V, E, W)$, Starting node v, Walk length L, Reweighted threshold Θ.
for l in L **do**
  $prob, v^{l+1}$ = NodeProbability$(v^{l-1}, v^l, W[v^l][:])$
  if $prob > \Theta$ **then**
    $W[v^l][v^{l+1}] = W[v^l][v^{l+1}]/2$
  end if
  Add v^l to $walk$
end for
return $walk$



(a) (b)

Fig. 4: The abnormal URL graphs

C. Stage 2: RVAE-based Temporal Detection

In the second stage, we propose RVAE-based temporal detection to detect compromised accounts from suspicious accounts. First, we construct temporal features based on multiple dimensions of access behavior. Then, we propose the RVAE-based detection method. Finally, we calculate the anomaly score of the user to determine whether the account is compromised.

1) Temporal features construction: First, we construct temporal features from three dimensions (access time, access device, and access type) to capture user access behavior over time and then quantify them into specific metrics. Other temporal features can be easily incorporated into our framework. The features we construct are as follows:

- Access time: We define this as a 1x24 vector, where each element represents the number of times a user accesses within a given hour of a day.
- First access time: We define this as the hour with the first visit by a user in a day.
- Most access time: We define this as the hour with the most accesses by a user in a day.
- Number of logins: We define this as the number of times a user logs in during a day.
- Number of User-Agents: We define this as the number of User-Agents a user uses in a day.
- User-Agent type: We define this as a 1x8 vector, where each element represents the number of times a user uses a particular user-agent type in a day.
- Visited type: We define this as a 1x8 vector. Websites are categorized into eight types. Each element represents the number of times a user visits that type of website in a day.

We need to preprocess the above features. Each feature $x \in \tilde{X}_i$ of a suspicious account b_i is normalized and then

grouped into time windows of size T. Each group of time series data is a $D \times T$ matrix, denoted as $x = [x_1, ..., x_T]$, and D is the dimension of the feature.

2) RVAE-based method: We propose a detection method based on RVAE. RVAE uses an encoder to compress the input data x, generates the latent vector z, and then uses a decoder to reconstruct x based on z. In other words, the encoder learns the posterior distribution $q_\phi(z|x)$ and the decoder learns the generative probability distribution $p_\theta(x|z)$. We input the temporal feature into RVAE, then reconstruct the feature. Since the data distribution of abnormal and normal behavior is inconsistent, the difference between the input feature and the reconstructed feature is large. We detect whether the user's behavior is abnormal based on this difference. Moreover, both the encoder and decoder of RVAE consist of GRU [24], thus RVAE is effective in detecting anomalies in the temporal feature of compromised accounts. The RVAE-based method is shown in Fig. 5, and we provide a specific introduction below.

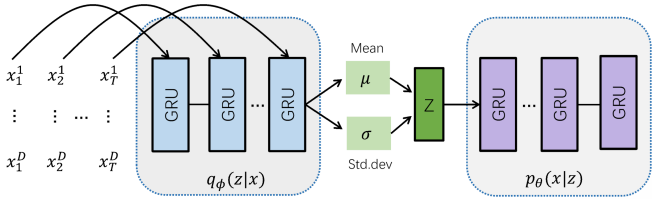


Fig. 5: RVAE-based method

Encoder: The encoder of RVAE takes the user's temporal feature as input and generates a compressed representation. Specifically, the GRU in the encoder takes the temporal feature $x = [x_1, ..., x_T]$ as input and then generates the hidden states $[h_1, ..., h_T]$, to obtain the summarization of the feature at different time steps. Then, the encoder uses h_T to compute the mean μ and variance σ of the latent space and computes the low-dimensional latent vector z, which is the compressed representation of the temporal feature. The specific formulas are defined as follows:

$$\mu = W_\alpha h_T \tag{1}$$

$$\sigma = W_\beta h_T \tag{2}$$

$$z = \mu + \sigma * \epsilon, \ \epsilon \in N(0, 1) \tag{3}$$

where W_α and W_β are the corresponding weight vectors.

Decoder: The decoder of RVAE uses the GRU network similar to the encoder. It uses the compressed representation of the temporal feature as the initial state input to the GRU. After obtaining the hidden state $h_t^{'}$ at time step t, the decoder extracts the information stored in the hidden state and reconstructs the temporal feature of the original input $\hat{x} = [\hat{x}_1, ..., \hat{x}_T]$. The formula is as follows:

$$\hat{x}_t = sigmoid(W_o h_t^{'}) \tag{4}$$

where W_o is the weight vector.

Loss: The loss function of RVAE consists of two parts, where λ serves as a trade-off. The left term is the reconstruction loss, which measures the difference between the input feature data and the reconstructed feature data, and helps RVAE detect anomalies in the input feature. The right term is the Kullback-Leibler (KL) divergence between the approximate posterior and prior of the latent vector z, which is a measure of the difference between probability distributions that helps RVAE learn a more meaningful latent representation of the input feature.

$$loss = -\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] + \lambda * D_{kL}[q_\phi(z|x)|p_\theta(z)] \tag{5}$$

3) Anomaly detection: We compute the reconstruction probability R as the anomaly score as follows:

$$R = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] \tag{6}$$

It is the probability of generating data given latent variables drawn from the approximate posterior distribution [25], representing the difference between the reconstructed feature and the input feature. For the anomalous user, whose reconstructed feature differs more from the input feature, the anomaly score is higher. After obtaining the anomaly scores for all features, we reduce them to obtain the user's overall anomaly score and finally determine whether the account is compromised or not.

V. Implementation And Evaluation

In this section, we introduce the implementation and evaluation of the compromised account detection experiments. First, we describe experimental settings, including the dataset, metrics, and comparison schemes. We then analyze the performance of different detection schemes. In addition, we perform ablation experiments, model comparisons, and analysis of the impact of key parameters. Finally, we summarize the experiment results.

A. Experimental Settings

To evaluate AB-TCAD, we collect data in a real production network, including the website access logs, access time, access device, and other information we need.

The collection period is from January 2023 through December 2023, four months of which are holidays. It is worth noting that the accounts for which we collect data on workdays and holidays remain consistent. The total dataset contains 1200 accounts, of which 130 accounts are identified as compromised accounts. These compromised accounts are stolen by real attackers and not obtained by artificial simulation. Specifically, we find these compromised accounts in a variety of ways, including unknown IP and client logins, abnormal login times, user reports, etc., and finally determine that they are compromised by periodic user callbacks. These compromised accounts are used for phishing, spamming, and spreading advertisements, etc., and about 20% of them have no malicious activity in the early stage. In addition, we split the dataset into 80% training set and 20% testing set.

We use precision, recall, and F1-score to evaluate the performance. Precision is the proportion of samples that are actually positive cases out of all the samples that are predicted to be positive cases. It measures how accurately a scheme determines positive cases. High precision helps us to accurately detect compromised accounts and minimize disruption to normal users. Therefore it is an important

metric in the network operation and maintenance. Recall is the proportion of samples that are correctly predicted as positive cases by a scheme out of all samples that are actually positive cases. High recall helps us to detect all compromised accounts as completely as possible. And F1-score combines precision and recall. Their formulas are as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{7}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{8}$$

$$\text{F1-score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

where TP is the number of true positives, FP is the number of false positives and FN is the number of false negatives.

We compare our approach with four existing representative schemes:

- **COMPA** [9]: COMPA is a behavior-based scheme that builds a profile of the user's behavior about posts and then trains a classifier to detect anomalies.
- **CADET** [20]: CADET is a scheme based on both behavior and content. It proposes a multi-view learning framework that uses a nonlinear autoencoder to learn feature embeddings from multiple views (e.g., content, time, source, and location) and then detects compromised accounts.
- **CAUTE** [14]: CAUTE is a content-based scheme. It learns a tweet-to-user encoder to infer user characteristics and a user-to-tweet encoder to predict tweet content. Then, it detects whether a post is published by a compromised account based on the residual error of the two encoders.
- **E2ECAD** [21]: E2ECAD is also a scheme based on both behavior and content. It is an end-to-end compromised account detection framework that captures the temporal correlation of tweets through LSTM and uses user context for content embedding.

B. Experimental Results

Due to the large difference in user behavior between workdays and holidays, we show the experimental results for these two periods separately. In Fig. 6(a), we show the precision of the different schemes. It can be seen that the existing schemes perform poorly. For workdays and holidays, the precision of COMPA is 59% and 48%, the precision of CADET is 75% and 71%, the precision of CAUTE is 53% and 42%, and the precision of E2ECAD is 65% and 62%, respectively. AB-TCAD achieves the highest precision with 85% and 81%. It can be noticed that CAUTE is the lowest since it detects users based on the relationship between the content of their posted messages and their characteristics, which makes it difficult to detect compromised accounts that do not have malicious posts in the early stage.

In Fig. 6(b), we show the recall of the different schemes. For workdays and holidays, the recall of COMPA is 46% and 38%, the recall of CADET is 54% and 51%, the recall of CAUTE is 48% and 35%, and the recall of E2ECAD is 32% and 23%, respectively. AB-TCAD achieves the highest recall with 66% and 62%. It can be seen that E2ECAD has the lowest recall, which is due to the fact that the features it detects do not fully represent the compromised accounts.

In Fig. 6(c), we show the F1-score of the different schemes. For workdays and holidays, the F1-score of COMPA is 51% and 42%, the F1-score of CADET is 62% and 59%, the F1-score of CAUTE is 50% and 38%, and the F1-score of E2ECAD is 42% and 33%, respectively. It can be seen that AB-TCAD achieves the highest F1-score with 74% and 70% respectively.

In addition, Fig. 7 shows the precision-recall curves of different schemes for both workdays and holidays. It is evident that as recall increases, precision decreases in all schemes. However, it is worth noting that AB-TCAD has the highest detection performance, outperforming other schemes on both workdays and holidays.

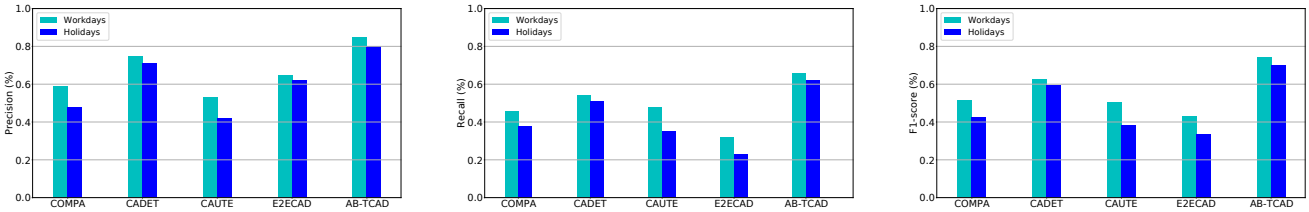
In all three metrics, we can see that the effectiveness during holidays is lower than during workdays. The reason may be that user behavior changes a lot during holidays, such as more random and sparse access time, and an increase in the variety and decrease in the quantity of websites accessed. However, AB-TCAD still achieves the best result.

C. Ablation Experiment

To evaluate the importance of our proposed new feature URL graph and to test the effectiveness of the AddEdge-GNN algorithm, we conduct ablation experiments. Fig. 8(a) and Fig. 8(b) illustrate the results of the experiment. It can be found that when both are used, the precision is 85% and 81% on workdays and holidays, respectively, and the recall is 66% and 62%, respectively. When the URL graph feature is not used, AB-TCAD fails to detect anomalies that compromised accounts collect data in the early stage, with the precision dropping to 62% and 53%, and the recall dropping to 48% and 42%, respectively. When the AddEdge component is not used, AB-TCAD loses its ability to predict the user's new access behavior, with the precision dropping to 75% and 73%, and the recall dropping to 53% and 51%, respectively. It can be demonstrated that the new feature of the URL graph helps us to significantly improve the detection performance, and the AddEdge-GNN algorithm further improves the detection effectiveness.

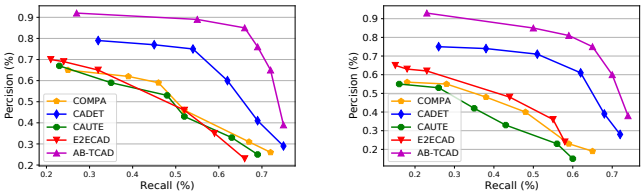
D. Model Comparison

We also analyze the AddEdge-GNN in comparison with other graph similarity models, including graph kernel [26], graph embedding [27], and GNN [28], and the results are shown in Fig. 9(a). The graph kernel-based approach uses a kernel function to predict the similarity between two graphs. The graph embedding-based approach uses the learned node level or graph level representation to predict the similarity. The GNN-based approach uses GNN to learn the graph representation and then computes the similarity. It can be seen that approaches using the GNN model are more effective because they can better capture the characteristics of the user's access behavior. And since AddEdge-GNN can predict new user behavior and reduce the probability of misjudgment (judging normal behavior changes as abnormal), the precision on workdays and holidays is 10% and 8% higher than the existing best method, respectively.



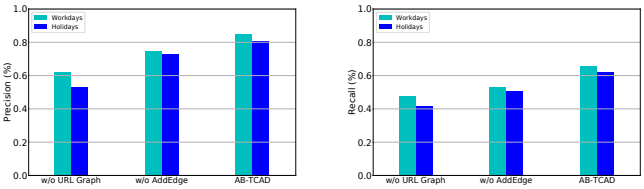
(a) Precision (b) Recall (c) F1-score

Fig. 6: The metrics of different detection schemes across different periods



(a) Workdays (b) Holidays

Fig. 7: The precision-recall curves of different schemes



(a) Precision (b) Recall

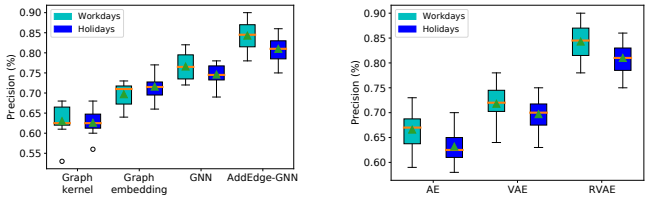
Fig. 8: The URL graph and AddEdge component ablation experiment

To demonstrate the importance of temporal feature detection, we also compare RVAE with other models as shown in Fig. 9(b). We construct non-temporal features in the above-mentioned multidimensions and chose the deterministic model autoencoder (AE) [29] and the probabilistic model VAE to detect anomalies. It can be seen that the precision of RVAE on workdays and holidays is 13% and 11% higher than the existing best method, respectively.

E. Impact of different parameters

We analyze the impact of some key parameters of AB-TCAD on detection precision.

- Sliding window length: Fig. 10(a) illustrates the impact of parameter N_{window} on the precision in the sliding window algorithm. When N_{window} is small, it cannot filter the disordered URLs; when it is large, it filters out some websites the user visits. Therefore, the size of N_{window} is critical. As shown in Fig. 10(a), when N_{window} is set to 6, we achieve the best precision on both workdays and holidays.
- Time range of the baseline URL graph: Fig. 10(b) illustrates the impact on the precision of the time range of the data collected when constructing the baseline URL graph. It can be seen that the highest precision is achieved with a time range of 60 days, both on holidays and workdays. When the time range is too short, capturing the access pattern is difficult, resulting in lower precision. However, when the time range is too long, it is difficult to determine whether changes in user behavior are anomalous (containing some useless historical data), resulting in slightly lower precision.



(a) Graph similarity models (b) Anomaly detection models

Fig. 9: Comparison of the precision of different models

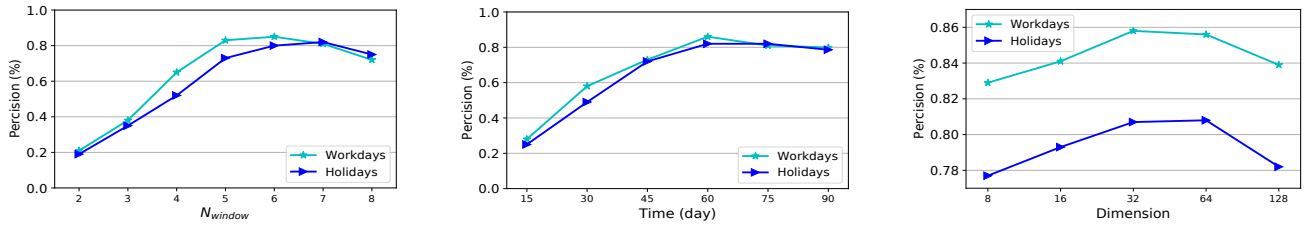
- Dimension of the graph vector: Fig. 10(c) shows the impact of the graph vector dimension (in the AddEdge-GNN algorithm) on the precision. The experiment indicates that the best result is achieved with a dimension of 32 on workdays and 64 on holidays. Overall, the precision increases and then decreases as the dimension increases. This is because a dimension that is too small is insufficient to encode information, while a dimension that is too large leads to overfitting.

F. Summary

Through the above experiments, we can find that the proposed new feature URL graph plays an important role in the detection of compromised accounts. As it can detect anomalies that attackers collect information in the early stage, it helps us to improve the multiple detection metrics. Compared with the existing models, our proposed AddEdge-GNN better detects the user's abnormal access pattern. It reduces misjudgments by predicting the user's new access behavior. In addition, the RVAE-based detection can better detect anomalies in temporal data and further improve the detection performance.

VI. Conclusion

As the number of Internet users continues to rise, the problem of compromised accounts is becoming increasingly severe, affecting network security. However, there are two main problems with existing schemes. One is that they cannot detect compromised accounts that do not have malicious activities in the early stage, and the other is that they do not consider the temporal features. To address these shortcomings, we propose an access behavior-based two-stage compromised account detection framework, called AB-TCAD. In the first stage, we propose a new feature called URL graph to represent the user's access pattern. To detect anomalies in the user access pattern, we propose the AddEdge-GNN algorithm to analyze the similarity between URL graphs and detect suspicious accounts. In the second stage, we propose the RVAE-based temporal detection. We detect anomalies in multidimensional temporal features of access behavior to find compromised accounts. Our final experiments demonstrate



(a) Sliding window length (b) Time range of the baseline URL graph (c) Dimension of the graph vector

Fig. 10: Impact of different values of sliding window length, time range of the baseline URL graph, and dimension of the graph vector on precision

that AB-TCAD achieves outstanding results in terms of both precision and recall metrics.

In future work, we plan to construct profiles based on different user roles to detect anomalies at a finer granularity. In addition, to further improve the detection performance, we plan to add more information to the URL graph, such as the duration of visiting a website. We also plan to explore deep learning models that can better detect anomalies in temporal features. In practical deployment, for the scalability problem when there are too many users, we can perform distributed computation, reduce the model complexity, etc. And for privacy data protection, we can encrypt and desensitize the data, establish strict access control mechanisms, etc.

Acknowledgment

This research is supported by Tsinghua University - Beijing Qihu Technology Co., Ltd Joint Research Center for Cyberspace Surveying and Mapping.

References

[1] J. Furedi, "Details of 33 million twitter accounts hacked and posted online." https://www.independent.co.uk/news/uk/home-news/details-33-million-twitter-accounts-hacked-posted-online-cyber-leaks-a7074416.html, 2016. Accessed: 12 September 2023.

[2] N. P. Vindu Goel, "Yahoo says 1 billion user accounts were hacked." https://www.nytimes.com/2016/12/14/technology/yahoo-hack.html, 2016. Accessed: 12 September 2023.

[3] B. Toulas, "Linkedin accounts hacked in widespread hijacking campaign." https://www.bleepingcomputer.com/news/security/linkedin-accounts-hacked-in-widespread-hijacking-campaign/, 2023. Accessed: 12 September 2023.

[4] J. Onaolapo, N. Leontiadis, D. Magka, and G. Stringhini, "{SocialHEISTing}: Understanding stolen facebook accounts," in *30th USENIX Security Symposium (USENIX Security 21)*, pp. 4115–4132, 2021.

[5] X. Ruan, Z. Wu, H. Wang, and S. Jajodia, "Profiling online social behaviors for compromised account detection," *IEEE transactions on information forensics and security*, vol. 11, no. 1, pp. 176–187, 2015.

[6] C. Liu, L. Sun, X. Ao, J. Feng, Q. He, and H. Yang, "Intention-aware heterogeneous graph attention networks for fraud transactions detection," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 3280–3288, 2021.

[7] B. Hu, Z. Zhang, C. Shi, J. Zhou, X. Li, and Y. Qi, "Cash-out user detection based on attributed heterogeneous information network with a hierarchical attention mechanism," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 946–953, 2019.

[8] L. Zheng, Z. Li, J. Li, Z. Li, and J. Gao, "Addgraph: Anomaly detection in dynamic graph using attention-based temporal gcn.," in *IJCAI*, vol. 3, p. 7, 2019.

[9] M. Egele, G. Stringhini, C. Kruegel, and G. Vigna, "Compa: Detecting compromised accounts on social networks.," in *NDSS*, vol. 13, pp. 83–91, 2013.

[10] C. VanDam, J. Tang, and P.-N. Tan, "Understanding compromised accounts on twitter," in *Proceedings of the International Conference on Web Intelligence*, pp. 737–744, 2017.

[11] S. Pv and S. M. S. Bhanu, "Ubcadet: detection of compromised accounts in twitter based on user behavioural profiling," *Multimedia Tools and Applications*, vol. 79, pp. 19349–19385, 2020.

[12] S. Barbon, R. A. Igawa, and B. Bogaz Zarpelão, "Authorship verification applied to detection of compromised accounts on online social networks: A continuous approach," *Multimedia Tools and Applications*, vol. 76, pp. 3213–3233, 2017.

[13] R. A. Igawa, A. Almeida, B. Zarpelão, and S. Barbon Jr, "Recognition on online social network by user's writing style," *iSys-Brazilian Journal of Information Systems*, vol. 8, no. 3, pp. 64–85, 2015.

[14] C. VanDam, F. Masrour, P.-N. Tan, and T. Wilson, "You have been caute! early detection of compromised accounts on social media," in *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 25–32, 2019.

[15] D. Seyler, L. Li, and C. Zhai, "Semantic text analysis for detection of compromised accounts on social networks," in *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 417–424, IEEE, 2020.

[16] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[17] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[18] M. Egele, G. Stringhini, C. Kruegel, and G. Vigna, "Towards detecting compromised accounts on social networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 4, pp. 447–460, 2015.

[19] B. Viswanath, M. A. Bashir, M. Crovella, S. Guha, K. P. Gummadi, B. Krishnamurthy, and A. Mislove, "Towards detecting anomalous user behavior in online social networks," in *23rd usenix security symposium (usenix security 14)*, pp. 223–238, 2014.

[20] C. VanDam, P.-N. Tan, J. Tang, and H. Karimi, "Cadet: A multi-view learning framework for compromised account detection on twitter," in *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 471–478, IEEE, 2018.

[21] H. Karimi, C. VanDam, L. Ye, and J. Tang, "End-to-end compromised account detection," in *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 314–321, IEEE, 2018.

[22] D. Seyler, L. Li, and C. Zhai, "Identifying compromised accounts on social media using statistical text analysis," *Journal of Environmental Sciences (China) English Ed*, 2018.

[23] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[24] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[25] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special lecture on IE*, vol. 2, no. 1, pp. 1–18, 2015.

[26] S. S. Du, K. Hou, R. R. Salakhutdinov, B. Poczos, R. Wang, and K. Xu, "Graph neural tangent kernel: Fusing graph neural networks with graph kernels," *Advances in neural information processing systems*, vol. 32, 2019.

[27] L. Wang, B. Zong, Q. Ma, W. Cheng, J. Ni, W. Yu, Y. Liu, D. Song, H. Chen, and Y. Fu, "Inductive and unsupervised representation learning on graph structured objects," in *International conference on learning representations*, 2019.

[28] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli, "Graph matching networks for learning the similarity of graph structured objects," in *International conference on machine learning*, pp. 3835–3845, PMLR, 2019.

[29] J. Zhai, S. Zhang, J. Chen, and Q. He, "Autoencoder and its various variants," in *2018 IEEE international conference on systems, man, and cybernetics (SMC)*, pp. 415–419, IEEE, 2018.