Sequence As Genes: An User Behavior Modeling Framework for Fraud Transaction Detection in E-commerce

Ziming Wang
Alibaba Group
Beijing, China
shaoqing.wzm@alibaba-inc.com

Qianru Wu
Alibaba Group
Beijing, China
qianru.wqr@alibaba-inc.com

Baolin Zheng
Alibaba Group
Beijing, China
baolin.zbl@alibaba-inc.com

Junjie Wang
Alibaba Group
Beijing, China
wangjunjie.wjj@alibaba-inc.com

Kaiyu Huang
Alibaba Group
Beijing, China
kaiyu.hky@alibaba-inc.com

Yanjie Shi
Alibaba Group
Beijing, China
clifford.syj@alibaba-inc.com

ABSTRACT

With the explosive growth of e-commerce, detecting fraudulent transactions in real-world scenarios is becoming increasingly important for e-commerce platforms. Recently, several supervised approaches have been proposed to use user behavior sequences, which record the user's track on platforms and contain rich information for fraud transaction detection. Nevertheless, these methods always suffer from the scarcity of labeled data in real-world scenarios. The recent remarkable pre-training methods in Natural Language Processing (NLP) and Computer Vision (CV) domains offered glimmers of light. However, user behavior sequences differ intrinsically from text, images, and videos. In this paper, we propose a novel and general user behavior pre-training framework, named ***S**equence **A**s **GE**nes* (**SAGE**), which provides a new perspective for user behavior modeling. Following the inspiration of treating sequences as genes, we carefully designed the user behavior data organization paradigm and pre-training scheme. Specifically, we propose an efficient data organization paradigm inspired by the nature of DNA expression, which decouples the length of behavior sequences and the corresponding time spans. Also inspired by the natural mechanisms in genetics, we propose two pre-training tasks, namely sequential mutation and sequential recombination, to improve the robustness and consistency of user behavior representations in complicated real-world scenes. Extensive experiments on four differentiated fraud transaction detection real scenarios demonstrate the effectiveness of our proposed framework.

CCS CONCEPTS

• **Applied computing** → **Electronic commerce**; • **Computing methodologies** → **Machine learning**.

KEYWORDS

User Behavior Modeling; Fraud Detection; Representation Learning

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599905

ACM Reference Format:
Ziming Wang, Qianru Wu, Baolin Zheng, Junjie Wang, Kaiyu Huang, and Yanjie Shi. 2023. Sequence As Genes: An User Behavior Modeling Framework for Fraud Transaction Detection in E-commerce. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3580305.3599905

1 INTRODUCTION

With the increasing development of the e-commerce industry and commercial innovation, an appreciable amount of marketing capital is continuously invested in online marketing activities, such as distributing shopping subsidies or discounting products. Due to the potential loopholes in mechanism design and imperfect activity rules, fraudsters can make illegal profits by disguising themselves as ordinary consumers. Hence, e-commerce platforms must establish robust anti-fraud systems to reduce financial losses.

Recently, a bunch of work [3, 16, 22, 23, 37, 40] has been proposed, demonstrating the effectiveness of user behavior sequence data in the anti-fraud system of e-commerce platforms. User behavior sequence data records the user's track on the platform, which contains a wealth of information for fraud detection. In these works, user behavior sequences have been shown to play an essential role in detecting user intentions and fraud tendencies. These works have achieved remarkable progress by exploiting labeled data and supervised learning. In real-world scenarios, however, labeled data are always scarce due to the high annotation cost. Hence, the prior efforts are essential but need to be more satisfactory. We are facing this opportunity, but more facing a huge challenge. The large-scale ever accumulated yet underutilized unlabeled data and emerging pre-training methods [10, 20, 24, 35] provide a promise to solve this problem. Although pre-training models have been extensively researched in NLP tasks, intrinsic differences exist between user behavior and textual sequences [22, 23], leading to a rethinking of the user behavior modeling for fraud transaction detection.

In this paper, inspired by genetics, we propose a novel and general user behavior pre-training framework, ***S**equence **A**s **GE**nes* (**SAGE**), to harness the power of large-scale unlabeled data. Not only has our SAGE framework achieved state-of-the-art performance in fraud transaction detection, but it also provides a new perspective for user behavior modeling. We view transactions as the *expression product* of the user behavior sequence, much like how proteins are the product of gene expression, which we refer to

as the "*Sequence As Genes*". In designing the SAGE framework, we carefully crafted the user behavior data organization paradigm and the representation pre-training scheme, drawing inspiration from the natural mechanisms in genetics.

For the user behavior data organization paradigm, we propose and solve two issues in the data pre-processing step. First, to be applicable to multiple clients and applications, we directly use the user's raw tracks instead of manually curated and simplified tracks for specific clients or applications [21, 22]. Intuitively, this design is more general and can be directly ported to other applications without modification. Second, long-term user behavior sequences have been demonstrated to be helpful for fraud transaction detection [3], but the pre-training for long sequences is still a hard problem [38]. Interestingly, we noticed that a large part of DNA (more than 98% for humans) is *non-coding*, meaning that these sections do not serve as patterns for protein sequences [36]. It inspires us to consider only intercepting crucial segments of the long-term user behavior sequence to expand its corresponding time span while avoiding too-long input sequences. In this intent, we regard the crucial segment of user behavior sequence as its "gene", called **snapshot** in this work. In e-commerce scenarios, we suppose the period neighboured the transactions' creation is most critical, so we truncate the neighboured user behavior segments at the creation of transactions as snapshots. The successful resolution of the long-term problem in turn validates our point of view.

For the representation pre-training scheme, we proposed two pre-training tasks, *i.e.,* **sequential mutation** and **sequential recombination**, which are inspired by the genetic transformations of DNA sequences [17, 25]. Inspired by the natural noising and repair mechanism in genetics, *i.e.,* chromosomal mutation, the sequential mutation task is proposed to learn robust user behavior representation, which randomly selects a mutation type and then encourages the model to reconstruct the original sequence from the noised sequence. In other words, the mutation task can be considered as breaking the sequence of user behaviors and rebuilding the original sequence from it to learn representation. It can be seen that the sequential mutation task borrows from the recent advances in the denoise autoencoding methods. Note that our method both noises and rebuilds the sequences at the snapshot level rather than the token or span level, which is different from previous methods. The sequential recombination task is devised to pull together the representation of homologous behavior sequences by means of user relationships of the same fraud ring, which significantly helps to detect fraudulent users and rings. To this end, we resort to contrastive learning techniques, whose successes depend mainly on the design of data augmentations [4–6, 8, 14]. This is because the design of data augmentations implies the prior hypothesis of domain data invariance knowledge. In fraud transaction detection, we assume that user snapshot sequences of the same fraud ring are homologous. Fig. 1 shows a toy example of snapshots taken from three users. The upper two snapshots come from users involved in a fraudulent ring and exhibit only minor differences, yet they are distinct from the normal user's snapshot (bottom one). This phenomenon is because fraud rings usually operate multiple accounts to carry out similar fraudulent activities. Inspired by the gene recombination that occurs between homologous chromosomes, we proposed a sequential recombination operator between homologous behavior sequences

Figure 1: A toy example of snapshots taken from three users. The upper two snapshots come from users involved in a fraudulent ring and exhibit only minor differences, yet they are distinct from the normal user's snapshot (bottom one).

to obtain augmented user behavior sequences. In practice, we first utilize inter-user relationship information to retain potential fraud rings and construct the augmented sample for performing contrastive learning by recombining the snapshot sequence of different users in the same ring.

We summarize our contribution as follows:

- We introduce a novel perspective, "*Sequence As Genes*", to user behavior sequences and conduct a general and effective user behavior modeling framework based on this perspective. As far as we know, this is the first work in the related field.
- We propose two novel pre-training tasks for fraud transaction detection: the sequential mutation task to learn robustness representation and the sequential recombination task to utilize inter-user relation information.
- We conduct extensive experiments on four real-world business scenarios for fraud transaction detection tasks, and extensive experimental results demonstrate the effectiveness and efficiency of the proposed framework.

The remainder of this paper is organized as follows. Section 2 reviews the related work, and Section 3 details our approach. Section 4 reports the fraud detection experiments on four real-world business scenarios for fraud detection tasks. Section 5 concludes the paper and discusses future work.

2 RELATED WORK

2.1 Fraud Detection

With the rapid growth of e-commerce, fraud detection has received more and more attention from academia and industry due to the huge loss caused by fraudulent activities [2]. Earlier work [29] focused mainly on rule-based methods that rely on human expert knowledge, owing to their simplicity and high interpretability. In addition, some traditional methods [1, 27, 39] extract statistical features from user profiles and historical behaviors and leverage supervised learning algorithms, including tree-based models and neural networks, to detect fraud transactions.

However, due to their lack of ability to deal with changeable and complicated scenarios [29], a line of works [3, 16, 22, 23, 37, 40] based on sequential user behaviors has been proposed. For example, [16] utilized the LSTM network to perform a sequence classification task for fraud detection. [3] viewed payments as an interleaved sequence and presented an entire RNN framework to capture intrinsic

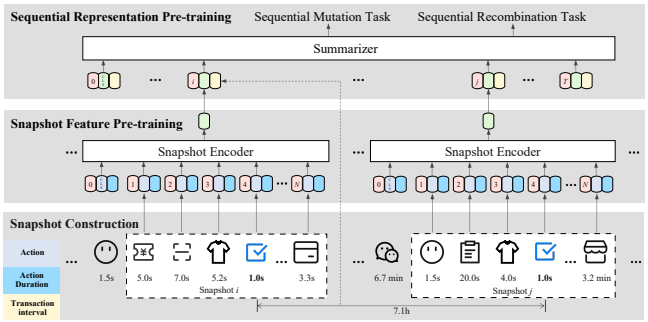

Figure 2: Overview of the *Sequence-As-Genes* (SAGE) framework for user behavior modeling.

patterns. [22] designed a tree-like structure to reorganize the user behavioral data for representing user intentions, combined with tree-based LSTM [32] to achieve better performance. Moreover, [22] proposed a heterogeneous graph attention network to unearth the cross-interaction information over transactions and intentions. However, these works are based on supervised learning techniques, which have limitations in many real-world scenarios, due to the lack of labeled data. There is a very recent work[21], presenting a user behavior pre-training model for fraud detection. However, this work was built on application-specific manual curation and simplified tracks, resulting in a lack of generalizability across different applications. To avoid this problem, we resort to the user's raw tracks on various clients and applications, which are more general and informative.

2.2 Representation Learning On Sequence Data

Here, we briefly review related works about representation learning techniques on sequence data, with a primary focus on denoise autoencoding and contrastive learning.

Denoise autoencoding. A large number of admirable works are variants of denoising autoencoders. To name some, Bert [10] has brought widespread attention to pre-training in NLP tasks with its fantastic performance, whose performance mainly comes from the Masked Language Modeling (MLM) task [24]. BART [20] was proposed to corrupt text with an arbitrary noising function and learn sequence-to-sequence model for reconstructing the original text, which achieved excellent performance in various tasks. TSDAE [35] is similar to BART, but it is devised for sentence embedding. Outside of the NLP domain, VideoMAE [33], inspired by MAE[13], presented video tube masking with an extremely high ratio for video reconstruction, which significantly improved performance for self-supervised video pre-training.

Contrastive learning. Unlike works based on denoising autoencoding, which works by predicting or reconstructing perturbed inputs, contrastive learning is based on the discriminative scheme that makes contrasts between similar and dissimilar samples. A series of work [4–8, 12, 14] has extensively promoted the rapid development of contrastive learning by pulling different views of the same image (positive) closer and pushing apart views from different images (negative) in the embedding space. Interestingly, recent works [7, 12] also have shown that contrastive learning works well without negative samples. By bringing the clusters of points belonging to the same class together and that points belonging to the different classes far apart, SupCon [18] leverage the label information to achieve better performance. Besides, SimCSE [11] applied contrastive learning techniques into the NLP domain by using dropout [30] as noise.

In this paper, we propose two pre-training tasks, *i.e.,* sequential mutation and sequential recombination, which leverage both denoise autoencoding and contrastive learning techniques for modeling user behavior. Current studies seldom explore it.

3 SAGE

3.1 Overview

As shown in Fig. 2, we propose a self-supervised pre-training framework to learn representation from user behavior data for fraud transaction detection. Our framework draws inspiration from the perspective of "*Sequence As Genes*" and can fall into several vital components. First, we propose an input data paradigm to organize long-term user behavioral data. Inspired by the fact that a large portion of DNA is non-coding, we only intercept the crucial segments of a long-term user behavior sequence, *i.e.,* the snapshots, to expand its time span while avoiding too-long input sequences. We employ two cascaded transformers [34] to extract user behavior representation. The former transformer, called the **snapshot encoder**, embeds each single snapshot to a real-value vector at first. Then the latter one, named the **summarizer**, aggregates snapshot features from a specific user to a sequential representation for long-term user behaviors. Our network is pre-trained with a two-stage technique. We first train the snapshot encoder in an MLM-like way. Then we carefully design two novel self-supervised learning tasks inspired by the perspective of "*Sequence As Genes*" to train the summarizer, *i.e.,* the sequential mutation task and the sequential recombination task.

We arrange the remaining part of this section as follows: we first detail the snapshot construction in the input data paradigm and then describe how to perform the snapshot feature pre-training to obtain snapshot representation. Next, we will elaborate on the design of the sequential mutation task and the sequential recombination task.

3.2 Input Data Paradigm

Based on the perspective of "*Sequence As Genes*", we regard user behavior sequences as DNA and transactions as the expression product of user behavior. Similar to the fact that a large part of DNA is non-coding, we noticed that most of the user behavior sequences on the e-commerce platform are verbose and informationless. In contrast, the behaviors neighboring the transactions contain the most crucial information for capturing fraudsters. Hence, we only intercept the user behavior segments close to each transaction as snapshots, and we use the same user's historical snapshots to expand the corresponding time spans of user behavior sequences while avoiding too-long input lengths. Each snapshot consists of three types of tokens as follows:

- **Action token.** The action token denotes a specific action in a certain APP (*e.g.,* item search at Taobao). By observation, fraudsters tend to visit specific pages more frequently. They usually

share some monotonous and brief routes during visiting, while the tracks of normal users are often luxuriant. Hence, we regard a page as a certain user action, and the names of the APPs are further combined within the token since our framework is deployed in a multi-platform environment. We use x^a to represent an action token in this paper. We add a unique special token [ORDER] in place the transaction occurs into the action token sequence. We perform a truncate and padding operation to ensure the length of the action sequence is N.

- **Action duration token.** We introduce an action duration token into this work since we notice that fraudsters statistically stay a shorter time at pages than normal users. This phenomenon comes from the fact that fraudsters have a solid criminal intention during visiting, while normal users tend to gather more information before making a decision. We calculate the duration of each action and discretize it as the action duration token corresponding to each action token. We use x^{ad} to represent the action duration token in this paper.
- **Transaction interval token.** Since our method only intercepts short segments close to transactions, the information describing the relative distance between snapshots is missing in the input data. We introduce the transaction interval token to address this problem. The transaction interval token is obtained by discretization of the time interval between each transaction and its next transaction, and it will be set as a default transaction interval token if the next transaction has not occurred. Unlike the other two types of tokens, each snapshot only has one transaction interval token instead of a token sequence. We use s^w to represent the transaction interval token.

In summary, each snapshot contains an action token sequence, an action duration token sequence, and one specific transaction interval token. For the user behavior sequences with multiple snapshot inputs, we assemble the current snapshot and the corresponding historical snapshots over the past 30 days of the same user together. Each final input sequences is truncated or padded to ensure the number of snapshots is T. Finally, we can formulate the inputs of our model as three vectors, namely the action token sequence $\boldsymbol{x}^a = [x^a_{1,1}, x^a_{1,2}, \ldots, x^a_{T,N}]$, the action duration token sequence $\boldsymbol{x}^{ad} = [x^{ad}_{1,1}, x^{ad}_{1,2}, \ldots, x^{ad}_{T,N}]$ and the transaction interval token sequence $\boldsymbol{s}^w = [s^w_1, s^w_2, \ldots, s^w_T]$.

The snapshot encoder input. Our snapshot encoder receives two types of embeddings as input, *i.e.,* the action embeddings, the action duration embeddings, respectively. We follow the transformer [34] utilizing an embedding matrix $\boldsymbol{W} \in \mathbb{R}^{|\mathcal{V}|\times d}$ to project high dimensional one-hot token representations to low dimensional dense vectors, where $\mathcal{V}$ denotes the token set and d denotes the dimensional of output embedding vectors. Given an action token sequence $\boldsymbol{x}^a_t$ of a specific snapshot, we utilize an action embedding matrix $\boldsymbol{W}^a \in \mathbb{R}^{|\mathcal{A}|\times d}$ to obtain responding action embeddings $\boldsymbol{E}^a = \{\boldsymbol{e}^a_1, \boldsymbol{e}^a_2, \cdots, \boldsymbol{e}^a_N\}$, where $\boldsymbol{e}^a_n \in \mathbb{R}^{1\times d}$ denotes an action embedding. Similar to the action embeddings, we obtain the action duration embeddings $\boldsymbol{E}^{ad}$ using an action duration embedding matrix $\boldsymbol{W}^{ad} \in \mathbb{R}^{|\mathcal{D}|\times d}$. In addition, we follow the Bert [10] to represent the position information by learnable position embeddings $\boldsymbol{E}^{ap}$. Finally, we formulate the inputs of the snapshot encoder by simply adding the three types of embedding together as: $\boldsymbol{E} = \boldsymbol{E}^a + \boldsymbol{E}^{ad} + \boldsymbol{E}^{ap}$.

The summarizer input. As shown in Fig. 2, our summarizer also receives two types of vectors as input, *i.e.,* the snapshot-level feature vectors and the transaction interval embeddings, respectively. For the snapshot-level feature vectors, we use a pre-trained frozen snapshot encoder to encode the $\boldsymbol{E}^a$, the $\boldsymbol{E}^{ad}$ and the $\boldsymbol{E}^{ap}$ of a snapshot into a snapshot-level feature vector $\boldsymbol{s}^v \in \mathbb{R}^{1\times d}$. In this way, for the user behavior sequences with multiple snapshot inputs, we can obtain the snapshot-level feature vectors: $\boldsymbol{S}^v = \{\boldsymbol{s}^v_1, \boldsymbol{s}^v_2, \cdots, \boldsymbol{s}^v_T\}$. Then, we use a transaction interval embedding matrix $\boldsymbol{W}^{ti}$ to obtain the transaction interval embeddings $\boldsymbol{s}^w \in \mathbb{R}^{1\times d}$. We also use learnable snapshot position embeddings $\boldsymbol{S}^{sp}$ for representing the snapshot position information. Finally, we formulate the inputs of the summarizer by adding the three types of vectors together as: $\boldsymbol{S} = \boldsymbol{S}^v + \boldsymbol{S}^w + \boldsymbol{S}^{sp}$.

3.3 Snapshot Feature Pre-training

The pre-training of our snapshot encoder module is based on a self-supervised learning task similar to the MLM. We basically follow the design of BERT [10] in this part. Still, due to the input of our snapshot encoder containing two types of embeddings, there are some differences in our embedding layer and loss function. For simplicity, we only present how we differ from the MLM task.

Mask action modeling task. Our Mask Action Modeling task (MAM) first randomly masks some percentage of the inputs and then encourages the model to predict those masked action tokens and action duration tokens. In practice, we first select the indices required to be masked and then mask the action tokens and action duration tokens simultaneously. For the predictors, we use two predictors to predict the corresponding action tokens and action duration tokens from the last hidden state, respectively. The Mask Action Modeling Loss can be defined as:

$$\begin{aligned}
\mathcal{L}^a_{MAM} &= -\sum_{i=1}^{Q} \log P_{\theta_{se},\theta_a}\left(x^a | \tilde{x^a}\right) \\
\mathcal{L}^{ad}_{MAM} &= -\sum_{i=1}^{Q} \log P_{\theta_{se},\theta_{ad}}\left(x^{ad} | \tilde{x^{ad}}\right) \\
\mathcal{L}_{MAM} &= \mathcal{L}^a_{MAM} + \mathcal{L}^{ad}_{MAM}
\end{aligned} \tag{1}$$

where θ_{se} is the parameters of the snapshot encoder, and θ_a, θ_{ad} correspond to the parameters of the action predictor and the action duration predictor, respectively. Q is the size of the masked indices set.

Snapshot representation. After the pre-training, we use the first-last-avg method [31] to obtain the snapshot level representation, which is represented as $\boldsymbol{s}^v$ in this work. Notably, the atomic input units of the following pre-training tasks in this section are all the snapshot features extracted from the frozen pre-trained snapshot encoder module.

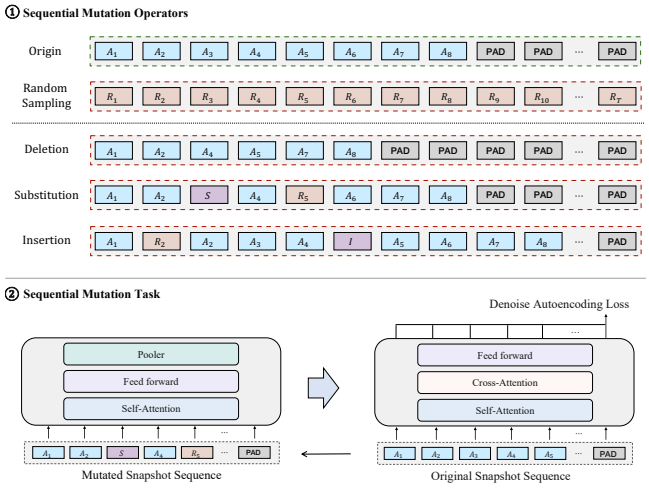

Figure 3: An illustration of sequential mutation operations and tasks.

3.4 Sequential Mutation Task

To learn sequential representation from a sequence composed of multiple snapshot features, a naive idea is using a mask-then-predict self-supervised pre-training task. However, inspired by the natural noising and repair mechanism in genetics, we borrow from the mutation transformations in chromosomal mutation to add noise to the snapshot sequence and encourage the model to reconstruct the original sequence. This task also follows the intuition that more noising methods in the denoise autoencoding task can help learn more robust representation. Next, we first describe the mutation operators used in this work and then introduce the loss function used in the sequential mutation task.

Mutation operators. We use the three types of mutation operators to add toxicity to the original snapshot sequence. Fig. 3 shows the toy examples of mutated snapshot sequences. These operators can add various toxicities, *e.g.,* losing information, shifting indices, or inconsistencies in behavior patterns. In practice, we randomly select a mutation operator as follows for a specific snapshot sequence and then perform the selected mutation on the snapshot sequence:

- **Deletion**: This operator randomly selects the toxic indexes by α_D probability and deletes the selected snapshots from the sequence and rearranges the remaining snapshots.
- **Substitution**: This operator randomly selects the toxic indexes by α_S probability and replaces the selected snapshots with a special placeholder snapshot `[S]` or a snapshot randomly sampling from other users.
- **Insertion**: This operator randomly selects the toxic indexes by α_I probability and inserts a special placeholder snapshot `[I]` or a snapshot randomly sampling from other users at each selected index.

After mutation, we perform a truncate and padding operation to ensure the length of the mutated snapshot sequence is T.

Sequential mutation loss. Our sequential mutation task uses an objective function similar to TSDAE [35] to learn user behavior representations by reconstructing corrupted sequences. For both original and mutated sequences, we add a special `[CLS]` snapshot to the head and a special `[EOS]` snapshot to the tail of each sequence. Then, we take the mutated snapshot-level features and corresponding transaction interval embedding as input and feed them into our summarizer module to obtain the sequential embedding of the mutated sequence. Then we feed this sequential embedding and the original sequence into a modified cross-attention decoder, which has the following cross-attention mode:

$$\begin{aligned} H^{(k)} &= \text{Attention}\left(H^{(k-1)}, [s], [s]\right) \\ \text{Attention}(Q,K,V) &= \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V \end{aligned} \tag{2}$$

where $H^{(k)} \in \mathbb{R}^{t\times d}$ is the decoder hidden states within t decoding steps at the k-th layer, d is the size of the sequential embedding, $[s] \in \mathbb{R}^{1\times d}$ is a one-row matrix including the sequential embedding vector and Q, K and V are the query, key and value, respectively.

Finally, the form of our denoise autoencoding loss is to predict the original snapshot-level features from the last hidden state of the decoder. Different from the cross-entropy loss used in TSDAE [35]. For the prediction of snapshot features, we first use a predictor head to obtain the prediction embeddings, and then use an InfoNCE [26] loss to maximize the similarity between each prediction embedding and the original snapshot feature corresponding to its location, and minimize the similarity between the prediction embeddings and other snapshot features in the batch, which can be defined as:

$$\begin{aligned} \mathcal{L}_M^v &= -\sum_{i=1}^{M} \log P_{\theta_s,\theta_d}\left(s^v|\tilde{s^v}\right) \\ &= -\sum_{i=1}^{M} \log \frac{\exp\left(v_i \cdot s_i/\tau_m\right)}{\sum_{j=1}^{J}\exp\left(v_i \cdot s_j/\tau_m\right)} \\ \mathcal{L}_M^w &= -\sum_{i=1}^{M} \log P_{\theta_s,\theta_{wp}}\left(s^w|\tilde{s^w}\right) \\ \mathcal{L}_M &= \lambda_1 * \mathcal{L}_M^v + \lambda_2 * \mathcal{L}_M^w \end{aligned} \tag{3}$$

where τ_m is a manually set temperature parameter, θ_s is the parameters of the summarizer module, and θ_d corresponds to the parameters of the decoder module, and θ_{wp} corresponds to the parameters of the transaction interval predictor module, and v_i is the output of the decoder predictor which corresponding the original snapshot-level feature input s_i, and M is the size of the toxic indices set, and J is the total number of the snapshot features in the current training mini-batch. λ_1, λ_2 are trade-off hyperparameters set manually.

3.5 Sequential Recombination Task

In fraud detection scenarios, there are two typical fraud behavior patterns: a single fraudster manipulating amounts of accounts or a group of fraudsters committing crimes synchronously similarly. Any of the two cases result in a group criminal tendency, in which the behaviors of members within the same group are usually consistent. Hence, we employ contrastive learning techniques to help to learn representation for detecting fraudulent users by pulling together the representation of user behavior sequences in the same

fraud ring. However, the success of contrastive learning methods often depends on the design of data augmentations. Interestingly, we found a natural "group relationship" (homologous chromosomes) and "data augmentation" (gene recombination) mechanism in genetics: first, the relationship between homologous chromosomes is very similar to the relationship between users in the same fraud ring; second, gene recombination can generate new genotypes in offspring by exchanging parts of the DNA sequence between the homologous chromosomes. This inspired us to obtain the potential user behavior sequence transformations by recombining the user behavior sequences of two users in the same user group. We carefully designed recombination operators to avoid the label leakage problem in contrastive learning or samples that are impossible to appear in the real world. Next, we briefly introduce our pseudo user group generating method, then explain our sequential recombination operator.

Pseudo user group generating. First, we construct a heterogeneous graph in which vertexes are users and edges are linked among users that share the same device, IP addresses, mobile numbers, receive addresses, or other connections within the lastest 180 days. Then we perform the Label Propagation Algorithm (LPA) [28] to obtain the coarse pseudo user group labels. Finally, we refine the pseudo user group labels according to whether the user group has used a certain amount of similar marketing funds. We use this refine mechanism to filter out fraudulent rings that are not targeting marketing funds.

Recombination operator. As shown in Fig. 4, given two snapshot sequences A and B, corresponding two users in a same pseudo user group, we attempt to augment the sequence by recombining those snapshot sequences, that is, exchanging some sub-segments from the other one. We carefully designed the recombination operator to simulate reliable user behavior transformations in the real world. Firstly, the adjacent snapshots of the same user may have similar fragments if the time interval is too close. Assigning such two snapshots to a recombined pair may cause label leakage in contrastive learning. To avoid this problem, we first merge the snapshots of each user that are close in time to form atomic recombine segments, and these atomic segments are distinguished by different shading in the "Split" step in Fig. 4. Next, when selecting segments for exchanging, we randomly select a span of contiguous atomic segments rather than a single one. In our practice, the average length of these spans is 2, and the red and green boxes in the "Select" step in Fig. 4 show these selected atomic segment spans. Furthermore, we also ensure the segments with temporal conflicts are separated into different augmented sequences. Finally, we exchange the select snapshots and rearrange the recombined sequences in reverse chronological order to obtain the augmented sequences, which illustrate in the "Recombine" step and "Reorder" step in Fig. 4.

Sequential recombination loss. We use the recombination operator to obtain two partially exchanged recombination snapshot sequences and feed them into our summarizer module, respectively. Then we use the last hidden state of the special [CLS] snapshot as the sequential embedding of each sequence, which can be described as h_i. Then, following the popular setting in contrastive learning [4], we employ a projection head to project the h_i to a hidden space

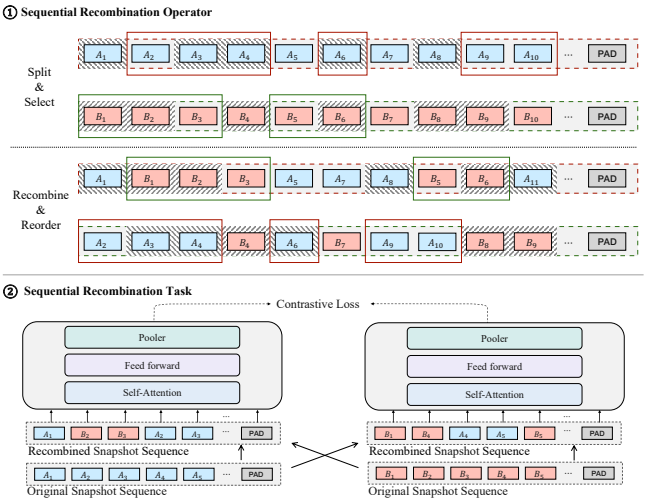

Figure 4: An illustration of the sequential recombination operator and task.

before performing the contrastive loss. We use z_i to represent the final vector of each snapshot sequence after this projection. We use a contrastive learning paradigm similar to SupCon [18]. Specifically, we take any two recombined snapshot sequences from the same pseudo-user group label as a positive example pair and take recombined sequences from different pseudo-user group labels as a negative example pair. Based on this setting, a sample may have multiple corresponding positive samples in a mini-batch training step. We follow SupCon [18] use $P(i)$ to represent the set of positive samples corresponding to sample i, so the loss function we use in the sequential recombination task can be formulated as:

$$\mathcal{L}_R = \sum_{i=1}^{R} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp\left(z_i \cdot z_p / \tau_r\right)}{\sum_{j=1}^{2R} \mathbb{1}_{[j \neq i]} \exp\left(z_i \cdot z_j / \tau_r\right)} \tag{4}$$

where R is the mini-batch size used in the sequential recombination task, and τ_r is a manually set temperature parameter.

3.6 Multi-task Training

Finally, we jointly optimize the sequential mutation task and the sequential recombination task by simply adopting a multi-task training strategy. The total loss is a linear weighted sum as follows:

$$\mathcal{L}_{total} = \lambda_1 * \mathcal{L}_M^v + \lambda_2 * \mathcal{L}_M^w + \lambda_3 * \mathcal{L}_R \tag{5}$$

where λ_1, λ_2, and λ_3 are hyperparameters set manually. In our experiments, the default values of λ_1, λ_2, and λ_3 are all set to 1.

4 EXPERIMENT

4.1 Experimental Setting

Pre-training datasets. We implement our proposed user behavior modeling framework on real-world data collected from multiple online e-commerce scenarios in Alibaba. Table 1 shows the number of involved samples and users for each of our proposed pre-training tasks. In the experiment section, "MAM" refers to the Mask Action Modeling task we used when pre-training the snapshot encoder, "SeqMuta." refers to the sequential mutation task, and "SeqReco." refers to the sequential recombination task. For the MAM task, the

Table 1: The statistical information of pre-training dataset.

Pre-train tasks	users	samples	avg.length
MAM	77.2 million	139.5 million	45.1
SeqMuta.	77.2 million	139.5 million	16.9
SeqReco.	6.8 million	54.5 million	19.9

"avg.length" is the average length of a single snapshot, that is, the average number of actions contained in the snapshot. With respect to the "SeqMuta." task and the "SeqReco." task, the "avg.length" refers to the average number of snapshots contained in the snapshot sequence.

Implementation details. Our snapshot encoder is a 6-layer, 8-head transformer encoder with the hidden size of 512, and so is our summarizer module. The extra decoder we used in the sequential mutation task is a 6-layer, 8-head transformer decoder. Following the weight tying technology widely used in NLP, we tie the parameters of our summarizer and decoder in the sequential mutation task training process. The extra projector we employ in the "SeqReco." task is an MLP with BatchNorm [15]. We use Adam [19] with a weight decay of 0.01 as the optimizer for all our pre-training tasks. For the snapshot encoder pre-training, we use a mini-batch size of 2048 and an initial learning rate of $2e-4$. In terms of the joint training of the "SeqMuta." task and the "SeqReco." task, the initial learning rate is set to $2e-5$ and the mini-batch size for the "SeqMuta." task and the "SeqReco." task is set to 512 and 256, respectively. The N in section 3.2 is set to 64 (which contains up to 48 pre-transaction actions and up to 16 post-transaction actions), and the T is set to 32. The τ_m used in section 3.4 and τ_r used in section 3.5 are all set to 0.07. It takes about 2 days to train the "SeqMuta." task by using 4 NVIDIA A100 GPUs with automatic mixed precision training, while the joint training of the "SeqMuta." task and the "SeqReco." task takes about 3.5 days. In the inference stage, the inference speed of the SAGE model is 205 samples per second by using a single NVIDIA 2080Ti GPU. For downstream fine-tuning, we take the `[CLS]` representation output by the last layer of the summarizer as the sequential representation.

Evaluation datasets. We conduct experiments for fraud transaction detection on four real business scenarios, *i.e.,* Tmall Supermarket, Tmall Global, Taobao, and Ali Health, each with different characteristics. For example, Tmall Supermarket sells daily necessities, and fraud transaction usually occurs in the form of batch accounts. Taobao sells mobile phones and home appliances, and fraud transaction manifests as long-term abnormal behavior of a single account. Note that the data is user behavior sequences, so we divide the data (ranging from 2022/09/04 to 2022/10/23) into the training and testing set at the user level. The statistical information of the training and testing data for four real business scenarios is shown in Table 2. In addition, the fraud rate in Table 2 does not reflect the fraud rate of transactions in Alibaba because we randomly sampled from fraudulent and benign transactions, respectively.

Evaluation protocol. In our experiments, we first conduct experiments to compare our method with a variety of state-of-the-art methods by training the entire model on the fraud detection task on four scenes. Then, to directly compare the performance of different pre-training tasks, we compare the representation quality by freezing the pre-trained model and only fine-tuning a one-layer MLP classifier on the downstream task. We measure the performance of different methods in terms of AUC, AP (Average Precision), and recall at 90% precision. In our scenario, we must ensure our model has high precision to avoid disrupting normal users. The precision over 90% is essential for a model to be deployed. So we observe the recall rate at 90% precision to compare different methods. In this paper, we represent the recall rate at 90% precision as "$\mathbf{R@P_{0.9}}$".

Table 2: The statistical information of the Fraud Transaction Detection dataset.

Scenes	Split	Fraud	Benign	Fraud Rate
Tmall Supermarket	Training	1,157,952	23,155,912	4.76%
	Testing	64,901	1,285,462	4.81%
Tmall Global	Training	907,810	18,150,125	4.76%
	Testing	51,102	1,011,520	4.81%
Taobao	Training	1,486,549	29,765,243	4.76%
	Testing	82,600	1,655,778	4.75%
Ali Health	Training	330,538	6,602,033	4.77%
	Testing	18,077	368,006	4.68%

4.2 Comparison of Various Methods

We first conduct experiments to compare our model with a variety of existing methods [9, 34]. It should be noted, due to the user intentions used in [21] were derived from 7655 manually defined user behavior sub-sequence, but [21] did not publicly disclose the details of the user intention definition for reproduction. Hence, we compare our method in this section with three state-of-the-art sequence modeling methods, *i.e.,* BiLSTM-max [9], HConvNet [9], and transformer [34]. To make a fair comparison, we followed the method we proposed to use multiple snapshot inputs and two-stage cascaded base models for each method. These improved models are named BiLSTM-max+, HConvNet+, and transformer+, respectively. The results are shown in Table 3.

Comparing our method SAGE with other methods in the upper part of the Table 3, it is shown that the SAGE method can significantly improve the performance of the baseline model (transformer+) on the fraud transaction detection task in all four scenarios. With a precision of 0.9, the recall rate of the SAGE method has an improvement of about +4.93% to the transformer+ method and about +3.44% to the best model without pre-training, *i.e.,* HConvNet. Those results reflect the effectiveness of our proposed pre-training method.

Label lacking scenario evaluation. To simulate the scenario lacking annotation labels (or cold start scenario), we conduct the experiments by using only 1% of the training labeled data. The results are shown in the lower part of the Table 3. The results show that in the scenario lacking annotation labels, the SAGE method can provide a more significant improvement than the baseline method. With a precision of 0.9, the recall rate of the SAGE method has an improvement of about +7.88% to the transformer+ method and about +7.51% to the HConvNet method.

Table 3: Overall performance of various methods on 4 real-world fraud transaction detection scenarios with the whole and 1% training labels. The evaluation metrics include AP (Average Precision), AUC, and recall at 90% precision, represented as R@P$_{0.9}$.

Methods	Tmall Supermarket			Tmall Global			Taobao			Ali Health			Avg.		
	AP	AUC	R@P$_{0.9}$	AP	AUC	R@P$_{0.9}$	AP	AUC	R@P$_{0.9}$	AP	AUC	R@P$_{0.9}$	AP	AUC	R@P$_{0.9}$
Use the whole training labels															
BiLSTM-Max	0.8463	0.9849	0.6105	0.9135	0.9946	0.7444	0.8682	0.9896	0.5889	0.9214	0.9904	0.8054	0.8874	0.9899	0.6863
HConvNet	0.8416	0.9843	0.5990	0.9098	0.9943	0.7216	0.8629	0.9891	0.5662	0.9179	0.9899	0.7917	0.8831	0.9894	0.6696
transformer	0.8160	0.9807	0.5263	0.8886	0.9924	0.6536	0.8392	0.9863	0.5012	0.8994	0.9881	0.7489	0.8608	0.9869	0.6075
BiLSTM-Max+	0.9100	0.9928	0.7763	0.9551	0.9973	0.9024	0.9218	0.9937	0.7731	0.9435	0.9924	0.8652	0.9326	0.9941	0.8293
HConvNet+	0.9163	0.9936	0.7963	0.9609	0.9977	0.9164	0.9296	0.9946	0.7934	0.9509	0.9932	0.8829	0.9394	0.9948	0.8473
transformer+	0.9125	0.9932	0.7807	0.9564	0.9974	0.9018	0.9224	0.9937	0.7762	0.9467	0.9931	0.8707	0.9345	0.9944	0.8324
SAGE (ours)	**0.9338**	**0.9953**	**0.8321**	**0.9687**	**0.9982**	**0.9382**	**0.9455**	**0.9961**	**0.8412**	**0.9630**	**0.9947**	**0.9151**	**0.9528**	**0.9961**	**0.8817**
imporv.(transformer+)	+2.13%	+0.21%	+5.14%	+1.23%	+0.08%	+3.64%	+2.31%	+0.24%	+6.50%	+1.63%	+0.16%	+4.44%	+1.83%	+0.17%	+4.93%
Only use 1% training labels															
BiLSTM-Max	0.7782	0.9729	0.4754	0.8692	0.9905	0.5773	0.7994	0.9810	0.3895	0.8236	0.9786	0.5667	0.8176	0.9808	0.5022
HConvNet	0.7796	0.9738	0.4671	0.8682	0.9906	0.5640	0.7950	0.9812	0.3703	0.8310	0.9803	0.5579	0.8185	0.9815	0.4898
transformer	0.7582	0.9695	0.4269	0.8425	0.9867	0.5027	0.7748	0.9773	0.3296	0.8210	0.9791	0.5397	0.7991	0.9782	0.4497
BiLSTM-Max+	0.8783	0.9884	0.6997	0.9369	0.9960	0.8500	0.8842	0.9888	0.6751	0.8946	0.9863	0.7361	0.8985	0.9899	0.7402
HConvNet+	0.8784	0.9891	0.7033	0.9374	0.9962	0.8490	0.8919	0.9907	0.6826	0.8979	0.9872	0.7510	0.9014	0.9908	0.7465
transformer+	0.8766	0.9893	0.7019	0.9378	0.9959	0.8378	0.8816	0.9874	0.6720	0.9100	0.9897	0.7594	0.9015	0.9906	0.7428
SAGE (ours)	**0.9084**	**0.9923**	**0.7725**	**0.9522**	**0.9971**	**0.8981**	**0.9175**	**0.9930**	**0.7645**	**0.9376**	**0.9912**	**0.8514**	**0.9289**	**0.9934**	**0.8216**
imporv.(transformer+)	+3.18%	+0.30%	+7.06%	+1.44%	+0.12%	+6.03%	+3.59%	+0.56%	+9.25%	+2.76%	+0.15%	+9.20%	+2.74%	+0.28%	+7.88%

Note that: (i) transformer here was not pre-trained. More results are provided in Table4. (ii) BiLSTM-Max+, HConvNet+, and transformer+ are three models that consist of two cascaded BiLSTM-Max, HConvNet, and transformer, respectively. These two cascaded models can capture long-term user behavior patterns in sequence data. (iii) By training on the dataset with only 1% labeled examples, the performance of methods in real-world scenarios where labeled data is scarce can be evaluated.

Table 4: Comparison of different pre-training methods on 4 real-world fraud transaction detection scenarios. The AUC, AP (Average Precision), and recall at 90% precision (represented as R@P$_{0.9}$) are chosen as evaluation metrics. To ensure a fair comparison, the representations are fixed, and only one layer of the MLP classifier is updated during parameter tuning.

Pre-train tasks	Tmall Supermarket			Tmall Global			Taobao			Ali Health			Avg.		
	AP	AUC	R@P$_{0.9}$	AP	AUC	R@P$_{0.9}$	AP	AUC	R@P$_{0.9}$	AP	AUC	R@P$_{0.9}$	AP	AUC	R@P$_{0.9}$
short-term modeling															
MLM* (baseline)	0.8152	0.9806	0.5303	0.8903	0.9924	0.6674	0.8331	0.9859	0.4742	0.8915	0.9874	0.7238	0.8575	0.9866	0.5989
MAM	0.8343	0.9838	0.5713	0.9007	0.9936	0.6906	0.8498	0.9880	0.5273	0.9019	**0.9892**	**0.7481**	0.8717	0.9886	0.6343
long-term modeling w/. MAM															
MSM (baseline)	0.8906	0.9902	0.7156	0.9322	0.9949	0.8208	0.8731	0.9861	0.6528	0.8844	0.9860	0.6950	0.8951	0.9893	0.7211
SeqMuta.	0.8900	0.9904	0.7061	0.9366	0.9955	0.8335	0.8855	0.9891	0.6777	0.9008	0.9881	0.7359	0.9032	0.9908	0.7383
SeqMuta.+SupCon*	0.8956	**0.9917**	**0.7257**	0.9407	0.9960	0.8341	0.8921	0.9891	0.6977	0.8935	0.9876	0.7078	0.9055	0.9910	0.7413
SeqMuta.+SeqReco.	**0.8956**	0.9912	0.7237	**0.9428**	**0.9962**	**0.8490**	**0.8955**	**0.9901**	**0.7005**	**0.9050**	0.9888	0.7366	**0.9097**	**0.9916**	**0.7525**

Note that: (i) the transformer model was used for short term modeling, while transformer+ model was used for long term modeling. (ii) MLM* and SupCon* are our implementations of the MLM[24] and SupCon[18], respectively. (iii) SeqMuta. and SeqReco. are abbreviations for the proposed Sequential Mutation and Sequential Recombination tasks , respectively.

4.3 Comparison of Pre-training Methods

As mentioned in section 4.2, we compared the performance of pre-training methods by freezing the representations and only fine-tuning a one-layer MLP classifier on the downstream tasks. For short-term behavior sequence modeling, we compare the performance of the methods whether they use the action duration token, corresponding to the row "MAM" and the row "MLM*" in Table 4, respectively. Specifically, MLM* only uses action tokens as input. Table 4 shows that action duration as an additional input can significantly improve the performance of fraud transaction detection tasks in all four business scenarios. For long-term behavior sequence modeling, to fairly compare with our method, we first implement a "mask-then-predict" pre-training method "MSM" as our long-term modeling baseline, which masks 30% snapshots in each snapshot sequence and then encourages the model to predict them. The MSM method has the same input form and model size as our method and uses the same form of loss function as $\mathcal{L}_M$ for pre-training. And for downstream fine-tuning, the MSM method takes the mean pooling of the last hidden state of the last layer of the summarizer as the sequential representation. In Table 4, we demonstrate the performance of our proposed tasks, where the row "SeqMuta." shows the performance of the pre-training model using only the sequential mutation task, and the row "SeqMuta.+SeqReco." shows the performance of joint training sequential mutation task and sequential recombination task. Comparing the baseline task MSM with the "SeqMuta." task, the result indicates that our proposed sequential mutation task can improve the quality of user behavior representation for fraud transaction detection tasks by using more noising operators. More results (different mask ratios or pooling methods for MSM) not shown in Table 4 are consistent with this conclusion. Comparing the "SeqMuta." task with the "SeqMuta.+SeqReco." task, it can be found that the joint training of our proposed tasks can further improve the performance of pre-trained user behavior representation, which verifies the effectiveness of utilizing user-related information to enhance performance for fraud transaction detection tasks. Finally, compare the short-term modeling methods with the long-term modeling methods in Table 4. It can be found that long-term modeling methods can significantly improve the

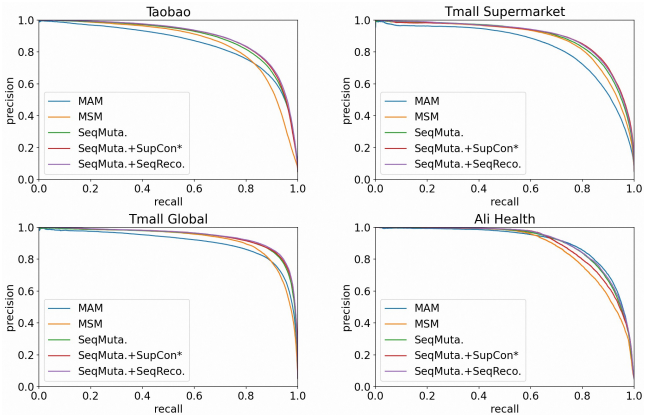

Figure 5: PR curve of various pre-training methods on 4 fraud transaction detection scenes. Best view in color and zoom in.

performance on downstream tasks, which verifies the necessity of long-term behavior modeling. We draw the PR curves of the best short-term method (MAM) and all the long-term methods in Fig.5. The Fig.5 shows that our proposed "SeqMuta.+SeqReco." tasks have the best performance.

4.4 Impact of Recombination Operator

We conduct ablation studies on the impact of components in the sequential recombination task. Specifically, we implement a baseline contrastive learning method that directly takes the snapshot sequence pairs from the same pseudo user relation group as positive pairs and the snapshot sequence pairs from different user groups as negative pairs. The experimental result of this method is shown as the row "SeqMuta.+SupCon*" in Table 4. In other words, the only difference between row "SeqMuta.+SupCon*" and row "SeqMuta.+SeqReco." is whether use the sequential recombination operator. Comparison of the last two rows in Table 4 indicates that the performance improvement of our sequential recombination task is not only brought by the contrastive learning paradigm but also partially from the sequential recombination operator we designed.

4.5 Impact of Mutation Toxicity Ratios

In this section, we explore the effects of different toxicity ratios on the sequential mutation task. The results are shown in Table 5. The α_I, α_S, and α_D donate the toxicity ratios of the insertion, substitution and deletion mutation operators, respectively. The values of the evaluation metrics are the average of 4 scenarios. For all experiments, we only perform the sequential mutation task on the pre-trained snapshot features. We first conduct three experiments by setting the same toxicity ratio of 0.2, 0.3 and 0.5 for all mutation operators. The experimental results show that the best set of the unique toxicity ratio is 0.5, which is distinct from the often chosen mask ratio of around 0.15 in MLM-like tasks. Next, we conduct experiments in a fixed α_D ratio at 0.5 and then modify the toxicity ratios of the other two operations, corresponding to the last three columns in Table 5. Experimental results show that the optimal setting is 0.5 for α_D and 0.2 for other mutation operations, which indices that there are different optimal toxicity ratios for different types of mutation operations.

Table 5: Comparison of the impact of various toxicity ratios used in sequential mutation operators. The values of the evaluation metrics are the average of 4 scenarios.

Metrics	α_I=0.2 α_S=0.2 α_D=0.2	α_I=0.3 α_S=0.3 α_D=0.3	α_I=0.5 α_S=0.5 α_D=0.5	α_I=0.2 α_S=0.2 α_D=0.5	α_I=0.3 α_S=0.3 α_D=0.5
AP	0.9038	0.9032	0.9058	**0.9065**	0.9033
AUC	0.9909	0.9908	0.9912	**0.9913**	0.9909
R@P$_{0.9}$	0.7381	0.7383	0.7451	**0.7456**	0.7398

Table 6: Comparison of online simulation performances. The SAGE† denotes the quantized SAGE model. The evaluation metric is the recall at the same precision as the online model.

Scenes	online	online+SAGE†	imporv.
Tmall Supermarket	0.8482	**0.9071**	+5.89%
Tmall Global	0.7523	**0.8138**	+6.15%

4.6 Online Simulation

In our scenario, due to the user behavior logs cannot be obtained in time on the server side, the SAGE model is planned to deploy on the client side (*e.g.,* the APPs), which requires distilling and quantizing the SAGE model to reduce its resources consume. Hence, we conduct an online simulation experiment to evaluate whether the quantized SAGE model can help prevent fraudulent transactions. For realistic evaluation, a full week of test data (ranging from 2022/10/24 to 2022/10/30) was used. The results are shown in Table 6, and the recall at the same precision as the online model is adopted as the metric. The results show the quantized SAGE model can significantly improve the recall of online models without compromising precision.

5 CONCLUSION AND FUTURE WORK

In this paper, we proposed SAGE, a general user behavior modeling framework for fraud transaction detection. From the perspective of "*Sequence As Genes*", we intercepted the crucial segments of the long-term user behavior sequences to decouple the length of behavior sequences and the corresponding time spans. We devised the sequential mutation task and the sequential recombination task to obtain robust and effective long-term user behavior representations. We pre-trained our SAGE model on large-scale behavioral data on the Alibaba platform and validated its performance on downstream fraud transaction detection tasks. Experimental results on four real scenarios fully demonstrated the effectiveness of the proposed framework. For future work, we believe that the interpretability of the SAGE model for fraud transaction detection needs to be put on the agenda to provide better services. Besides, it is essential to integrate with graph neural networks, which can play a crucial role in discovering the inter-relationships of users.

REFERENCES

[1] Siddhartha Bhattacharyya, Sanjeev Jha, Kurian Tharakunnel, and J Christopher Westland. 2011. Data Mining for Credit Card Fraud: A Comparative Study. *Decision support systems* 50, 3 (2011), 602–613.

[2] Richard J Bolton and David J Hand. 2002. Statistical Fraud Detection: A Review. *Statist. Sci.* 17, 3 (2002), 235–255.

[3] Bernardo Branco, Pedro Abreu, Ana Sofia Gomes, Mariana SC Almeida, João Tiago Ascensão, and Pedro Bizarro. 2020. Interleaved Sequence RNNs for Fraud Detection. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3101–3109.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of International conference on machine learning*. 1597–1607.

[5] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. 2020. Big Self-supervised Models Are Strong Semi-supervised Learners. *Advances in Neural Information Processing Systems* 33 (2020), 22243–22255.

[6] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. 2020. Improved Baselines with Momentum Contrastive Learning. *arXiv preprint arXiv:2003.04297* (2020).

[7] Xinlei Chen and Kaiming He. 2021. Exploring Simple Siamese Representation Learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 15750–15758.

[8] Xinlei Chen, Saining Xie, and Kaiming He. 2021. An Empirical Study of Training Self-Supervised Vision Transformers. In *Proceedings of IEEE International Conference on Computer Vision*. 9640–9649.

[9] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, 670–680.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. 4171–4186.

[11] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*.

[12] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. 2020. Bootstrap Your Own Latent: A New Approach to Self-supervised Learning. *Advances in Neural Information Processing Systems* 33 (2020), 21271–21284.

[13] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked Autoencoders Are Scalable Vision Learners. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 16000–16009.

[14] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 9729–9738.

[15] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of International Conference on Machine Learning*. 448–456.

[16] Johannes Jurgovsky, Michael Granitzer, Konstantin Ziegler, Sylvie Calabretto, Pierre-Edouard Portier, Liyun He-Guelton, and Olivier Caelen. 2018. Sequence Classification for Credit-card Fraud Detection. *Expert Systems with Applications* 100 (2018), 234–245.

[17] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. 2021. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications* 80 (2021), 8091–8126.

[18] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised Contrastive Learning. *Advances in Neural Information Processing Systems* 33 (2020), 18661–18673.

[19] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations*.

[20] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*. 7871–7880.

[21] Can Liu, Yuncong Gao, Li Sun, Jinghua Feng, Hao Yang, and Xiang Ao. 2022. User Behavior Pre-training for Online Fraud Detection. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3357–3365.

[22] Can Liu, Li Sun, Xiang Ao, Jinghua Feng, Qing He, and Hao Yang. 2021. Intention-aware Heterogeneous Graph Attention Networks for Fraud Transactions Detection. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3280–3288.

[23] Can Liu, Qiwei Zhong, Xiang Ao, Li Sun, Wangli Lin, Jinghua Feng, Qing He, and Jiayu Tang. 2020. Fraud Transactions Detection via Behavior Tree With Local Intention Calibration. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3035–3043.

[24] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A Robustly Optimized Bert Pretraining Approach. *arXiv preprint arXiv:1907.11692* (2019).

[25] Melanie Mitchell. 1998. *An Introduction to Genetic Algorithms*. MIT press.

[26] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748* (2018).

[27] Jon T.S. Quah and M. Sriganesh. 2008. Real-time Credit Card Fraud Detection Using Computational Intelligence. *Expert Systems with Applications* 35, 4 (2008), 1721–1732.

[28] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near Linear Time Algorithm to Detect Community Structures in Large-scale Networks. *Physical review E* 76, 3 (2007), 036106.

[29] Saharon Rosset, Uzi Murad, Einat Neumann, Yizhak Idan, and Gadi Pinkas. 1999. Discovery of Fraud Rules For Telecommunications—challenges and Solutions. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 409–413.

[30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[31] Jianlin Su, Jiarun Cao, Weijie Liu, and Yangyiwen Ou. 2021. Whitening sentence representations for better semantics and faster retrieval. *arXiv preprint arXiv:2103.15316* (2021).

[32] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China, 1556–1566.

[33] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. 2022. VideoMAE: Masked Autoencoders are Data-Efficient Learners for Self-Supervised Video Pre-Training. In *Advances in Neural Information Processing Systems*.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *Advances in Neural Information Processing Systems* 30 (2017).

[35] Kexin Wang, Nils Reimers, and Iryna Gurevych. 2021. TSDAE: Using Transformer-based Sequential Denoising Auto-Encoderfor Unsupervised Sentence Embedding Learning. In *Empirical Methods in Natural Language Processing (EMNLP)*.

[36] Tyra G Wolfsberg, Johanna McEntyre, and Gregory D Schuler. 2001. Guide to the Draft Human Genome. *Nature* 409, 6822 (2001), 824–826.

[37] Dongbo Xi, Fuzhen Zhuang, Bowen Song, Yongchun Zhu, Shuai Chen, Dan Hong, Tao Chen, Xi Gu, and Qing He. 2020. Neural Hierarchical Factorization Machines for User's Event Sequence Analysis. In *Proceedings of International Conference on Research on Development in Information Retrieval*. 1893–1896.

[38] Wenhan Xiong, Anchit Gupta, Shubham Toshniwal, Yashar Mehdad, and Wen-tau Yih. 2022. Adapting Pretrained Text-to-Text Models for Long Text Sequences. *arXiv preprint arXiv:2209.10052* (2022).

[39] Dianmin Yue, Xiaodan Wu, Yunfeng Wang, Yue Li, and Chao-Hsien Chu. 2007. A Review of Data Mining-based Financial Fraud Detection Research. In *Proceedings of IEEE International Conference on Wireless Communications, Networking and Mobile Computing*. 5519–5522.

[40] Panpan Zheng, Shuhan Yuan, and Xintao Wu. 2019. SAFE: A Neural Survival Analysis Model for Fraud Early Detection. In *Proceedings of AAAI Conference on Artificial Intelligence*, Vol. 33. 1278–1285.