applied sciences

MDPI

Article

A Contrastive Learning Framework for Detecting Anomalous Behavior in Commodity Trading Platforms

Yihao Li and Ping Yi *

School of Cyber Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China; liyihao321@sjtu.edu.cn
* Correspondence: yiping@sjtu.edu.cn

Featured Application: The work can be applied to commodities, equities, e-commerce, and social networking platforms to detect anomalies in each user's account to provide timely notification and thus reduce losses.

Abstract: For bulk commodity, stock, and e-commerce platforms, it is necessary to detect anomalous behavior for the security of users and platforms. Anomaly-detection methods currently used on these platforms train a model for each user since different users have different habits. However, the model cannot be trained adequately due to insufficient individual user behavior data. In this study, to utilize information between users and avoid underfitting, we propose a contrastive learning framework to train a complete global model (GM) for anomaly detection in a trading platform. By confusing the data between different users to generate negative samples, the model can learn the differences between users by contrastive learning. To reduce the need for individual user behavior data, this framework uses a GM instead of a model for each user to learn similarities between users. Experiments on four datasets show that models trained using our framework achieve better area-under-the-curve (AUC) scores than do the original models, proving that contrastive learning and GM are useful for anomaly detection in trading platforms.

Keywords: anomalous behavior detection; contrastive learning; trading platforms

check for updates

Citation: Li, Y.; Yi, P. A Contrastive Learning Framework for Detecting Anomalous Behavior in Commodity Trading Platforms. *Appl. Sci.* **2023**, *13*, 5709. https://doi.org/10.3390/app13095709

Academic Editor: Emilio Soria-Olivas

Received: 15 April 2023
Revised: 3 May 2023
Accepted: 4 May 2023
Published: 5 May 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The Internet is inextricably linked to information and property security. On trading platforms such as those for bulk commodities, stocks, and e-commerce, a misoperation by a user can cause serious property security issues. For example, a shareholder of Shenzhen Changfang Company mistakenly sold 16,000 shares by entering the wrong stock code, and an operator of a manufacturing giant, Sany Heavy Industry, mistakenly sold nearly 100,000 shares (https://www.cnbc.com/2020/09/09/more-chinese-companies-blaming-trading-typos-for-insider-stock-sales.html, accessed on 10 April 2023). These events suggest that to protect the interests of users, trading platforms should introduce anomaly-detection algorithms to determine anomalous user behavior and notify users on time. However, in trading platforms, the habits of different users vary widely, and behavior that is normal for one user may be anomalous for another [1]. Therefore, separate custom models for anomaly detection are needed for different users, and generic algorithms similar to intrusion detection cannot be used to detect user anomalies. The data on trading platforms are also unlabeled, which poses a challenge for anomaly-detection.

Cheng et al. focused on the anomalous behavior of individual users in a social network [2]. They first extracted and normalized node network features and then built a node evolution model based on the extracted node features. Finally, they used an anomaly-detection method for a multivariate time series to detect possible anomalies. Gupta et al. at eBay also focused on the differences in detecting anomalous behavior between users.

They used density estimation or principal component analysis (PCA) to generate profiles per system user and predict user activities based on their prior profiles [3]. Both studies used anomaly-detection algorithms designed for a single multivariate time series when detecting user behavior anomalies. When existing unsupervised methods are used for detection, a model needs to be trained for each user with only a small amount of data; thus, the model cannot be trained adequately. In this study, we aimed to solve the unsupervised training and the underfitting problems.

The methods used by Cheng et al. and Gupta et al. aim to detect outlier points in a time series, a system, or some datasets, such as the KDDCUP99 dataset [4,5]. These methods only focus on anomalies in the entire dataset or system, such as price manipulation and deception; we assume that they have only "one user" of the data. NASA used long short-term memory (LSTM) algorithms to detect radiation, power, instrumentation, and computational activities for spacecraft to ensure their proper functioning [6]. Audibert et al. used autoencoders (AEs) to overcome the difficulties of the absence of standard data in detecting anomalies in IT systems [7]. Yamauchi et al. detected user behavior to determine the occurrence of attacks against IoT devices [8]. Anomaly-detection methods for multi-user data have been designed to detect anomalous users. Jiang et al. used graph convolutional networks to detect malicious threat user groups by detecting malicious behaviors [9]. Kim et al. also used user behavior modeling along with density estimation, PCA, and clustering to detect insider threats [10]. Tang et al. used graph spectra for anomaly-detection [11].

1.1. Motivation for Contrastive Learning

There are many users on trading platforms and as mentioned above, different users have different behavioral habits, referred to as "differences between users" in this paper.

However, the existing methods for detecting anomalous user behavior on trading platforms work independently for each user, do not consider differences between users, and therefore cannot train models adequately. To learn the global characteristics of the user population from unlabeled data, we use a contrastive learning method to generate labels and provide the loss function for model training.

By replacing the last action of a normal behavior sequence with the action of other users, anomalous sequences can be generated and fed into sequence models, as in Figure 1. Therefore, being fed with normal and anomalous sequences, sequence models can learn from the contrast between the last and former actions and be trained more efficiently on unlabeled trading platform data.

1.2. Motivation for the Global Model (GM)

Existing anomaly-detection methods used in trading platforms consider users as isolated individuals, which causes model underfitting and is inconsistent with the human judgment process.

In Table 1, the average number of user actions in some trading platforms is only a few dozen. That is, when training a model for each user, the amount of data that can be used for training is insufficient and very likely to cause model underfitting. More data are needed to train the models.

Table 1. The characteristics of datasets after preprocessing.

Dataset	Users	Behavior Sequence Length (Averaged)	Dimensions		
			Decimal	One-Hot	Embedding
Commodity	562	27,366	11	2	1
Eshop	1624	32	1	33	2 + 3
Yoochoose	68,790	15	0	12	2
Taobao	969,529	103	0	4	5

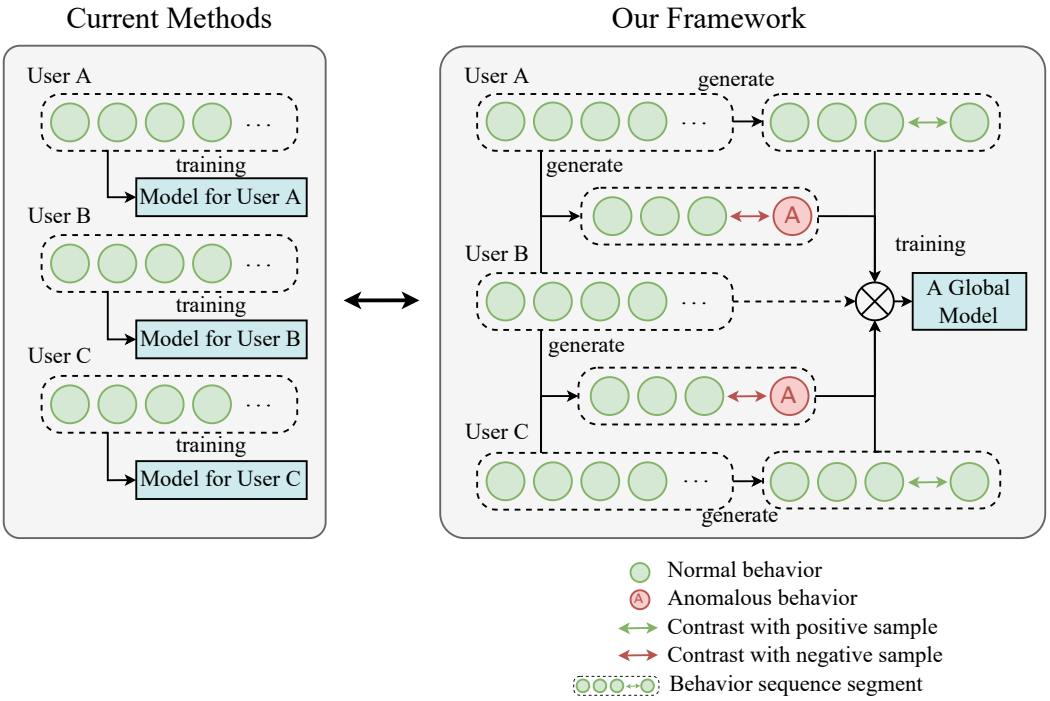


Figure 1. Our contrastive learning framework trains a neural network model with mutual supervision on trading platforms. To perform contrastive learning, our framework generates negative samples by replacing the last action in the user behavior sequence with the action of other users to obtain anomalous behavior sequences. To make better use of similarities between users, our framework trains a global model instead of a model for each user.

Although users have different behavioral habits on trading platforms such as those for commodities, stocks, and e-commerce, they also have certain similar behavioral habits. For example, while users may have their investment preferences when making purchases, they avoid incurring losses in their commodity or stock trading. Meanwhile, digital enthusiasts may prefer to buy computer accessories, while stock speculators may choose to do many trading operations in a short period, and so on.

Therefore, when performing anomaly detection, judging individual users based only on their behavior without considering the behavior of similar users is insufficient. That is, making correct judgments requires similarities between users. A model can learn common sense only when it is used to recognize different users. We train the model using behavior data from all users to learn similarities, rather than building a model that incorporates prior knowledge. In addition, since the model is trained in advance, using similarities between users can also reduce the need for individual user data during detection.

Therefore, we propose a new contrastive learning framework to detect anomalous behavior on trading platforms using neural network sequence models. And our motivations are shown in Figure 2. The main contributions are as follows:

- Differences between users are utilized to perform supervised training on unlabeled trading platform data using contrastive learning.
- To use similarities between users and avoid underfitting, we use GM to perform anomaly detection for all users instead of a separate model for each user, which can reduce complexity and increase efficiency.
- Our models were tested on four trading platform datasets and achieved the best results on three of the datasets, with AUC scores of 0.85+. This demonstrated that the behavioral habits of users on these trading platforms are different, which is the basis of our framework.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 proposes the design and modification for sequence models. Section 4 presents the experiment. Section 5 presents the conclusions of the study.

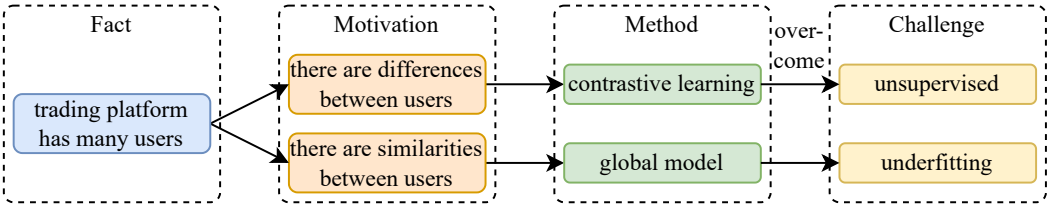


Figure 2. We can train models using multi-user records on trading platforms. Differences and similarities between users can assist in the training of models.

2. Related Works

On trading platforms, the existing anomalous-user-action-detection methods detect anomalies for multivariate time series. We divide anomaly-detection methods into traditional anomaly-detection methods and model-based anomaly-detection methods according to whether the methods use artificial neural network models.

Traditional anomaly-detection methods usually distinguish outliers based on computable features, and most focus on the presence of outliers in the data. The local outlier factor (LOF) method was proposed in 2000 [12]. It calculates the density of points around each point and compares the density to determine whether a point is an anomaly. Many anomaly-detection methods are based on other computable features [13–18]. Li et al. developed unsupervised outlier detection using the empirical cumulative distribution functions (ECOD) method, which detects anomalous behavior based on the distribution of user behavior by using the evidence that anomalous behavior is sparse in the distribution [19]. We compare these methods in Section 4.

Model-based methods detect anomalies by computing the distance between their estimated or predicted actions and the original actions [20]. Malhotra et al. used an LSTM autoencoder network to reconstruct behavior sequences and then detect anomalies by comparing the distance between the two sequences [21]. Sharma et al. used a similar network to detect malicious users [22]. Kieu et al. proposed a model consisting of an embedding method and a recurrent neural network (RNN) model [23]. They designed a variety of metrics within and between sliding windows and used metric sequences instead of behavior sequences for reconstruction. Munir et al. proposed a deep learning-based anomaly detection approach (DeepAnT), which uses a sliding window method to convert time series into fixed-length intervals [24]. They then used a CNN to extract features from the intervals and used a fully connected network to predict user action at the next moment. DeepAnT reduces training time and complexity while maintaining the performance of RNNs. Wen et al. also used CNNs instead of RNNs based on time series segmentation [25].

However, both traditional and neural network approaches require training a model for each user when used for anomalous action detection on trading platforms. Without optimizing for the trading platform, these models are trained in an unsupervised manner using only the behavior data of one user, which fails to optimally use the available data.

3. Methodology

3.1. Problem Definition

The problem this study aimed to solve is the detection of anomalous trading behavior using unlabeled trading platform data. There are n users on a platform, and user i has a behavior sequence $X_i = \{x_{i1}, x_{i2}, \cdots, x_{il_i}\}$ of length l_i. For an anomaly-detection method

M, Equation (1) should be satisfied, where a_{ij} is an unknown action, 0 denotes normal, and 1 denotes anomalous.

$$M\left(\left\{x_{i1}, x_{i2}, \cdots, x_{i(j-1)}\right\}, a_{ij}\right) = \begin{cases} 0 & a_{ij} \text{ is normal} \\ 1 & a_{ij} \text{ is anomalous} \end{cases} \tag{1}$$

3.2. Previous Methods

To detect the anomalous behavior of a user i, the previous methods described need to train a separate model M_i on the behavior sequence $\{x_{i1}, x_{i2}, \cdots, x_{i(j-1)}\}$. The model M_i is then used to perform anomaly detection for action i for the target Equation (2). However, a short user behavior sequence may cause model underfitting.

Existing sequence models are designed for unsupervised training, so they need to estimate or predict user behavior and calculate the deviation between the obtained behavior and the actual behavior for backpropagation. For evaluation, the larger the bias is, the more likely the behavior is anomalous. For example, RNN-AE uses the RNN decoder to reconstruct user behavior to estimate it, and DeepAnT uses a fully connected network to predict user behavior at the next moment [21,24].

$$M\left(\left\{x_{i1}, x_{i2}, \cdots, x_{i(j-1)}\right\}, a_{ij}\right) = M_i(a_{ij}), M_i \text{ trained on } \{x_{i1}, x_{i2}, \cdots, x_{i(j-1)}\} \tag{2}$$

3.3. Contrastive Learning Loss

There are many users on trading platforms. If differences between users can be used to train models, the effect should be much better than that of using the behavior data of only one user.

Instead of estimating or predicting user behavior to obtain a loss function, we propose to train the model using a contrastive learning approach to enable the model to learn to make judgments on positive and negative data. For the sequence model, a continuous segment of user i's behavior sequence is a positive segment. The action $x_{i'j'}, i' \neq i$ is likely to be an anomaly for this user i, while user i' is randomly selected. Thus, for each positive segment, a negative segment can be obtained by replacing the last action x_{ij} with the action $x_{i'j'}$ of any other user i' where the action $x_{i'j'}$ is anomalous.

Contrastive learning is mainly used in computer vision and natural language processing to maximize the distance between different labels in latent space and to minimize the distance between the same labels [26]. When we apply it to detect anomalous behavior, we want to maximize the distance in latent space between the last and the former behaviors in the negative segment and minimize the distance in the positive segment. With the model classifying input normal segments as 0 and anomalous segments as 1, we have the contrastive loss Equation (3). $L_{CrossEntropy}$ is cross-entropy loss. We can instantly obtain a nearly infinite amount of negative samples with contrastive learning. By training with both positive and negative samples on top of this, contrastive learning is naturally more effective.

$$\begin{aligned} L_1 &= L_{CrossEntropy}(M_i(\{x_{i1}, x_{i2}, \cdots, x_{i(j-1)}\}, x_{i'j'}), 1) \\ L_2 &= L_{CrossEntropy}(M_i(\{x_{i1}, x_{i2}, \cdots, x_{i(j-1)}\}, x_{ij}), 0) \\ L &= L_1 + L_2 \end{aligned} \tag{3}$$

3.4. Global Model

Users also have some similarities. To take advantage of these similarities and avoid underfitting, instead of training a separate model for each user, we can train a global model M using behavior sequences of all users so that the model will judge the action a_{ij} directly without retraining. In addition, common sense can be shared among multiple users because we use the same model to recognize different users.

Because earlier behaviors have less impact on subsequent user behaviors, we feed the model the most recent m actions. If the sequence model is based on an RNN network, we can use it to detect a long behavior sequence, but it is still impossible to train the model on the entire sequence.

$$\begin{aligned} M\Big(\{x_{i(j-m+1)}, x_{i(j-m+2)}, \cdots, x_{i(j-1)}\}, x_{ij}\Big) &= 0 \\ M\Big(\{x_{i(j-m+1)}, x_{i(j-m+2)}, \cdots, x_{i(j-1)}\}, x_{i'j'}\Big) &= 1, i' \neq i \end{aligned} \tag{4}$$

Therefore, we can modify any sequence model to detect anomalous trading behavior for any user on unlabeled data.

- To use differences between users, we propose a new contrastive learning loss (CLL) method to train sequence models with supervision on unlabeled trading platform data. The method takes the actions of the detected user as normal data and the actions of other users as anomalous data to transform unsupervised training into supervised training, which improves the training efficiency.
- To use similarities between users and avoid underfitting, we use GM to handle multiuser anomalous action detection problems instead of training a separate model for each user on the platform. The model uses historical behavior to identify users and common sense learned from others to make a comprehensive judgment.

3.5. Model Modifications for Contrastive Learning

To be trained with the contrastive learning framework, the structure of the existing model needs to be modified to use CLL. We modified two representative models, the RNN-AE model proposed by Sharma et al. [21], which we call the RNN model in this paper, and the DeepAnT model proposed by Munir et al. [24]. The former is an estimation model, and the latter is a prediction model as defined by Blázquez-García [20]. The modifications of these two models are shown in Figure 3.

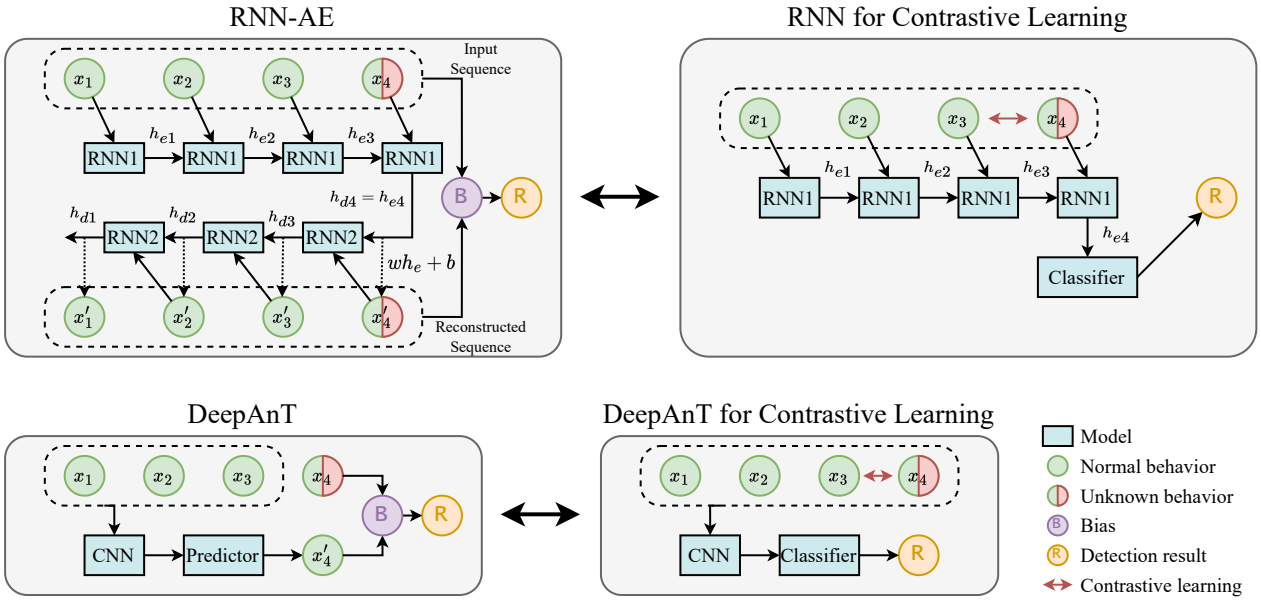


Figure 3. Modifications for RNN-AE and DeepAnT to be trained contrastively. There are two types of RNN sequences in the original RNN-AE, but only one after the modification. h_e and h_d are the hidden states of the RNN of the encoder and the decoder, respectively.

The RNN-AE model uses two RNN sequences as the encoding and decoding parts of the autoencoder. This model inputs the real behavior sequences and outputs the reconstructed behavior sequences in reverse order. Since the data are unlabeled, an autoencoder

is a common scheme for training on unlabeled data. Therefore Sharma et al. proposed a model based on RNN using the loss function of reconstruction. Thus, during training, the loss function makes the output and input sequences as close as possible. During detection, the RNN-AE model calculates the difference between the reconstructed and original sequences. The greater the difference between the two sequences is, the higher the probability that the sequence contains anomalous behavior. The architecture of the autoencoder adopted by the model is designed to introduce the reconstruction loss function of the autoencoder. While contrastive learning is another method to provide loss functions on unlabeled data, our RNN model can be trained properly without the reconstruction loss function. Therefore, we simplified the RNN-AE model by removing the decoding part of the model and reducing it to a simple RNN sequence. In our modified RNN model, the last RNN unit is followed by a classifier to determine whether the last action in the input behavior sequence is anomalous. When training our model, the positive and negative samples generated by contrast learning are input separately, and the model can be trained according to the CLL function Equation (3).

Unlike the RNN-AE model that provides the loss function through an autoencoder, DeepAnT uses a prediction loss function. With the input of previous behavior sequences, the model uses a CNN to extract temporal features from the user behavior sequence and predict the behavior at the next moment. The loss function of DeepAnT calculates the distance between the actual and the predicted behavior. This is also a common loss function on unlabeled data. When using our comparison learning loss function instead, it is necessary to change the output of DeepAnT into an anomaly classifier and adjust the input of DeepAnT to include the current action. Subsequently, the modified DeepAnT model can be trained in a similar manner to our modified RNN model.

Training a neural network requires backpropagation for a particular value and calculating a loss function. Since the data are unlabeled, existing models use various loss functions to train on it. Contrastive learning provides one of these loss functions, which is not different from the other methods in this respect.

3.6. Training

We describe the entire contrastive learning framework. First, the model is modified to use our proposed CLL instead of the original loss of the model. Second, the model is trained using GM instead of a model for each user as in Figure 4.

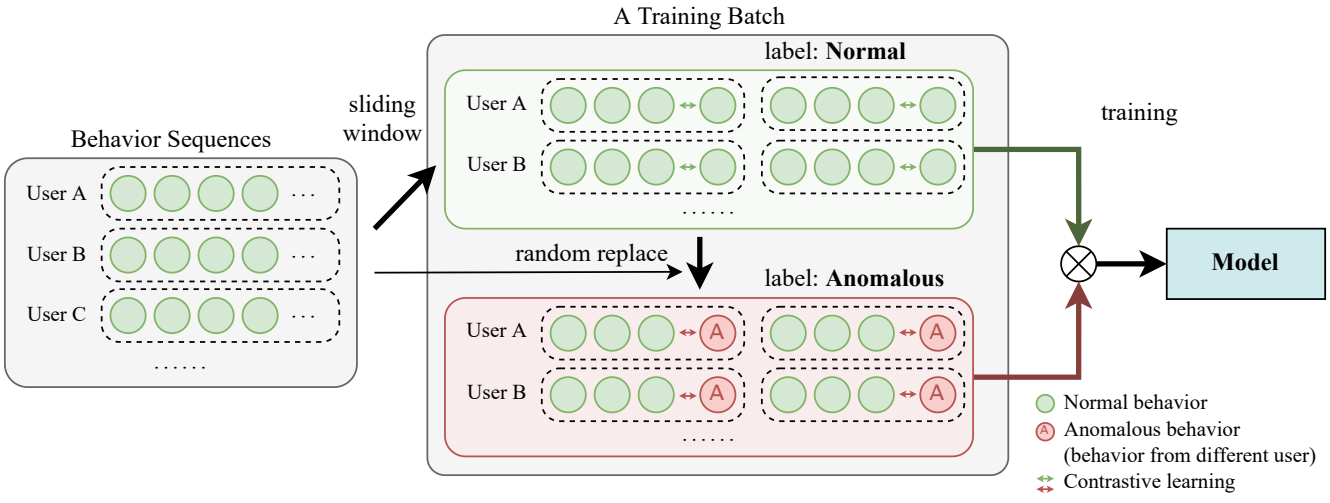


Figure 4. Anomalous sequences are generated by replacing the last action of normal sequences for contrastive learning to train GM.

In the training set, the sliding window method generates sequence segments of the behavior sequence of each user. The training set contains different users and is not dif-

ferentiated during training, as in Figure 5. A batch may contain sequence segments from different users. During training, these normal segments are input in each batch to minimize the distance in latent space between the last and the former actions. In addition, anomalous segments are generated on the spot based on normal segments to maximize the distance between the last and the former actions. A CLL L is calculated based on L_1 and L_2 using Equation (3) and then backpropagated to train the model. The last actions of normal sequences are randomly replaced by the actions of other users to obtain anomalous sequences.

The same process used for training can be used for testing and using the model. Taking contrastive-learning-modified DeepAnT as an example, to determine whether the action a_ij of user i is anomalous, we need to find the previous actions first and assemble them into a behavior sequence $\{x_{i1}, x_{i2}, \cdots, x_{i(j-1)}, a_{ij}\}$. The sequence is fed into the model for the determination of anomalies.

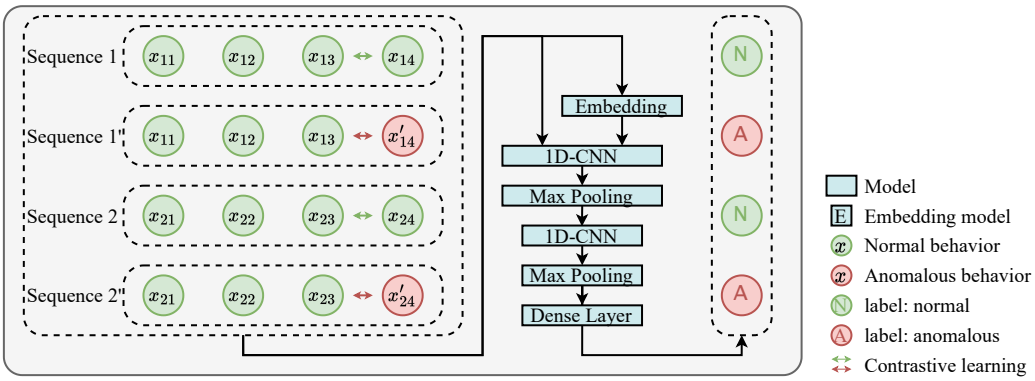


Figure 5. The training and testing methods of contrastive learning framework-based DeepAnT model.

The contrastive-learning-modified RNN model can follow the same process as that of DeepAnT. However, since the RNN model does not require a fixed input length as do CNNs, the connection between two RNN units is looser with the hidden state being used for information transfer. Therefore, in using an RNN for anomaly-detection, its hidden state can be stored in the database and retrieved from the database after the user action occurs. Then, the generated hidden state is stored in the database after the detection. The test process is shown in Figure 6. As the input data size is much smaller, the overhead of using RNNs in this manner is negligible.

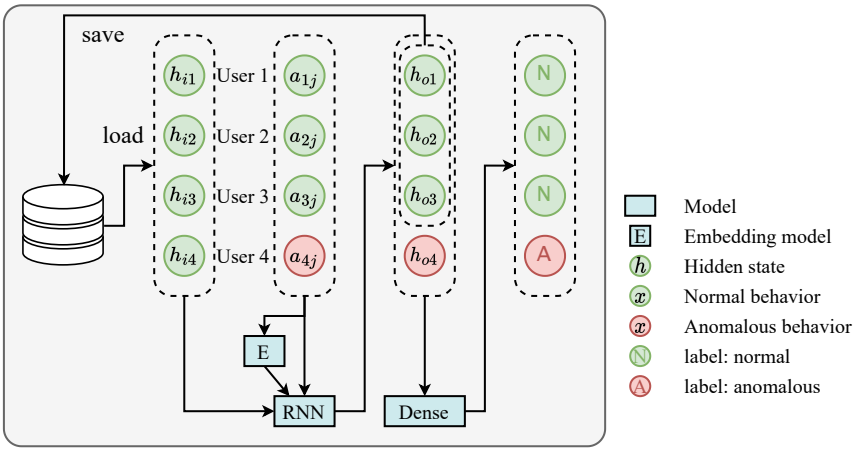


Figure 6. The training and testing methods for contrastive learning framework-based RNN models using an external database.

Because the model is trained on the corresponding normal and anomalous sequences in the same batch for contrast, as explained in Section 3.3, if the model determines that the

last action does not match the previous sequences after training, that action is considered an anomalous action.

Models trained by our framework learn the differences between different users to distinguish them with specificity. Because the same model is used during training, it also contains similarities with other time series. The contrastive learning framework enables the model to learn user information on trading platforms more efficiently.

4. Experiment

We performed experiments on four datasets to validate the proposed contrastive learning framework. Since the datasets do not have anomalies, we generated the anomalous actions. First, we tested the performance of the anomaly-detection methods and trained some of them using our framework. Ablation studies proved that contrastive learning among multiple users is effective.

4.1. Dataset and Preprocessing

For the experiment, we used a private bulk commodity dataset, (Commodity), Eshop dataset [27], Yoochoose dataset (https://recsys.acm.org/recsys15/challenge, accessed on 6 August 2022), and Taobao dataset (https://tianchi.aliyun.com/dataset/649, accessed on 6 August 2022) [28–30]. These datasets are from different platforms, each with unlabeled records of many users.

The Commodity dataset was provided by a partner company and included bulk commodity user transaction data for users from 2014 to 2015. We added the ratio of the historical price to the current price to the data. The Eshop dataset comprises click stream data of an online shopping website. We treated each connection as a user. The Yoochoose dataset is also a visit dataset from a shopping website. We only kept records with the main categories between 1 and 12. The Taobao dataset contains user behavior data, including click, star, buy, etc., from Taobao, a Chinese e-commerce platform.

The characteristics of the datasets after preprocessing are shown in Table 1. The detailed dimensions of these datasets are shown in Table 2. We handled enumerated data using one-hot codes whenever possible and embedded only data with a large number of data types. Numeric-type data were normalized to between 0 and 1 and concatenated with the one-hot code data. The above two types of data were processed before the model was trained so that all models in the experiment could use them. Some data used as one-hot codes were also used as embedded data. Because processing embedding-type data requires a specific neural network task, only the neural network models used the embedding-type data.

Table 2. The dimension of datasets after preprocessing.

Dataset	Dimension Type	Dimension	Meaning	Range/Type
Commodity	Decimal	1	Trade hour	[0, 24)
	Decimal	1	Trade number	[1, 46,191]
	Decimal	9	History price ratio	[0.494, 2.029]
	One-hot	2	Trade direction	2 types
	One-hot	2	Opening and liquidation	2 types
	Embedding	1	Commodity code	6 types
Eshop	Decimal	1	Price	[18, 82]
	One-hot	4	Main category	4 types
	One-hot	14	Color	14 types
	One-hot	6	Location	6 types
	One-hot	2	Model Photography	2 types
	One-hot	2	Price 2	2 types
	One-hot	5	Page	5 types
	Embedding	2	Country	47 types
	Embedding	3	Clothing model	217 types

Table 2. *Cont.*

Dataset	Dimension Type	Dimension	Meaning	Range / Type
Yoochoose	One-hot	12	Category	12 types
	Embedding	2	Category	12 types
Taobao	One-hot	4	Action type	4 types
	Embedding	5	Category	9438 types

4.2. Models and Experimental Settings

In the experiment, we compared several models that are used for anomaly detection in trading platforms. These models are from the PYOD library [31]: angle-base outlier detection (ABOD), cluster-based local outlier factor (CBLOF), unsupervised outlier detection using empirical cumulative distribution functions (ECOD), isolation forest (iForest), lightweight online detector of anomalies (LODA), local outlier factor (LOF), one-class support vector machine (OCSVM), and principal component analysis (PCA). We implemented the remaining neural network models using PyTorch [32] and added the embedding layer to handle enumeration types. These were the deep autoencoder (DAE), DeepAnT, recurrent neural network autoencoder (RNN-AE), our framework DeepAnT, and our framework RNN. The DAE model is a simple deep autoencoder trained per user and uses reconstruction bias to judge anomalous behavior.

On each trading platforms, users were divided into training-users and test-users. We trained the models using different training and testing methods according to the limitations of each model.

Existing methods to detect user behavior anomalies on trading platforms use separate models for each user. Therefore, we trained an individual model for each test-user for all traditional anomaly-detection methods and the DAE model. Since the inputs of the traditional anomaly detection and the DAE model are individual user actions, the basic units of the training and test sets are user actions. The training sets are the first 2/3 behavior sequence actions of the corresponding users. The test sets consisted of two parts. The normal part was the actions of the last 1/3 of the behavior sequence of the corresponding users. The anomalous part was the actions of other arbitrarily selected users. The number of normal and anomalous actions in the test set was guaranteed to be the same.

Regarding the sequence model of DeepAnT and RNN-AE, the basic units of the training and test sets were user behavior sequence segments. We also trained an individual model for each test-user. The training set was generated by passing a sliding window over the first 2/3 of the behavior sequences of the corresponding test-users. The test set was divided into two parts. The normal part was generated by passing a sliding window over the last 1/3 of the behavior sequences of these users. The anomalous part was generated from the normal part by replacing the last action of the segment with the action of the other test-user. In the ablation study, we demonstrate another training and testing method. The number of normal and anomalous actions in the test set was guaranteed to be the same.

Contrastive learning frameworks take different training and testing schemes when training modified DeepAnT and RNN models. Our framework uses GMs, which do not require training a model for each user and can be trained directly on the records of training-users. The basic units of the training and test sets were user behavior sequence segments. The training set was divided into two parts. The normal part was generated from the behavior sequence of the training-user using a sliding window, and the anomalous part was generated from the normal part by replacing the last action of the segments with the action of the other training-user. The test set was generated in the same way as with the sequence models, such as DeepAnT and RNN-AE, except that the complete behavior sequence of the test-user was used instead of the last 1/3.

All models were trained and tested on a computer with an NVIDIA 1080Ti graphics card. Their runtime might have been affected by their specific implementation.

4.3. Experiment Results

The results of our experiments are shown in Table 3. Our modified DeepAnT and RNN models achieved high average AUC scores on the four datasets. On the Commodity and Yoochoose data, traditional models such as ECOD and PCA also performed well, achieving AUC scores of 0.8 or even 0.9. This indicates that the model can distinguish the behavior of a user from those of the other users. The results provide sufficient proof of the basis of this study: that different users have different behavioral habits.

Table 3. Average AUC score [1] of models.

Model	AI Model	Dataset			
		Commodity	**Eshop**	**Yoochoose**	**Taobao**
ABOD [14]		0.710	0.560	0.358	0.471
CBLOF [13]		0.721	0.582	0.716	0.509
DAE	✓	0.654	0.634	0.822	0.651
ECOD [19]		0.805	0.720	**0.936**	0.523
iForest [16]		0.662	0.550	0.543	0.495
LODA [18]		0.461	0.478	0.838	0.505
LOF [12]		0.634	0.561	0.807	0.505
OCSVM [15]		0.703	0.606	0.836	0.505
PCA [17]		0.802	0.588	0.729	0.487
DeepAnT [24]	✓	0.578	0.586	0.814	0.640
RNN-AE [21]	✓	0.659	0.604	0.827	0.609
Our Framework DeepAnT	✓	0.897	**0.919**	0.871	0.851
Our Framework RNN	✓	**0.907**	0.905	0.879	**0.878**

[1] The average AUC score was calculated as follows: the AUC score for detecting anomalous actions for each user was calculated and then averaged across users. The check mark indicates that the model is an artificial intelligence model. Bold indicates that the model performed best on that dataset.

Table 3 also shows that after modification, the performance of the sequence models greatly improved. The performances of the original DeepAnT and RNN-AE were not much different from those of other models. Nonetheless, the average AUC scores of the modified models were almost 0.9, which is much better than those of the original models. This shows that our contrastive learning framework is effective.

In the Taobao dataset, traditional models were unlikely to obtain high AUC scores because their information mainly consisted of enumerated and embedded data. In the Yoochoose dataset, the traditional model OCSVM performed very well, which shows that for this dataset, users were more inclined to engage in certain actions consistently, which also implies that most models could achieve good results on this data. The ECOD model performed well on most of the data because it used the distribution of user behavior for anomaly detection, proving that the method of anomaly detection based on user data characteristics is feasible.

4.4. Ablation Studies

To further evaluate the performance of our framework, we experimented with partially modified models. In the previous experiments, the original DeepAnT and RNN-AE models were trained separately for each test-user. Whereas DeepAnT used prediction loss while RNN-AE used reconstruction loss, we applied the improvements of the loss function and GM to the original model separately in the ablation experiments to demonstrate the effectiveness of the improvements in both aspects.

The model structure and loss function of the DeepAnT+CLL and RNN+CLL models were changed based on the original DeepAnT and RNN models in Section 3.5. The prediction/reconstruction loss functions of these models were modified to CLL. We kept their training and testing methods unchanged. They were still trained and tested separately on each test-user.

The DeepAnT+GM and RNN-AE+GM models were trained according to Section 3.6, and we used the same testing approach as with our framework models. Since they still used the prediction loss and the reconstruction loss, respectively, there was no need to generate comparative anomaly data for training. The models were trained on normal behavior records generated from training-users and then tested using test-user data. The test set was the same as the one for testing the model based on our framework, and the anomalous data were similarly inserted.

Figures 7 and 8 show the results. On the Commodity dataset, changing the loss function worked much better than introducing GM. This is because the GM was introduced to solve the problem of insufficient single-user behavior data. The number of users in the Commodity dataset was large; thus, the effect of introducing GM was not that significant. As a comparison, on the Eshop dataset, the amount of behavior data of each user was small, so introducing GM resulted in greater improvement compared to introducing CLL.

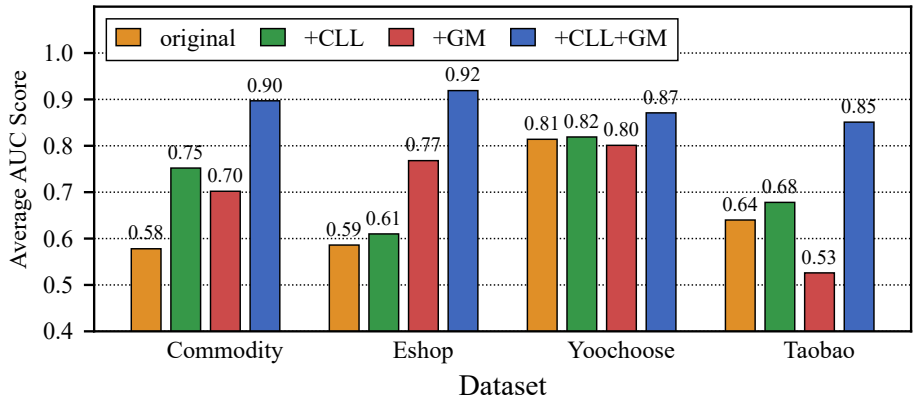


Figure 7. Average AUC scores of the original and partially modified DeepAnT models.

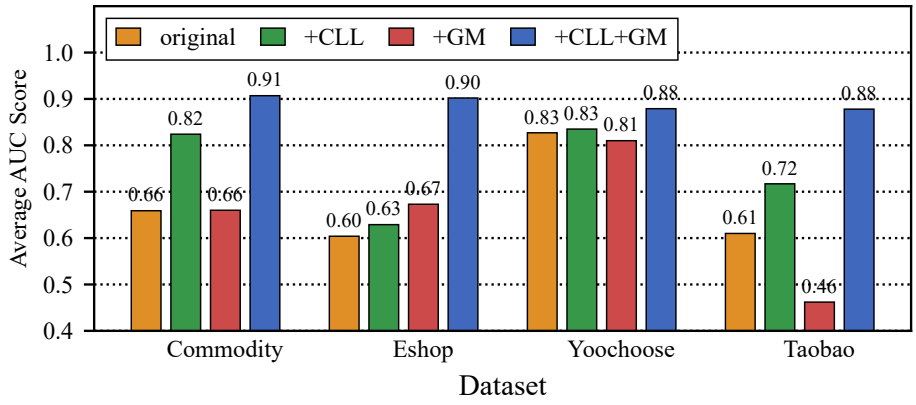


Figure 8. Average AUC scores of the original and partially modified RNN (RNN-AE) models.

On the Yoochoose dataset, because using the data of a single user is sufficient to determine the behavior of the user according to the ECOD model in Section 4.3, the introduction of GM had no effect. In addition, owing to the limited number of each user's behavior segments, it was impossible to train using a sufficient number of negative samples, so the introduction of CLL slightly improved the performance. Thereafter, the AUC score was improved when a large amount of contrastive learning was performed based on the GM.

On the Taobao dataset, the model performed worse after the addition of GM, while the addition of both GM and CLL achieved a great improvement. This indicates that GM alone cannot help the model extract features when the dataset feature extraction is difficult. The introduction of multiuser data in unsupervised training may interfere with

the model's judgment of anomalous actions. Therefore, when feature extraction is difficult, it is necessary to introduce CLL for supervision.

The ablation studies show the following results.

- The performance of the model can be greatly improved by exploiting the differences and similarities between users.
- The introduction of CLL can help the model to better extract features through supervision. The introduction of GM can alleviate the underfitting problem.
- Trading platforms can use GM or LLC according to their needs, but it is better to use both.

We also compared the time consumption of the original and framework-based models, and the results are shown in Table 4. Separate models were trained for 64 epochs with insufficient amounts of user data. The training epochs were reduced accordingly to reduce the training time. GM was trained on these datasets for 1, 16, 16, and 1 epoch(s), respectively. Because training a separate model for each user implies generating a large number of models, training individual models on a dataset with a large number of users and a small number of records per user, such as for the Yoochoose dataset, consumes much more time than is consumed for other datasets. GM also uses a larger batch during training. Therefore, it was hundreds of times faster on the Yoochoose dataset. Thus, GM has higher parallelism and lower time consumption than does the original model.

Using GM also reduces the difficulty of training. In experiments, a separate model needs a reduced number of parameters to solve the underfitting problem.

Table 4. Total time (min) of the DeepAnT model and partially modified DeepAnT models including training and testing time.

Model	Dataset			
	Commodity	**Eshop**	**Yoochoose**	**Taobao**
Original DeepAnT	92.2	2.2	282.0	6357.2
+CLL	67.4	2.7	352.3	7436.3
+GM	35.6	**0.3**	1.9	**238.4**
+CLL+GM	**28.4**	0.4	**0.6**	426.5

Bold indicates that the model performed best on that dataset.

5. Conclusions

There are differences and similarities between users. Anomalous behavior is also different for each user. This study focused on detecting the anomalous behavior of a single user on trading platforms and is the first to introduce interuser information in the detection of anomalous user action by contrasting learning.

To use differences between different users, CLL was designed for unlabeled data. With minor modifications, any neural network sequence model can learn the contrast information between the last and the former actions. To use similarities between users and overcome the underfitting problem, we designed the GM to detect all users, rather than a separate model for each user. The model can also distinguish and adapt the detection for different users. Through experiments, we demonstrated that CLL is more efficient than are other loss functions and that GM improves the effectiveness of the model and greatly reduces the training and testing times, which are indispensable. This suggests that similarities and differences between users should be considered when performing anomaly detection on trading platforms.

In addition, we experimentally illustrated that both traditional and neural network anomaly-detection methods can distinguish the behaviors of different users, proving that users have different behaviors.

For trading platforms, the security of user property is very important. Using anomaly-detection methods can better protect the security of user property. Considering interuser information, as demonstrated in this paper, can greatly improve the performance of these

methods on multiuser trading platforms. These anomaly-detection methods should be used as a loss recovery measure. When an anomalous action is detected, it means that user property may be at risk, and it is recommended that the trading platform notifies the user of the action by email, SMS, etc.

However, we only used one kind of negative sample in this study. We hope that based on this study, many contrastive learning anomaly-detection models with more kinds of negative samples can be designed for trading platforms to utilize the interuser information.

Author Contributions: Conceptualization, Y.L. and P.Y.; methodology, Y.L.; software, Y.L.; validation, Y.L.; investigation, Y.L.; resources, P.Y.; data curation, P.Y.; writing—original draft preparation, Y.L.; writing—review and editing, Y.L.; visualization, Y.L.; supervision, P.Y.; project administration, P.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China (2019YFB1405000).

Data Availability Statement: Because the commodity trading dataset is from a real trading platform, it cannot be made public for commercial privacy reasons.

Acknowledgments: Yihao would like to thank his friends. They have provided him with emotional support and valuable advice.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AE	autoencoder
ABOD	angle-based outlier detection
AUC	area under the curve
CBLOF	cluster-based local outlier factor
CLL	contrastive learning loss
CNN	convolutional neural network
DAE	deep autoencoder
DeepAnT	deep learning-based anomaly detection approach
ECOD	unsupervised outlier detection using empirical cumulative distribution functions
GM	global model
LODA	lightweight on-line detector of anomalies
LOF	local outlier factor
LSTM	long short-term memory
iForest	isolation forest
RNN	recurrent neural network
OCSVM	one-class support vector machine
PCA	principal component analysis

References

1. Shaw, H.; Taylor, P.J.; Ellis, D.A.; Conchie, S.M. Behavioral Consistency in the Digital Age. *Psychol. Sci.* **2022**, *33*, 364–370. [CrossRef] [PubMed]
2. Cheng, Q.; Zhou, Y.; Feng, Y.; Liu, Z. An unsupervised ensemble framework for node anomaly behavior detection in social network. *Soft Comput.* **2020**, *24*, 6421–6431. [CrossRef]
3. Gupta, C.; Sinha, R.; Zhang, Y. Eagle: User profile-based anomaly detection for securing Hadoop clusters. In Proceedings of the 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, USA, 29 October–1 November 2015; pp. 1336–1343.
4. Bay, S.D.; Kibler, D.; Pazzani, M.J.; Smyth, P. The UCI KDD archive of large data sets for data mining research and experimentation. *ACM SIGKDD Explor. Newsl.* **2000**, *2*, 81–85. [CrossRef]

5. Carletti, M.; Terzi, M.; Susto, G.A. Interpretable Anomaly Detection with DIFFI: Depth-based feature importance of Isolation Forest. *Eng. Appl. Artif. Intell.* **2023**, *119*, 105730. [CrossRef]
6. Hundman, K.; Constantinou, V.; Laporte, C.; Colwell, I.; Soderstrom, T. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 387–395.
7. Audibert, J.; Michiardi, P.; Guyard, F.; Marti, S.; Zuluaga, M.A. Usad: Unsupervised anomaly detection on multivariate time series. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual, 6–10 July 2020; pp. 3395–3404.
8. Yamauchi, M.; Ohsita, Y.; Murata, M.; Ueda, K.; Kato, Y. Anomaly Detection in Smart Home Operation From User Behaviors and Home Conditions. *IEEE Trans. Consum. Electron.* **2020**, *66*, 183–192. [CrossRef]
9. Jiang, J.; Chen, J.; Gu, T.; Choo, K.K.R.; Liu, C.; Yu, M.; Huang, W.; Mohapatra, P. Anomaly Detection with Graph Convolutional Networks for Insider Threat and Fraud Detection. In Proceedings of the IEEE Military Communications Conference, Norfolk, VA, USA, 12–14 November 2019; pp. 109–114.
10. Kim, J.; Park, M.; Kim, H.; Cho, S.; Kang, P. Insider threat detection based on user behavior modeling and anomaly detection algorithms. *Appl. Sci.* **2019**, *9*, 4018. [CrossRef]
11. Tang, J.; Li, J.; Gao, Z.; Li, J. Rethinking graph neural networks for anomaly detection. In Proceedings of the International Conference on Machine Learning, PMLR, Baltimore, MD, USA, 17–23 July 2022; pp. 21076–21089.
12. Breunig, M.M.; Kriegel, H.P.; Ng, R.T.; Sander, J. LOF: Identifying density-based local outliers. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, TX, USA, 16–18 May 2000; pp. 93–104.
13. He, Z.; Xu, X.; Deng, S. Discovering cluster-based local outliers. *Pattern Recognit. Lett.* **2003**, *24*, 1641–1650. [CrossRef]
14. Kriegel, H.P.; Schubert, M.; Zimek, A. Angle-based outlier detection in high-dimensional data. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, 24–27 August 2008; pp. 444–452.
15. Schölkopf, B.; Williamson, R.C.; Smola, A.; Shawe-Taylor, J.; Platt, J. Support vector method for novelty detection. *Adv. Neural Inf. Process. Syst.* **1999**, *12*.
16. Liu, F.T.; Ting, K.M.; Zhou, Z.H. Isolation forest. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 413–422.
17. Papadimitriou, S.; Sun, J.; Faloutsos, C. Streaming pattern discovery in multiple time-series. In Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, 30 August–2 September 2005; pp. 697–708.
18. Pevný, T. Loda: Lightweight on-line detector of anomalies. *Mach. Learn.* **2016**, *102*, 275–304. [CrossRef]
19. Li, Z.; Zhao, Y.; Hu, X.; Botta, N.; Ionescu, C.; Chen, G. Ecod: Unsupervised outlier detection using empirical cumulative distribution functions. *IEEE Trans. Knowl. Data Eng.* **2022.** [CrossRef]
20. Blázquez-García, A.; Conde, A.; Mori, U.; Lozano, J.A. A review on outlier/anomaly detection in time series data. *ACM Comput. Surv.* **2021**, *54*, 1–33. [CrossRef]
21. Malhotra, P.; Vig, L.; Shroff, G.; Agarwal, P. Long Short Term Memory Networks for Anomaly Detection in Time Series. In Proceedings of the 23rd European Symposium on Artificial Neural Networks, ESANN 2015, Bruges, Belgium, 22–24 April 2015.
22. Sharma, B.; Pokharel, P.; Joshi, B. User behavior analytics for anomaly detection using LSTM autoencoder-insider threat detection. In Proceedings of the 11th International Conference on Advances in Information Technology, Bangkok, Thailand, 1–3 July 2020; pp. 1–9.
23. Kieu, T.; Yang, B.; Jensen, C.S. Outlier detection for multidimensional time series using deep neural networks. In Proceedings of the 2018 19th IEEE International Conference on Mobile Data Management, Aalborg, Denmark, 25–28 June 2018; pp. 125–134.
24. Munir, M.; Siddiqui, S.A.; Dengel, A.; Ahmed, S. DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series. *IEEE Access* **2018**, *7*, 1991–2005. [CrossRef]
25. Wen, T.; Keyes, R. Time Series Anomaly Detection Using Convolutional Neural Networks and Transfer Learning. *CoRR* **2019**, abs/1905.13628. Available online: http://xxx.lanl.gov/abs/1905.13628 (accessed on 29 January 2023).
26. Jaiswal, A.; Babu, A.R.; Zadeh, M.Z.; Banerjee, D.; Makedon, F. A survey on contrastive self-supervised learning. *Technologies* **2020**, *9*, 2. [CrossRef]
27. Lapczyński, M.; Bialowąs, S. Discovering Patterns of Users' Behaviour in an E-shop-Comparison of Consumer Buying Behaviours in Poland and Other European Countries. *Stud. Ekon.* **2013**, *151*, 144–153.
28. Zhu, H.; Li, X.; Zhang, P.; Li, G.; He, J.; Li, H.; Gai, K. Learning tree-based deep model for recommender systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 1079–1088.
29. Zhu, H.; Chang, D.; Xu, Z.; Zhang, P.; Li, X.; He, J.; Li, H.; Xu, J.; Gai, K. Joint optimization of tree-based index and deep model for recommender systems. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 3971–3980.
30. Zhuo, J.; Xu, Z.; Dai, W.; Zhu, H.; Li, H.; Xu, J.; Gai, K. Learning optimal tree models under beam search. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual, 13–18 July 2020; pp. 11650–11659.

31. Zhao, Y.; Nasrullah, Z.; Li, Z. PyOD: A Python Toolbox for Scalable Outlier Detection. *J. Mach. Learn. Res.* **2019**, *20*, 96:1–96:7.
32. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 8026–8037.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.