Online Fraud Detection via Test-Time Retrieval-Based Representation Enrichment

Yiran Qiao[1,2,3*], Ningtao Wang[4*], Yuncong Gao[1,2,3*], Yang Yang[4], Xing Fu[4], Weiqiang Wang[4], Xiang Ao[1,2,3†]

[1]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, China
[2]Key Lab of AI Safety, Chinese Academy of Sciences, Beijing 100094, China
[3]University of Chinese Academy of Sciences, CAS, Beijing 100049, China
[4]Independent Researcher
yrqiao@gmail.com, ntwang25@gmail.com, gaoyuncong21s@ict.ac.cn, name_yy@126.com, fux008@gmail.com, wang.weiqiang@gmail.com, aoxiang@ict.ac.cn

Abstract

Anti-fraud machine learning systems are perpetually confronted with the significant challenge of *concept drift*, driven by the continuous and intense evolution of fraudulent techniques. That is, outdated models trained on historical fraudulent behaviors often fall short in addressing the evolving tactics of malicious users over time. The key issue lies in effectively tackling the rapid and significant evolution of fraudsters' behaviors to detect these emerging and unforeseen anomalies. In this paper, we propose a solution by directly accessing real-time data and introducing a lightweight plug-in approach named TRE (Test-time Retrieval-based Representation Enrichment). Considering the similarity among samples, TRE employs a retriever to efficiently identify the top-K most relevant recent samples and implements an aggregation strategy to provide neighboring embeddings to the predictor. It thus adjusts the trained classifiers during the test time, providing them with the information from the latest unlabeled data. Extensive experiments on three large-scale real-world datasets demonstrate the superiority of TRE. By consistently incorporating information from the nearest neighbors, TRE demonstrates high adaptability and surpasses existing methods in performance.

Introduction

Online services typically evolve more rapidly than the formulation of regulatory measures, leaving fraudsters with potential vulnerabilities to exploit. Meanwhile, fraudsters swiftly adapt to new trends and technologies, consistently developing advanced and inventive methods to evade detection by platform anti-fraud systems (Liu et al. 2021a; Li et al. 2022b; Zhang et al. 2022). As a result, the outdated models, trained on historical fraudulent behaviors, struggle to detect the continually evolving patterns exhibited by malicious users effectively. This issue, where the distribution of a data stream varies over time, is known as *concept drift* (Gama et al. 2014), which has long been a central challenge in fraud detection (Soemers et al. 2018; Hu et al. 2021).

For example, Figure 1 shows the AUC performance of two fraud detection models on an online payment platform in China. These models were trained on user behavior sequence data up to 09/2022 and evaluated on five monthly test datasets from 10/2022 to 02/2023. The trend of decreasing performance observed in both models highlights a significant case of *incremental concept drift* (Lu et al. 2019). Outdated models fail to identify evolving fraud tactics, progressively exposing more vulnerabilities that fraudsters can exploit. Consequently, the urgency to address this concept drift escalates as performance degradation worsens.

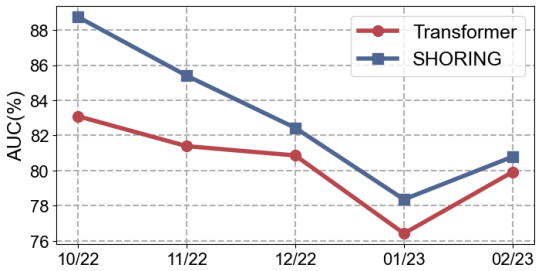


Figure 1: The AUC of Transformer (Vaswani et al. 2017) and SHORING (Li et al. 2022a) on five out-of-time (OOT) test datasets. Both incremental and recurring concept drifts can be inferred from the observed performance decreases.

Additionally, a notable decline in performance is observed in January 2023, coinciding with the Chinese New Year holiday. The approaching of significant holidays typically presents opportune moments for executing scams. Fraudsters are motivated by the pressure to boost their earnings for festive expenses, and facilitated by people's lowered vigilance and increased spending. This surge in fraudulent activities challenges outdated models to capture the

*These authors contributed equally.
†Corresponding Author. Xiang Ao is also at CASMINO Ltd., Suzhou 215000, China.

ever-evolving behavioral patterns. The subsequent rebound in performance in the following month also confirms this interpretation. Such a significant recurrent fall exemplifies another typical concept drift, i.e., *recurring concept drift* (Lu et al. 2019), posing severe fraudulent risks periodically.

To tackle the issue of concept drift, an intuitive solution is periodical re-training (Kantchelian et al. 2013; Pendlebury et al. 2019). However, the high cost of periodic re-training may make it impractical. The rapid evolution of fraud behaviors combined with the massive data volumes from online platforms makes frequent re-training excessively resource-intensive and infeasible (Tounekti, Ruiz-Martinez, and Gómez 2019).

Hence, an advanced strategy is to integrate real-time data during the inference stage, known as test-time training (TTT), to adapt to the latest data distribution. Following this route, some approaches apply self-supervised tasks such as rotation prediction (Sun et al. 2020), self-reconstruction (Prabhudesai et al. 2023), and student-teacher feature prediction (Bartler et al. 2022) to implicitly guide models in learning the current distribution. Other methods modify loss functions (Liang, Hu, and Feng 2020; Wang et al. 2020) or leverage reliable samples (Niu et al. 2022) to reduce the discrepancy between the model's predictions and the test data distribution.

Despite significant advances in TTT research, whether through the use of proxy tasks or modifications to training objectives, these approaches still fall short of directly addressing the fundamental changes in data distributions. To mitigate this gap, we leverage the most recent data from the test dataset to build a retrieval database, serving as a memory bank to help the model perceive and adapt to the latest data distribution.

In this paper, we propose a lightweight retrieval-based approach, Test-time Retrieval-based Representation Enrichment (**TRE** for short), which can be incorporated with almost all the fraud detection frameworks. TRE adopts a retriever to recognize top-K relevant samples from the retrieval set, i.e., the test dataset. Given that the performance of the K-Nearest Neighbor (KNN) algorithm is highly sensitive to the parameter k, we develop an adaptive-k component to optimize the performance of TRE. After obtaining dense representations aggregated from neighboring samples, we devise an aggregation strategy to incorporate this updated information into the predictor of TRE. Additionally, we offer a lightweight version of TRE that replaces KNN methods with SimHash-based indexing to balance efficiency and effectiveness, addressing the high time complexity of KNN methods. By consistently integrating nearest neighbor information, TRE exhibits remarkable adaptability and surpasses existing methods in performance.

We conduct extensive experiments on three real-world datasets from industry to illustrate the superiority of TRE. The results show that TRE can significantly enhance existing fraud detection models based on user behavior sequences. The main contributions are summarized as follows:

- We emphasize the significance of test-time training in dynamic risk discrimination scenarios. TTT is essential for enhancing a model's robustness and predictive capacity when dealing with distribution-shift data.
- We propose the Test-time Retrieval-based Representation Enrichment (TRE) method, a lightweight plug-in approach, which addresses data concept drift by introducing an adaptive KNN retriever and representation enrichment techniques.
- Extensive experiments on three real-world datasets demonstrate the state-of-the-art performance and broad applicability of our method TRE.

Related Work

Retrieval-Enhanced Representation Learning

Retrieval-enhanced models update input representations using retrieved items, which contain patterns that help the model learn more expressive representations (Zamani et al. 2022). To name some, Guided Transformer (Hashemi, Zamani, and Croft 2020) incorporates cross-attention to enhance input contextualization with information retrieved from a text corpus. NMIR (Hashemi, Zamani, and Croft 2021) then introduces a methodology for learning multiple representations of query intents using retrieval results and the Guided Transformer network for representation adaptation. RETRO (Borgeaud et al. 2022) demonstrates that retrieval using the Guided Transformer can produce superior outcomes with smaller models. Similarly, in the computer vision domain, T-MAD (Xu et al. 2021) retrieves texture memory to assist in image in-painting. Mask2CAD (Kuo et al. 2020) utilizes retrieval from a dataset of 3D models to comprehend the underlying 3D structure of objects depicted in a 2D image.

Databases retrieved by existing retrieval-enhanced representation learning methods are often sourced from large external corpora or generated from training data. While these methods enhance knowledge, they fail to effectively address distribution drift in test datasets. In contrast, TRE constructs a retrieval database directly from the test dataset using an unsupervised approach.

Test-Time-Training under Distribution Shift

Test-time training (TTT) is an approach that entails integrating test data during the inference stage to optimize the model, thereby enhancing its performance (Sun et al. 2020).

Some test-time training methods use agent tasks to update model parameters. For instance, the original study (Sun et al. 2020) uses rotation prediction as an auxiliary task to predict the rotation angle of rotated images. TTT++ (Liu et al. 2021b) substitutes this task for a contrastive learning task. TTT+MAE (Gandelsman et al. 2022) employs a masking autoencoder reconstruction task. These tasks not only regularize the model during training but also improve performance when optimizing on test samples (Tula et al. 2023).

Another group of test-time training methods updates the original model at test time using pseudo-losses, such as entropy, without modifying the training strategy. For example, SHOT (Liang, Hu, and Feng 2020) minimizes the expected entropy of individual predictions while maximizing the entropy of predicted classes across the entire test set.

TENT (Wang et al. 2020) modifies batch normalization layers, and EATA (Niu et al. 2022) selectively minimizes the entropy of output predictions.

Current TTT methodologies exhibit shortcomings in effectively addressing rapid data distribution. However, it is crucial to acknowledge the dynamic and impactful evolution of fraudulent activities in online fraud detection scenarios. Our primary goal is to enhance our ability to promptly respond to emerging risks by directly accessing real-time data.

Methodology

Problem Statement

In this paper, we formulate our task as a binary sequence classification problem. For each target transaction i, a prior user behavior sequence $x_i = \{x_{i1}, x_{i2}, \cdots, x_{iT}\}$ is provided, with T as the sequence length. The behavior sequence consists of user click actions collected from an online payment App and encoded already. No further details about the target transaction i are accessible, except for the behavior sequence. Each transaction i is labeled by $y_i \in \{0, 1\}$ to indicate whether it is fraudulent $(y_i = 1)$ or not.

Formally, the task is to learn a model f_W to classify the transactions based on their prior behaviors: $f_W : \{x_i\} \rightarrow \mathbf{y} = \{y_i\}$. Given a set of labeled transactions $\mathbb{D}_{\text{train}} = \{(x_i, y_i)\}_{i \in \text{train}}$, our goal is to predict the fraudulent nature of some unlabeled transactions $\mathbb{D}_{\text{test}}$. To address data concept drift, we employ the unlabeled $\mathbb{D}_{\text{test}}$ to construct our retrieval dataset, denoted as $\mathbb{D}_{\text{retrieval}}$.

Overview

As shown in Figure 2(a), the proposed TRE consists of four components: an encoder E, an index, a retriever R, and an aggregator to produce the final prediction.

Based on the four components, the pipeline of the TRE framework includes four modules: (i) **Index Building**: we employ the encoder E to encode every instance from the dataset $\mathbb{D}_{\text{retrieval}}$ to generate an embedded retrieval set $\mathcal{D}_{\text{retrieval}}$, upon which an index of embeddings is built. (ii) **Query Generation**: The same encoder E is used to map an input instance x_q to the embedded space, generating the query embedding $\mathbf{x_q}$. (iii) **Retrieval**: Via the retriever R, we utilize the query embedding $\mathbf{x_q}$ to retrieve the top-K instances from the index, forming the retrieved set $\mathbb{S}_q$. (iv) **Response Utilization**: To leverage the retrieval response $\mathbb{S}_q$, the aggregator first condenses the top-K embeddings within $\mathbb{S}_q$ and the query embedding $\mathbf{x_q}$, forming aggregated embeddings based on different values of k for KNN. Then, these aggregated embeddings are passed to f_θ to generate predictive logits for each k. Moreover, based on the aggregated embeddings and the logits from f_θ, an adaptive-k component determines the optimal value for k through f_σ, subsequently making the final prediction (c.f. Figure 2(b)).

Index Building & Query Generation

Encoder We use the same encoder E to embed every input instance to generate queries, and map the retrieval set $\mathbb{D}_{\text{retrieval}}$ to the embedded space for index building:

$$\mathbf{x_q} = E(x_q), \tag{1}$$

$$\mathcal{D}_{\text{retrieval}} = \{\mathbf{x_i} = E(x_i) : x_i \in \mathbb{D}_{\text{retrieval}}\}. \tag{2}$$

Here the encoder E can be implemented as any sequence model, enabling the general applicability of TRE as a plug-in method. Note that randomly initializing the encoder E results in random retrieval set embeddings, leaving KNN search inefficient. Therefore, as query instances are input in real time, we periodically update both the encoder and the index to maintain retrieval efficiency. In training, we update the encoder when saving checkpoints, which immediately triggers an index-rebuilding process. The encoder remains static when testing, i.e., when $\mathbf{x_q} \in \mathbb{D}_{\text{test}}$.

Index We employ two off-shell tools to build the index for KNN search: First, the Inverted File (IVF) method is used to cluster all input embeddings into groups. Next, within each cluster, we apply the Hierarchical Navigable Small World (HNSW) indexing method to enable efficient retrieval, ensuring an optimal time complexity $O(\log(\log|\mathbb{D}_{\text{retrieval}}|))$.

Retrieval

Since the test-time instances can improve the prediction accuracy for target instances, the retriever R aims to select the most similar samples from the test-time retrieval dataset.

For every given query embedding $\mathbf{x_q}$, based on the index, we execute a retrieval operation within $\mathcal{D}_{\text{retrieval}}$. The retriever R is used to obtain top-K similar neighbors, denoted as $\mathbb{S}_q \subseteq \mathcal{D}_{\text{retrieval}}$:

$$\mathbb{S}_q = R\left(\mathbf{x_q}, \mathcal{D}_{\text{retrieval}}\right). \tag{3}$$

The similarity between embeddings is evaluated using the L2 distance $d(\mathbf{x_q}, \mathbf{x})$.

For implementation, we apply the offline deployment of Facebook AI Similarity Search (FAISS) (Johnson, Douze, and Jégou 2019), an open-source library designed for efficient similarity search and clustering of dense embeddings.

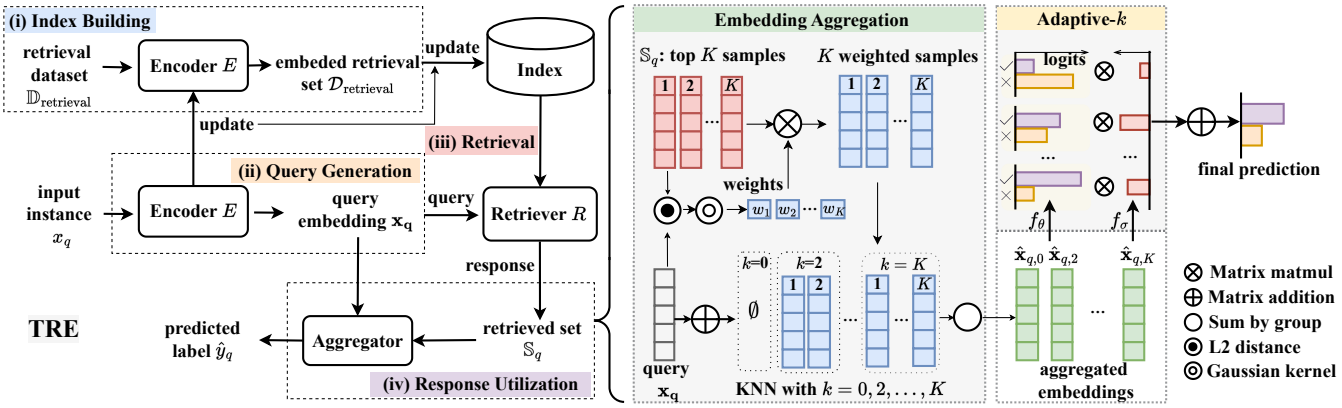
Response Utilization

Given that the retrieved samples with high noise levels might introduce excessive disturbances that overshadow their informational value (Wang et al. 2023), we employ a specially designed aggregator to filter out noises and integrate the informative samples.

Embedding Aggregation In some cases, critical samples retrieved can cause interference. For instance, when $K = 10$ and K-nearest neighbor (KNN) retrieval is performed for a negative sample, the result includes the 10 closest embeddings. If the first two retrieved embeddings are positive samples nearing the target sample's embedding, while the remaining eight are negative samples far from the query embedding, averaging the weights of these retrieved embeddings may introduce noises instead.

To address this concern, we employ a weighting strategy on the retrieved set of embeddings $\mathbb{S}_q$, before aggregating these retrieved embeddings and the query embedding to obtain the ultimate composite embedding.

Specifically, we compute the weights for each retrieved embedding through the Gaussian kernel. For the query embedding $\mathbf{x_q}$ and any dense embedding $\mathbf{x}$ in $\mathbb{S}_q$, we define the



(a) The entire process (b) The structure of the aggregator

Figure 2: An overview of the proposed TRE. TRE considers the similarity between samples and employs a retriever to efficiently locate the top-K relevant latest samples from the index, aiding the network in making label predictions by applying embedding aggregation with the adaptive-k component.

Gaussian kernel function as

$$W(\mathbf{x_q}, \mathbf{x}) = \exp\left\{-\frac{d^2(\mathbf{x_q}, \mathbf{x})}{2\sigma^2}\right\}, \tag{4}$$

where σ is a hyperparameter. Then we have

$$\xi(\mathbf{x_q}) = \sum_{\mathbf{x} \in \mathbb{S}_{\text{u}}} W(\mathbf{x_q}, \mathbf{x}) \cdot \mathbf{x}. \tag{5}$$

The weighted composite embedding $\xi(x_q)$ is used to enhance the original embedding x_q as in:

$$\hat{\mathbf{x}}_\mathbf{q} = \mathbf{x_q} + \lambda \cdot \xi(\mathbf{x_q}), \tag{6}$$

where λ is a learnable network parameter.

Adaptive-k K-nearest neighbor (KNN) algorithms typically exhibit high sensitivity to the chosen value of k. The optimal k value depends on both the specific dataset and the downstream task, making it challenging to ascertain the ideal value before model training. To resolve this issue, we develop an adaptive-k component.

Firstly, we consider a set of candidate values for k with an upper bound K,

$$\mathcal{V}_k = \{0\} \cup \{k_i \in \mathbb{N} \mid \log_2 k_i \in \mathbb{N}, k_i \leq K\}. \tag{7}$$

Secondly, we present an adaptive-k network and allow the model to evaluate the implementation of distinct $k \in \mathcal{V}_k$ and select the optimal k value, generating the final prediction:

$$p(\hat{y} \mid x_q) = \sum_{k_i \in \mathcal{V}_k} p_{k_i}(\hat{y} \mid x_q) \cdot p_{\text{Ada}}(k_i), \tag{8}$$

where

$$p_{k_i}(\hat{y} \mid x_q) = f_\theta(\hat{\mathbf{x}}_{\mathbf{q},\mathbf{k_i}}), p_{\text{Ada}}(k_i) = \text{softmax}(f_\sigma(\hat{\mathbf{x}}_{\mathbf{q},\mathbf{k_i}})). \tag{9}$$

Here f_θ is implemented as a multi-layer perceptron (MLP) and $\hat{\mathbf{x}}_{\mathbf{q},\mathbf{k}}$ is derived by executing Eq. 6 on $\mathbb{S}_{q,k}$, which is constructed as follows:

$$\mathbb{S}_{q,k} = \{x_i | x_i \in \mathbb{S}_q, i \leq k\}, k \in \mathcal{V}_k. \tag{10}$$

Logit Adjustment Due to the imbalance of fraud datasets, we implied logit adjustment following (Menon et al. 2021). After the predictor f_θ takes in the aggregated embedding $\hat{\mathbf{x}}_\mathbf{q}$ of the target instance, a simple but effective way to handle class imbalance is adding an offset to the output logit $f_\theta(\hat{\mathbf{x}}_\mathbf{q})$. Correspondingly, we minimize the following loss function to train models within our TRE framework:

$$\begin{aligned}\mathcal{L} = -\{ &\sum_{(x_q, y) \in \mathbb{D}} y \log[p(\hat{y} = 1 \mid x_q) + \log(\pi_y)] \\ &+ (1 - y) \log[1 - p(\hat{y} = 0 \mid x_q) - \log(\pi_y)]\},\end{aligned} \tag{11}$$

where π_y is the imbalance ratio of $\mathbb{D}_{\text{train}}$.

A Lightweight Non-parametric Version We also present a lightweight, non-parametric variant of TRE that excludes both KNN search and embedding aggregation processes.

We use the SimHash (Sadowski and Levin 2007) algorithm, which involves defining M random directions, adding a perturbation in each direction to the input vector, and then obtaining a hash vector:

$$H(X_z) = \langle h_1(X_z), h_2(X_z), \ldots, h_M(X_z) \rangle. \tag{12}$$

When two sample embeddings' hash vectors share the same hash value, we assign them to the same hash bucket.

To expedite the retrieval process, rather than storing the entire hash table, we maintain a lightweight index containing the mean value of embeddings in each hash bucket. During testing, we update this mean value in the index based on changes in the vectors within the hash bucket. As a result, for a query embedding $\mathbf{x_q}$, we directly use the mean value of the hash bucket to which $\mathbf{x_q}$ belongs in the lightweight index as $\xi(\mathbf{x_q})$ and participate in the operation of Eq. 6.

Experiments

Datasets

We collected three industrial datasets from a mobile payment platform on the premise of complying with security and privacy policies, covering fraud detection, account takeover (ATO) detection, and money laundering detection tasks. The training data spans from 09/01/2022 to 09/30/2022, while the testing data spans from 02/01/2023 to 02/28/2023. We also selected data from six months after the training period as the out-of-time (OOT) testing dataset. It was made to validate the effectiveness of the TRE in scenarios where concept drift occurs naturally over time.

We follow the general definition of fraud detection scenarios, categorizing fraudulent transactions as positive samples and benign transactions as negative ones. For each transaction, we backtrack the user's behaviors in the last 24 hours prior to the target transaction. The maximum length of the behavior sequence is limited to 300, and we apply padding to behavior sequences shorter than this length. The statistical information of three datasets is shown in Table 1.

Note that our datasets do not contain detailed transaction information, e.g., user IDs and transaction types. Instead, we only rely on the behavior sequences to predict the fraudulent nature of the target transactions. Each behavior sequence consists of 300 discrete user actions arranged chronologically. A user action is defined as an event recorded when a user interacts with a clickable button within the app, with each button being associated with a unique positional identifier.

Task	Dataset	#Pos.	#Neg.	#Pos. Rate
Fraud	Train	151,322	1,764,768	7.90%
	Test	128,227	1,173,876	9.92%
ATO	Train	1,390	1,914,700	0.07%
	Test	809	1,291,321	0.06%
Money Laudering	Train	8,316	1,907,774	0.43%
	Test	6,696	1,285,434	0.52%

Table 1: The statistical information of the datasets.

Since the ATO and money laundry datasets are highly imbalanced, we conducted a 1:10 undersampling on the negative examples during training. Besides, we detail the goals of the three tasks as follows:

Fraud detection. The task of fraud detection refers to predicting the fraud probability of a user's behavior, which is crucial to the healthy development of the payment platform.

ATO detection. The task of account takeover (ATO) detection aims to identify the risk associated with unauthorized access to a user's account, which is vital for maintaining the security and trustworthiness of online platforms and protecting user information and assets.

Money laundering detection. The task of money laundering detection involves assessing the likelihood that a user's behavior involves illicitly obtained funds being used to legitimize those funds, which is a critical component in safeguarding the financial systems and preventing illegal financial transactions.

Experimental Settings

Evaluation Metrics Our experiments evaluated the performance of TRE on the fraud detection, ATO detection, and money laundering tasks using three commonly used metrics: **AUC**, **KS**, and **R@P**$_{0.1}$.

AUC is defined as the area under the ROC curve, which measures the model's ability to distinguish between classes. KS assesses the discrepancy between the cumulative distribution of positive and negative samples, which reflects the differentiation ability of models. R@P$_{0.1}$ is the recall score when the precision equals 10%, which indicates the ability to detect top-ranking positive samples. Higher AUC, R@P$_{0.1}$ and KS indicate the stronger risk discrimination ability of the model.

Compared Methods We compare TRE with 9 baselines which can be divided into two categories. The first category includes end-to-end sequential fraud detection models. **STAN** (Cheng et al. 2020) is a spatio-temporal attention-based neural network for credit card fraud detection. **NHMF** (Xi et al. 2020) is a two-level approach to capture hierarchical information in user event sequences for fraud detection. **SAH-RNN** (Lin et al. 2021) is a sequential model that incorporates web page structures into behavior sequences for fraud detection. **SHORING** (Li et al. 2022a) is a conditional high-order interaction network designed for fraud detection. The second category contains general sequential models. **Transformer** (Vaswani et al. 2017) is a widely used classic sequential model. **Longformer** (Beltagy, Peters, and Cohan 2020) is a variant of Transformer with global attention mechanisms and sliding windows. **Reformer** (Kitaev, Kaiser, and Levskaya 2020) is a Transformer variant with locality-sensitive hashing techniques. **RetNet** (Sun et al. 2023) is a recently proposed attention-based model. **TextCNN** (Kim 2014) is a CNN-based model with multiple one-dimensional convolutional layers of different kernel sizes.

Implementation Details Our TRE model is implemented using PyTorch (Paszke et al. 2019) in the Linux environment. We validate our training set by extracting 20% random samples. Early stopping is applied with a patience of 5 epochs. The embedding size and the batch size are set to 128 and 512, respectively. We perform the dropout at each layer as 0.1, and the total parameter count of the Retriever and the Predictor are 50.1K, as 1% of a Transformer encoder. AdamW (Loshchilov and Hutter 2018) is used as the optimizer with an initial learning rate of 1e-4.

Main Experiment

We compared TRE with various baselines on three datasets to demonstrate its superiority. The results are reported as the average over four runs with different random seeds: 2021, 2022, 2023, and 2024, as shown in Table 2. We have the following conclusions:

Datasets		Fraud Dataset			ATO Dataset			Money Laundering Dataset		
Methods	Metrics	AUC	KS	R@$P_{0.1}$	AUC	KS	R@$P_{0.1}$	AUC	KS	R@$P_{0.1}$
Fraud	STAN	0.7982	0.4601	0.5223	0.6512	0.2556	0.2454	0.8012	0.4723	0.4932
	NHFM	0.8110	0.4609	0.5292	0.6856	0.2812	0.3101	0.8489	0.5421	0.5612
	SAH-RNN	0.7961	0.4517	0.4960	0.6714	0.2557	0.2488	0.8502	0.5433	0.5811
	SHORING	0.8079	0.4710	0.5208	0.6885	0.2819	0.3108	<u>0.8540</u>	0.5487	0.5833
Sequential	Transformer	0.7992	0.4692	0.5482	0.6478	0.2376	0.2203	0.8175	0.4792	0.5183
	Transformer+TRE	0.8233	0.4966	**0.5702**	0.6699	0.2699	0.2662	0.8475	0.5337	0.5731
	Longformer	0.8111	0.4774	0.5519	0.6775	0.2612	0.2724	0.8283	0.5024	0.5152
	Longformer+TRE	<u>0.8253</u>	<u>0.5031</u>	0.5634	<u>0.6908</u>	<u>0.2892</u>	0.3010	0.8482	0.5396	0.5669
	Reformer	0.7945	0.4522	0.5226	0.6543	0.2505	0.2488	0.8018	0.4755	0.4909
	Reformer+TRE	0.8154	0.4803	0.5442	0.6794	0.2701	0.3172	0.8146	0.4810	0.5557
	RetNet	0.8034	0.4681	0.5443	0.6658	0.2555	0.2686	0.8283	0.5024	0.5152
	RetNet+TRE	0.8083	0.4682	0.5473	0.6722	0.2639	0.2923	0.8552	<u>0.5520</u>	<u>0.5868</u>
	TextCNN	0.8116	0.4801	0.5510	0.6720	0.2772	<u>0.3433</u>	0.8268	0.5031	0.5650
	TextCNN+TRE	**0.8276**	**0.5050**	<u>0.5687</u>	**0.7079**	**0.3289**	**0.3706**	**0.8624**	**0.5603**	**0.6140**

Table 2: Comparison of TRE with baselines on fraud detection, ATO detection, and money laundering detection tasks, measured by AUC, KS, and R@$P_{0.1}$ metrics. The bold scores represent the best statistically significant results, while the underlined ones indicate the second best.

- For all three downstream tasks, it is evident that our proposed method TRE significantly enhances performance (t-test with p-value <0.01) compared to all vanilla baselines.
- Compared to all the sequential models specifically designed for fraud detection scenarios, TRE enables general sequential models to achieve the highest performance across all datasets.
- This wide-range superiority suggests that TRE can learn useful information from unlabeled test data and thus improve the performance of downstream tasks.
- Among the compared baselines, TextCNN outperforms other attention-based methods. We interpret this as TextCNN's local invariance being more suitable for adapting to user behavior sequences compared to the global invariance of self-attention mechanisms. The focus of TextCNN on local continuity within sequences aligns more closely with the characteristics of user behavior sequences (Liu et al. 2020). Additionally, previous studies have highlighted that attention-based models struggle with sequence data affected by concept drift (Shao et al. 2023).

Comparison with Self-Supervised Methods

We compared TRE with self-supervised approaches under the TTT paradigm. Adjustments were made to adapt the existing methods, originally proposed for image datasets, to sequence datasets.

- **TTT+MAE** (Gandelsman et al. 2022): TTT+MAE is a domain adaptation method that trains an additional Mask AutoEncoder (MAE) on the test dataset. We applied it to our behavior sequence dataset by reproducing an MAE for sequential data, introducing noises by randomly masking user behavior nodes with Bernoulli probabilities.
- **TTT+DAE**: We also replace the MAE with a Denoising AutoEncoder (DAE) on our behavior sequence dataset, introducing noises by randomly substituting some user behavior nodes.

Methods	AUC	KS	R@$P_{0.1}$
TTT+MAE	0.8074	<u>0.4782</u>	<u>0.5532</u>
TTT+DAE	<u>0.8077</u>	0.4603	0.5380
Transformer+TRE	**0.8233**	**0.4966**	**0.5702**

Table 3: Comparison of TRE with self-supervised methods on the fraud detection task.

As Table 3 depicts, current TTT methods, initially proposed in the computer vision domain, demonstrate limited improvement when applied to user behavioral sequence data. One explanation for this is that these methods aim to enhance prediction performance by designing self-supervised tasks to develop more powerful encoders. However, in the context of fraud detection, the priority is to adapt to evolving fraudulent schemes rather than simply improving the encoder's strength.

A Lightweight Non-parametric Implementation

We provided a lightweight, non-parametric version of TRE and evaluated its performance in fraud detection tasks. Specifically, this lightweight TRE implementation relies on the randomness of SimHash (Sadowski and Levin 2007). Consequently, in the non-parametric experiments, we did not use a fixed seed but conducted four random runs to calculate the average scores.

As shown in Table 4, non-parametric TRE can increase efficiency by more than twofold compared to the parametric TRE, with an acceptable trade-off in performance enhancement. This lightweight, non-parametric version of TRE makes our solution for concept drift more feasible for practical deployment in anti-fraud systems.

Methods	AUC	KS	R@P$_{0.1}$	Epoch Time
Transformer	0.7992	0.4692	0.5482	**3.8719**
+TRE-Non	0.8193	0.4850	0.5513	6.7889
+TRE	**0.8233**	**0.4966**	**0.5702**	15.4670

Table 4: Comparison between the parametric and non-parametric versions of TRE on the fraud detection task. All results were derived using a V100-16GB GPU, with Epoch Time measured in GPU seconds.

Methods	AUC	KS	R@P$_{0.1}$
Transformer	0.7992	0.4692	0.5482
+TRE	**0.8233**	**0.4966**	**0.5702**
+TRE w/o K	0.8218	0.4869	0.5510
+TRE w/o W	0.8157	0.4805	0.5535
+TRE w/o K,W	0.8154	0.4857	0.5591
+TRE w/o T	0.8034	0.4680	0.5443

Table 5: Ablation study of TRE on the fraud detection task.

Ablation Study

To specifically validate the effectiveness of TRE, we conducted ablation studies to evaluate the contributions of each component. To ensure a fair comparison, we chose Transformer as the base encoder for all the variants of TRE:

- **TRE**: The proposed TRE.
- **TRE w/o K**: TRE without the adaptive-k. After extensive experimentation, we have determined that a fixed value of $k = 20$ serves as our chosen optimal hyperparameter.
- **TRE w/o W**: TRE without the gaussian kernel weight. Instead of employing a weighting strategy, we average the embeddings returned by the retrieval process.
- **TRE w/o K, W**: TRE without the above two components.
- **TRE w/o T**: TRE without the construction of the retrieval index with test data. We substituted the testing dataset with the training dataset to build the index.

Table 5 depicts that the complete TRE yields the highest performance increase compared to the baseline Transformer. Additionally, each variant, without one or two key components, still enhances fraud detection performance. This finding highlights the necessity of our full design, incorporating all the key components ablated in experiments.

Specifically, TRE w/o T exhibits a notable performance decrease compared to TRE, making it close to the baseline level. This suggests that the notable enhancement achieved by TRE can primarily be credited to its capability to accommodate the distribution of the test dataset. Moreover, our strategy of weighted embedding aggregation surpasses the conventional average aggregation. And the adaptive-k component for KNN is proven to be more effective than the manually optimized hyperparameter.

TRE against Data Concept Drift

Comparison over months To demonstrate the presence of concept drift over time and the mitigating effects of our model, we conducted experiments on five monthly test sets from 10/2022 to 02/2023. The results are shown in Figure 3.

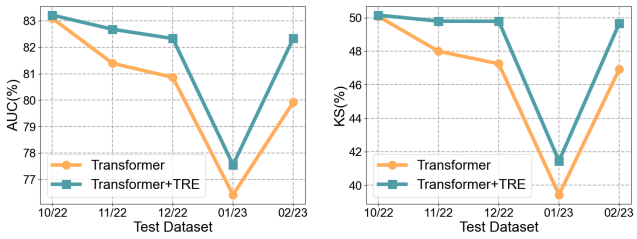


Figure 3: The performance of TRE on fraud detection task evaluated on different monthly testing datasets: 10/2022, 11/2022, 12/2022, 01/2023, and 02/2023.

Dataset	0%	5%	20%	50%	100%
Transformer	83.09	82.46	81.55	79.30	79.92
+TRE	83.21	83.35	82.60	82.10	82.33

Table 6: The models' AUC performance across test sets with increasing proportions of 02/2023 data, ranging from 0% (only 10/2022 data) to 100% (10/2022 data + 02/2023 data).

TRE effectively mitigates the performance decay over time, ensuring consistent performance. Besides, TRE appears to positively impact model performance in addressing unexpected declines, which may be linked to specific events, such as the Chinese New Year holiday in January 2023.

Comparison across proportions To further illustrate how concept drift influences fraud detection models' performance and how TRE addresses this issue, we progressively incorporated data from 02/2023 into the testing set, which initially consisted of data from 10/2022. The results, presented in Table 6, revealed a significant performance decline as the proportion of 02/2023 data increased. This finding supports our hypothesis that concept drifts between training and testing datasets intensify over time. Compared to the vanilla Transformer, incorporating TRE led to notable performance improvement on test sets with varying proportions of recent data, demonstrating the efficacy of our plug-in approach in mitigating this phenomenon.

Conclusion

In this paper, we concentrated on the online fraud detection scenario and proposed the TRE approach to deal with the issue of concept drift in anti-fraud machine learning systems. Considering nearest neighbor information during testing, we addressed the challenge of the model's performance decline when confronted with a shifted data distribution. The TRE approach employs a retriever to locate dense representations from neighboring samples and an aggregation strategy to provide up-to-date information to the predictor. We conducted extensive experiments on three industrial user behavior datasets to assess the performance of TRE across three downstream financial risk management tasks. Extensive experiments validate that TRE outperforms the previous state-of-the-art methods and demonstrates high adaptability.

Acknowledgments

The research work is supported by National Key R&D Plan No. 2022YFC3303303, the National Natural Science Foundation of China under Grant No. U2436209, 62476263, the Project of Youth Innovation Promotion Association CAS, Beijing Nova Program 20230484430, the Innovation Funding of ICT, CAS under Grant No. E461060.

References

Bartler, A.; Bühler, A.; Wiewel, F.; Döbler, M.; and Yang, B. 2022. Mt3: Meta test-time training for self-supervised test-time adaption. In *International Conference on Artificial Intelligence and Statistics*, 3080–3090. PMLR.

Beltagy, I.; Peters, M. E.; and Cohan, A. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Borgeaud, S.; Mensch, A.; Hoffmann, J.; Cai, T.; Rutherford, E.; Millican, K.; Van Den Driessche, G. B.; Lespiau, J.-B.; Damoc, B.; Clark, A.; et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, 2206–2240. PMLR.

Cheng, D.; Xiang, S.; Shang, C.; Zhang, Y.; Yang, F.; and Zhang, L. 2020. Spatio-temporal attention-based neural network for credit card fraud detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 362–369.

Gama, J.; Žliobaitė, I.; Bifet, A.; Pechenizkiy, M.; and Bouchachia, A. 2014. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4): 1–37.

Gandelsman, Y.; Sun, Y.; Chen, X.; and Efros, A. 2022. Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems*, 35: 29374–29385.

Hashemi, H.; Zamani, H.; and Croft, W. B. 2020. Guided transformer: Leveraging multiple external sources for representation learning in conversational search. In *Proceedings of the 43rd international acm sigir conference on research and development in information retrieval*, 1131–1140.

Hashemi, H.; Zamani, H.; and Croft, W. B. 2021. Learning Multiple Intent Representations for Search Queries. Association for Computing Machinery, New York, NY, USA, 669–679.

Hu, S.; Zhang, X.; Zhou, J.; Ji, S.; Yuan, J.; Li, Z.; Wang, Z.; Chen, Q.; He, Q.; and Fang, L. 2021. Turbo: Fraud detection in deposit-free leasing service via real-time behavior network mining. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 2583–2594. IEEE.

Johnson, J.; Douze, M.; and Jégou, H. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3): 535–547.

Kantchelian, A.; Afroz, S.; Huang, L.; Islam, A. C.; Miller, B.; Tschantz, M. C.; Greenstadt, R.; Joseph, A. D.; and Tygar, J. 2013. Approaches to adversarial drift. In *Proceedings of the 2013 ACM workshop on Artificial intelligence and security*, 99–110.

Kim, Y. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Kitaev, N.; Kaiser, Ł.; and Levskaya, A. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Kuo, W.; Angelova, A.; Lin, T.-Y.; and Dai, A. 2020. Mask2cad: 3d shape prediction by learning to segment and retrieve. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, 260–277. Springer.

Li, H.; Fu, X.; Wu, R.; Xu, J.; Xiao, K.; Chang, X.; Wang, W.; Chen, S.; Shi, L.; Xiong, T.; et al. 2022a. Design Domain Specific Neural Network via Symbolic Testing. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 3219–3229.

Li, Q.; He, Y.; Xu, C.; Wu, F.; Gao, J.; and Li, Z. 2022b. Dual-Augment Graph Neural Network for Fraud Detection. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 4188–4192.

Liang, J.; Hu, D.; and Feng, J. 2020. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International conference on machine learning*, 6028–6039. PMLR.

Lin, W.; Sun, L.; Zhong, Q.; Liu, C.; Feng, J.; Ao, X.; and Yang, H. 2021. Online credit payment fraud detection via structure-aware hierarchical recurrent neural network. In *IJCAI*, 3670–3676.

Liu, C.; Zhong, Q.; Ao, X.; Sun, L.; Lin, W.; Feng, J.; He, Q.; and Tang, J. 2020. Fraud Transactions Detection via Behavior Tree with Local Intention Calibration. In *KDD*, 3035–3043.

Liu, Y.; Ao, X.; Qin, Z.; Chi, J.; Feng, J.; Yang, H.; and He, Q. 2021a. Pick and choose: a GNN-based imbalanced learning approach for fraud detection. In *Proceedings of the web conference 2021*, 3168–3177.

Liu, Y.; Kothari, P.; Van Delft, B.; Bellot-Gurlet, B.; Mordan, T.; and Alahi, A. 2021b. Ttt++: When does self-supervised test-time training fail or thrive? *Advances in Neural Information Processing Systems*, 34: 21808–21820.

Loshchilov, I.; and Hutter, F. 2018. Fixing weight decay regularization in adam.

Lu, J.; Liu, A.; Dong, F.; Gu, F.; Gama, J.; and Zhang, G. 2019. Learning under Concept Drift: A Review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12): 2346–2363.

Menon, A. K.; Jayasumana, S.; Rawat, A. S.; Jain, H.; Veit, A.; and Kumar, S. 2021. Long-tail learning via logit adjustment. In *International Conference on Learning Representations*.

Niu, S.; Wu, J.; Zhang, Y.; Chen, Y.; Zheng, S.; Zhao, P.; and Tan, M. 2022. Efficient test-time model adaptation without forgetting. In *International conference on machine learning*, 16888–16905. PMLR.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *NIPS*.

Pendlebury, F.; Pierazzi, F.; Jordaney, R.; Kinder, J.; and Cavallaro, L. 2019. {TESSERACT}: Eliminating experimental bias in malware classification across space and time. In *28th USENIX Security Symposium (USENIX Security 19)*, 729–746.

Prabhudesai, M.; Goyal, A.; Paul, S.; Van Steenkiste, S.; Sajjadi, M. S.; Aggarwal, G.; Kipf, T.; Pathak, D.; and Fragkiadaki, K. 2023. Test-time adaptation with slot-centric models. In *International Conference on Machine Learning*, 28151–28166. PMLR.

Sadowski, C.; and Levin, G. 2007. Simhash: Hash-based similarity detection.

Shao, Z.; Wang, F.; Xu, Y.; Wei, W.; Yu, C.; Zhang, Z.; Yao, D.; Jin, G.; Cao, X.; Cong, G.; et al. 2023. Exploring Progress in Multivariate Time Series Forecasting: Comprehensive Benchmarking and Heterogeneity Analysis. *arXiv preprint arXiv:2310.06119*.

Soemers, D.; Brys, T.; Driessens, K.; Winands, M.; and Nowé, A. 2018. Adapting to concept drift in credit card transaction data streams using contextual bandits and decision trees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Sun, Y.; Dong, L.; Huang, S.; Ma, S.; Xia, Y.; Xue, J.; Wang, J.; and Wei, F. 2023. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*.

Sun, Y.; Wang, X.; Liu, Z.; Miller, J.; Efros, A.; and Hardt, M. 2020. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, 9229–9248. PMLR.

Tounekti, O.; Ruiz-Martinez, A.; and Gómez, A. F. S. 2019. Users supporting multiple (mobile) electronic payment systems in online purchases: An empirical study of their payment transaction preferences. *IEEE Access*, 8: 735–766.

Tula, D.; Paul, S.; Madan, G.; Garst, P.; Ingle, R.; and Aggarwal, G. 2023. Is it an i or an l: Test-time Adaptation of Text Line Recognition Models. *arXiv preprint arXiv:2308.15037*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, D.; Shelhamer, E.; Liu, S.; Olshausen, B.; and Darrell, T. 2020. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*.

Wang, Z.; Luo, Y.; Zheng, L.; Chen, Z.; Wang, S.; and Huang, Z. 2023. In Search of Lost Online Test-time Adaptation: A Survey. *arXiv preprint arXiv:2310.20199*.

Xi, D.; Zhuang, F.; Song, B.; Zhu, Y.; Chen, S.; Hong, D.; Chen, T.; Gu, X.; and He, Q. 2020. Neural hierarchical factorization machines for user's event sequence analysis. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, 1893–1896.

Xu, R.; Guo, M.; Wang, J.; Li, X.; Zhou, B.; and Loy, C. C. 2021. Texture memory-augmented deep patch-based image inpainting. *IEEE Transactions on Image Processing*, 30: 9112–9124.

Zamani, H.; Diaz, F.; Dehghani, M.; Metzler, D.; and Bendersky, M. 2022. Retrieval-enhanced machine learning. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2875–2886.

Zhang, G.; Li, Z.; Huang, J.; Wu, J.; Zhou, C.; Yang, J.; and Gao, J. 2022. efraudcom: An e-commerce fraud detection system via competitive graph neural networks. *ACM Transactions on Information Systems (TOIS)*, 40(3): 1–29.