Received 13 September 2023, accepted 19 September 2023, date of publication 13 October 2023,
date of current version 19 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3324371

RESEARCH ARTICLE

DTITD: An Intelligent Insider Threat Detection Framework Based on Digital Twin and Self-Attention Based Deep Learning Models

ZHI QIANG WANG[1], (Member, IEEE), AND ABDULMOTALEB EL SADDIK[1,2], (Fellow, IEEE)
[1]Multimedia Communications Research Laboratory (MCRLab), School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON K1N 6N5, Canada
[2]Department of Computer Vision, MBZUAI, Abu Dhabi, United Arab Emirates

Corresponding author: Zhi Qiang Wang (zwang315@uottawa.ca)

ABSTRACT Recent statistics and studies show that the loss generated by insider threats is much higher than that generated by external attacks. More and more organizations are investing in or purchasing insider threat detection systems to prevent insider risks. However, the accurate and timely detection of insider threats faces significant challenges. In this study, we proposed an intelligent insider threat detection framework based on Digital Twins and self-attentions based deep learning models. First, this paper introduces insider threats and the challenges in detecting them. Then this paper presents recent related works on solving insider threat detection problems and their limitations. Next, we propose our solutions to address these challenges: building an innovative intelligent insider threat detection framework based on Digital Twin (DT) and self-attention based deep learning models, performing insight analysis of users' behavior and entities, adopting contextual word embedding techniques using Bidirectional Encoder Representations from Transformers (BERT) model and sentence embedding technique using Generative Pre-trained Transformer 2 (GPT-2) model to perform data augmentation to overcome significant data imbalance, and adopting temporal semantic representation of users' behaviors to build user behavior time sequences. Subsequently, this study built self-attention-based deep learning models to quickly detect insider threats. This study proposes a simplified transformer model named DistilledTrans and applies the original transformer model, DistilledTrans, BERT + final layer, Robustly Optimized BERT Approach (RoBERTa) + final layer, and a hybrid method combining pre-trained (BERT, RoBERTa) with a Convolutional Neural Network (CNN) or Long Short-term Memory (LSTM) network model to detect insider threats. Finally, this paper presents experimental results on a dense dataset CERT r4.2 and augmented sporadic dataset CERT r6.2, evaluates their performance, and performs a comparison analysis with state-of-the-art models. Promising experimental results show that 1) contextual word embedding insert and substitution predicted by the BERT model, and context embedding sentences predicted by the GPT-2 model are effective data augmentation approaches to address high data imbalance; 2) DistilledTrans trained with sporadic dataset CERT r6.2 augmented by the contextual embedding sentence method predicted by GPT-2, outperforms the state-of-the-art models in terms of all evaluation metrics, including accuracy, precision, recall, F1-score, and Area Under the ROC Curve (AUC). Additionally, its structure is much simpler, and thus training time and computing cost are much less than those of recent models; 3) when trained with the dense dataset CERT r4.2, pre-trained models BERT plus a final layer or RoBERTa plus a final layer can achieve significantly higher performance than the current models with a very little sacrifice of precision. However, complex hybrid methods may not be required.

INDEX TERMS Digital twin, cybersecurity, insider threat, deep learning, transformer, BERT, RoBERTa, GPT-2, data augmentation, artificial intelligence, machine learning, UEBA.

The associate editor coordinating the review of this manuscript and approving it for publication was Chien-Ming Chen.

I. INTRODUCTION

With the increasing popularity of networks and the rapid development of modern hacker techniques, more and more organizations are facing cybersecurity threats from insiders rather than from outside hackers. According to the Insider Threat Report 2019, approximately 60% of organizations experienced one or more insider attacks in 2019 [1]. Because an insider has permitted access to an organization's system, it is possible to cause greater loss and damage than an external hacker. The 2018 U.S. State of Cybercrime Survey indicates that 30% of respondents showed that incidents produced by insider attacks are more expensive or harmful than external attacks [2]. Therefore, insider threats are the largest challenge that cybersecurity needs to address. Organizations spent an average of $11.45 million in 2020 to handle insider threats while they paid $15.38 million in 2021, which rose 34% in one year [3].

As per Gartner's definition, "an insider threat is a malicious, careless or negligent threat to an organization that comes from people within the organization, such as employees, former employees, contractors or business associates, who have inside information concerning the organization's security practices, data and computer systems [4]"

There are three common types of insider threat actors:

- Traitor — an insider that already has legal privileges but abuse their access to carry out malignant activities to the organization's resources.
- Masquerader — an insider that does not have any legitimate privileges or has lower authorization but exploits credentials to take malicious actions against the organization.
- Careless user — an insider who carelessly makes mistakes to leak sensitive and/or proprietary data because of unintentional actions, such as neglect or wrong configurations.

Regarding insiders' malignant activities, insider threat can be classified into the following three categories:

- Fraud — unauthorized inserting, deleting or modifying an organization's data
- Data theft — reading unauthorized data or data exfiltration, for example, embezzling intellectual property from the organization
- System sabotage — directly utilizes information technology to sabotage or harm an organization's system, for example, disruption against data integrity or service availability.

The latest insider risk report [5] showed that data theft is the primary insider threat activity. 42% of insider threat incidents were associated with intellectual property or data exfiltration. Incidental or unintentional disclosure accounts for the percent. Nineteen percent were caused by system sabotage, nine percent due to fraud.

With the rapid growth of insider threats and the huge losses generated by insider threats, highly efficient Insider Threat Detection Systems that can offer secure protection against insider risks have become increasingly important. However, the design and implementation of such a system faces the following challenges:

A. POOR DETECTION PERFORMANCE

It is difficult to detect insider threats especially new or unknown threats. Insider users often have legal privileges in accessing an organization's system. Malicious behavior is buried in a large number of normal user behaviors. Thus, their hidden behavior pattern is not easy to find, resulting in a high negative positive rate.

B. DIFFICULT DATA INTEGRATION

It is difficult to integrate and process data in a timely manner owing to heterogeneous data sources, large data volume, low data quality, and multi-domain data schemas across data siloes.

C. HIGHLY IMBALANCED DATASET

Although we have a large number of records of normal user behaviors, there are few available anomalous data instances. Collection and updating of anomaly data instances are slow and difficult for legal and regulatory reasons. Therefore, the datasets that we can take advantage of are few and often highly imbalanced.

D. INEFFICIENT MODELING METHOD

Insiders' behavior has hidden temporal patterns, but the patterns may drift or shift over time. Most of the existing modeling methods only focus on the behavior category features and ignore the temporal pattern. The remaining modeling methods attempt to model sequential relationships, but cannot handle long temporal sequences since the subsequent step cannot utilize the temporal information learned in the preceding stags. In addition, these models are too complex, computationally expensive, and time-consuming to train. Therefore, it is impossible to implement online training or accurate real-time detection.

E. LACKING EXPLANATION

it is not easy to explain the detection results, why get the results and what need to be changed to improve.

To overcome the above challenges, this paper proposes DTITS, a novel intelligent framework for Insider Threat Detection based on Digital Twins (DT) and Self-attention Based Deep Learning Models. The DTITS comprises three parts:

1) DATA PREPARATION SECTION

This section is composed of an ETL pipeline, Data Stage, and Data preprocessing module. The ETL pipeline module collects and forwards the related log or contextual data to the Data Stage. The data preprocessing module cleanses data, extracts features, designates a numeric token to user activity type and occurrence time, and builds the daily behavior vector for each user. It can also perform contextual word embedding, contextual word embedding substitution, or context sentence embedding using a pre-trained model, such as Bidirectional Encoder Representations from Transformers (BERT) and

Generative Pre-trained Transformer 2 (GPT 2), to augment an imbalanced dataset.

2) MODEL CONSTRUCTION SECTION

This section consists of a DT service, scenario definition, custom model building (mode training, validation, and test) module, and the fine-tuning module of the pre-trained models. This paper proposed a customized transformer named DistilledTrans and applied the original transformer model, pre-trained transformer model (BERT, RoBERTa) plus a final layer, and the hybridmethods combining a pre-trained transformer model (BERT, RoBERTa) with Convolutional Neural Network(CNN) or Long Short Term Memory(LSTM) models to detect insider threats. **Hybrid model** in the context of this study refers to a combination of different deep learning models or a combination of deep learning and machine learning models.

3) INSIDER THREAT DETECTION SECTION

This section is composed of a model inference pipeline, dashboard, and remediation module. The model inference module utilizes a well-tuned trained model to detect whether an insider's activity is abnormal. If yes, the remediation module uses predefined strategies to take measures to disrupt or deter the user's activity to remediate the insider risk and notify the related stakeholders.

The main contributions of this work are as follows:

1) To the best of our knowledge, this work is one of the earliest attempts to learn insider users' behavior patterns using the Digital Twin and Natural Language Processing (NLP) deep learning method based on the self-attention mechanism. This solution is not only able to timely and continuously monitor the insider risk profile of an organization but also easily perform in-depth analysis to find root causes and quickly take appropriate remediation actions against insider threats.
2) This study effectively overcomes the highly imbalanced data issue by performing novel language data augmentation approaches: contextual word embedding insert and substitution predicted by the BERT model and context embedding sentence predicted by the GPT-2 model. This study proves that the latter is a more effective data-augmentation approach for detecting insider threats.
3) This study proposes the insider threat detection framework DTITD, which is an effective all-in-one insider risk detection solution. The complete insider threat detection workflow effectively promotes risk management by identifying and mitigating insider threats, reducing potential harm to the organization, and ensuring the fair treatment of employees.
4) This study built the original transformer model, proposed a custom transformer model called DistilledTrans, fine-tuned pre-trained models (BERT, Robert) plus a final layer, and hybrid models (BERT+CNN, BERT+LSTM) to detect insider threats. Extensive experiments were performed to compare their performances with those of existing models against dense and sporadic datasets. The results demonstrated the superiority of these models over existing approaches. Additionally, they can readily identify unknown or new threats because they detect deviations from the normal behavior profile of each user and label unusual behaviors as insider threats.
5) The experimental results reveal that DistilledTrans trained with sparse dataset CERT r6.2 augmented by contextual embedding sentences outperforms the state-of-the-art models in terms of all evaluation metrics, including accuracy, precision, recall, F1-score, and Area Under the ROC Curve (AUC). Additionally, the structure is much simpler, and thus, the training time and computing cost are much less than those of the recent models. It would be very helpful not only to detect insider threats more accurately and quickly, but also to reduce the overall environmental impact by lowering power consumption and optimizing computing resources.
6) When training models using the dense dataset CERT r4.2, pre-trained models BERT plus a final layer or RoBERTa plus a final layer can achieve significantly higher performance than all the current models, including various hybrid models and two transformer models, with only a very little sacrifice of precision. Meanwhile, the models are much more concise and simpler than all baseline models. This proves that the self-attention structure is a good substitute for convolutional and recursive mechanisms rather than a supplement. The hybrid model makes the structure too deep and complex so that information is lost, thus damaging the performance.

The rest of this paper is organized as follows. In section I, this paper provides a brief introduction to our study. Related work is summarized in Section II. Section III presents the DTITD framework in detail. The experimental setup is presented in Section IV. Section V shows the experimental results and provides a comparative analysis. Finally, conclusions and future work are presented in Section VI.

II. RELATED WORK

In this section, we present related work on insider threat detection. Initially, researchers used a rule-based method to detect insider threats. Nguyen et al. [6] applied rules to detect data exfiltration using signature matching. For instance, if a user accesses more frequently than the threshold or access from an illegal location, the user and his/her activity will be labeled as an anomaly. Hanley and Montelibano [7] proposed a set of hands-on rules for identifying data exfiltration based on Email, HR and LDAP data. ELICIT [8] was introduced to utilize hand-coded rules and signature matching to detect insiders based on contextual data and network traffic. However, rule-based methods require deep domain expertise to perform in-depth analytics to identify signatures of insider threats. The response to insider risk is relatively slow because it is an after-damage solution. With the rapid growth of unknown insider threats, even if the known threat had little variation, it could not be detected because its signature was not known.

In the last decade, similar to some scholars' use of machine learning models with optimized hyperparameters to detect external intrusion, such as Android malware [46], many researchers have tried to apply machine learning models to detect insider threats. This paper summarizes machine-learning model-related works according to their data sources: host data, network data, and context data.

Host data, including system calls, commands, and host logs, are gathered from each computer or server. Maxion and Townsend [9] applied naïve classification to designate posterior probabilities for every test command based on users' historical data. Next, they determined whether a command sequence came from this user by checking the cumulative posterior probabilities. Salem et al. [10], [11] proposed two systems for quantifying abnormalities using the Hellinger Distance. Every subsequent command was converted into a binary vector in agreement with whether a command happened historically. Subsequently, a one-class Support Vector Machine (SVM) was trained to detect the abnormal subsequence according to the frequency of the command. Kudłacik et al. [12] presented a solution for building a fuzzy user profile based on the frequency vector of a user command. All local user profiles can accurately compose the user's daily behavior and temporal patterns. The average probability of a test command can determine whether a command sequence is abnormal. Song et al. [13] introduced a behavioral biometric-motivated system to prevent insider threats. They extracted 18 features from gathered logs to profile the pattern of a user's behavior, for example, the number of files accessed. They were tested using a Gaussian mixture model (GMM), SVM, and Kernel Density Estimation (KDE). Finally, they found that GMM performed the best. A graph-based system was also proposed to address insider threats [19]. They used a bipartite graph approach to model users' relationships with host-based data from various logs, file systems, and psychological surveys. Then, they utilized an Isolation Forest (IF) to detect malicious users from graphs. Al-Shehari et al. [49] applied various random sampling methods, including under-sampling, over-sampling, and hybrid sampling, to deal with the highly imbalanced classes of the CERT r4.2 dataset. They then trained several machine learning models, including Extreme Gradient Boosting (XGB), Decision Tree (DT), Random Forest (RF), and K-Nearest Neighbor (KNN), on these balanced datasets to detect insider data leakage. The results were compared with those of the baselines (the same models trained by the imbalanced datasets) and the existing models. The experimental results demonstrated that their models improved detection performance by effectively solving the class imbalance problem.

Network data are collected from network logs, including proxy, firewall, VPN, LDAP, HTTP browser, and email. DISCLOURE was proposed by Bilge et al. [14] to detect botnets by analyzing the network flow as a whole. The extracted features include flow attribution, client access habits, and temporal information. They used Classification Algorithms such as the DT, SVM and RF models and an ensemble of multiple sub-detectors to detect malicious activities. In [15] and [18], one detector was built on a specific type of network log and then combined into an ensemble model to detect threats so that the information across different types of network logs in the system was allied. More than 100 features were extracted from a great variety of network logs, including email, proxy, and LDAP etc. Several machine learning detectors, including the Markov model, Logistic Regression (LR), and KNN, deal with a particular subgroup of features. Finally, they allocated a score to the user and entity in terms of the number of malicious activities. An alternative method for handling network logs is to employ a machine learning system to handle a full set of features obtained from various types of network logs. Beehive [16] applied PCA to reduce the dimensions of features and then chose 15 features per day per host. They then applied a k-means clustering model to detect abnormal behaviors.

Contextual data are the contextual information of a human user, such as HR and psychological data. Brdiczka et al. [17] applied Structural Anomaly Detection (SAD) to detect anomalies in social networks. A sequential Bayesian model was used to model user connections. They then improved users' psychological profiles by using three features (motivation, personality, and emotional state). Finally, they combined user connections and psychological features to generate an abnormal score for the user.

However, these machine-learning (ML)-based model structures are too shallow to capture complicated user behavior patterns that are often nonlinear and hidden. This often leads to a high negative positive rate and a low detection rate. In addition, they require handmade feature engineering, which is time-consuming and requires domain expertise. Moreover, the training process is often offline and requires periodic re-training. Thus, it is not a good fit for online training, real-time detection, or reaction. Furthermore, ML models are often trained using large amounts of historical data across domains. Because huge amounts of redundant and dirty data exist, parsing and processing these data usually requires a long time. Therefore, like some researchers who presented a deep learning model [47] or a combination of a deep learning model and machine learning model [48] to optimize parameters and detect external cyber-attacks such as malware, an increasing number of scholars have turned to deep learning models to handle insider threats. Compared to ML models, deep learning models can capture deep and salient nonlinear correlations to learn the representation. For example, some deep learning models have demonstrated good performance on sequential data, such as user activities. Thus, it is a good

TABLE 1. The machine learning methods of insider threat detection.

DATA SOURCE	DATA	MODEL	NOTE
HOST	COMMAND SEQUENCE	NAÏVE BAYES CLASSIFIER [9]	
		SVM [10][11]	QUANTIFY THE ABNORMALITY WITH THE HELLINGER DISTANCE
		FUZZY LOGIC [12]	BUILD A FUZZY USER PROFILE ON THE FREQUENCY VECTOR OF A USERS' COMMAND
	VARIOUS LOGS, FILE SYSTEM, AND PSYCHOLOGICAL SURVEY	BIPARTITE GRAPH [19]	USE GRAPH APPROACH TO MODEL USERS' RELATIONSHIP WITH HOSTS-BASED DATA
	VARIOUS LOGS	XGB, DT, RF KNN[49]	DT + UNDER-SAMPLING PERFORMS THE BEST
	BIOMETRICS	GMM, SVM [13]	GMM PERFORMS THE BEST
NETWORK	NETWORK TRAFFIC	AN ENSEMBLE OF MULTIPLE THESE SUB-DETECTORS INCLUDING DECISION TREE, SVM AND RF MODEL[14]	DETECT BOTNETS BY ANALYZING NETWORK FLOW AS A WHOLE
	NETWORK LOG	AN ENSEMBLE MODEL OF SUBDECTOR INCLUDING MARKOV MODEL, LR KNN [15][18]	EACH SUBDETECTOR DEALS WITH A PARTICULAR SUBGROUP OF THE FEATURES
		K-MEANS [16]	USE PCA TO REDUCE THE DIMENSIONS OF FEATURES AND THEN CHOOSE 15 FEATURES ON A PER DAY PER HOST BASIS
CONTEXT	HR AND PSYCHOLOGICAL DATA	SEQUENTIAL BAYESIAN MODEL[17]	COMBINED USERS' CONNECTION AND THE PSYCHOLOGICAL FEATURES TO GENERATE AN ABNORMAL SCORE FOR THE USER

fit for modeling complex user behaviors and accurately identifying abnormal users. In addition, deep learning models are easier to integrate heterogeneous data sources across multiple domains. Moreover, feature engineering is relatively simpler, without too much prior knowledge and human labor efforts. Furthermore, some unsupervised learning approaches have strong learning capabilities without labeling works.

In this study, we generally classify deep learning model related works according to their model architectures: **deep feedforward neural network (FNN)**, Convolutional Neural Network (CNN), **Recurrent Neural Network (RNN)**, and **Graphic Convolutional Network (GCN)**.

FNN: Lin et al. [20] proposed an unsupervised hybrid Deep Belief Network (DBN) based model to detect insider threats. The model was built using multilayer Restricted Boltzmann Machines (RBMs) piled together. Features extracted from audit logs in five domains (logon, device, email, http, and file log data) were loaded into the model. The output of the learned features from the last salient layer was then entered into the One-Class Support Vector Machine (OCSVM) model for insider threat detection. This is a good fit for the solution of combining multi-domain features rather than a single domain to analyze user behavior patterns. However, it cannot model temporal data and produces many false alarms because of the information loss during feature extraction. An ensemble of multiple deep autoencoder-based models was proposed in [21]. The training data were obtained from logon, file, device, and http logs. Features from each data source were fed to the respective deep autoencoders to modeling the normal behavior of the users. Deep autoencoders are used to minimize reconstruction errors. The outputs that had relatively higher reconstruction errors than the threshold between the input and decoded data were assembled to compose an ensemble. A user's overall maliciousness score was jointly calculated based on this.

This FNN approach effectively enhances the accuracy with a low negative positive rate without prior expertise, because the cascaded multiple encoder and decoder can capture hidden nonlinear relationships between user behaviors. However, because most features are simple aggregated counts of activities, which are frequency-based, the approach is too general for insider threat detection because it still cannot capture temporal patterns. Additionally, the autoencoders use data late fusion in which deep autoencoders are working on each category data first and then incorporated rather than fusion data together at the beginning. Therefore, the approach loses the opportunity to capture correlations across multiple domains in the early phase.

RNN: User activities ordered by time within a time window can be generally regarded as a time sequence. To capture temporal patterns, RNN-based models have attracted increasing interest. The basic concept is that an RNN model is trained to predict the subsequent activities of a user. If the prediction results deviate substantially from the user's real activities, the user's activities may be anomalous. However, the classic RNN is not easy to train, especially for long sequences, because of the exploding or vanishing gradient. Researchers often use Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) instead to model temporal pattern. An unsupervised insider threat detection system was proposed by Tutor et al. [22]: the Feature Extraction module aggregates the counts of user activity from various sources (logon, device, email, http, and file log data) and outputs a temporal vector per user per day. The LSTM model and density clustering model took it as input and output a series of hidden state vectors. Then, the outputs were input into the structured prediction model to compute the anomaly scores using the negative log-probability of user actions. This type of model is a more appropriate method for Insider Threat Detection, as it can effectively learn temporal user behavior patterns by using hidden units from normal temporal sequences and identify malicious behavior when the activity is essentially different from normal behavior predicted from the trained model. Additionally, the model can be gradually updated online; thus, it is able to automatically evolve with new patterns over time. However, this is too slow to achieve long-term temporal dependency. Moreover, it is able to find outliers of users' behaviors but cannot assure whether they are true positives with high confidence. Furthermore, it may fail to detect malicious activities within one day. In contrast to this method, the system proposed by Lu and Wong [23] not only has a historical user behavior analytics component similar to this method but also has an online monitor and real-time detection component based on LSTM. The model can use a time sequence as an input to detect insider threats and self-evolve during the online monitoring process. More context data were extracted from the CERT dataset to enrich the feature set, such as employees' roles, positions, personalities, and relationships with other employees. All of them increase confidence in determining insider threats. Yuan et al. [24] trained an LSTM model to predict the next user's activity and obtained a group of temporal hidden states to produce a fixed-size feature matrix, which is fed into the CNN classifier to generate the probability of abnormal behavior based on the discrepancy from normal activity sequences. Hence, their model can detect abnormal behaviors occurring in a single day. Yuan et al. [25] designed a hierarchical neural temporal point-process model to detect insider threats. Their system captured the users' behaviors, including not only activity types but also time information at two levels: a lower-level LSTM was used to predict whether the next session was malicious by learning intra-session information, for example, activity types and activity time intervals on the intra-session level, while an upper-level LSTM was used to learn inter-session information to predict the intervals between two sessions and the duration of the next session. Finally, they computed an anomalous score according to the gap between the predicted results and the real action time and type. Haq et al. [45] adopted the word embedding methods Word2vec and GLoVe to encode user activity information and then fed it into the LSTM model. Next, they compared state-of-the-art ML models such as eXtreme Gradient Boosting (XGBoost), Ada-Boost, Random Forest (RF), K-nearest Neighbors algorithm (KNN), and Logistic Regression(LR). Their results showed that XGBoost performed best in terms of accuracy (92%). Nevertheless, their word2vec+ LSTM and GLoVe + LSTM models achieved accuracies of only 73.4% and 74.0%, respectively. Singh et al. [50] introduced a hybrid learning method for detecting insider threats. They utilized a bidirectional LSTM model to extract features, a feed-forward artificial neural network to select relevant features using distance measurements, Euclidean distance, Cosine similarity, and standard deviation, and a SVM classifier to detect abnormal users. Meanwhile, they applied the genetic algorithm's fast global search strategy for the SVM's initial kernel selection to enhance its performance. Finally, alerts were triggered for every user if their aggregated anomaly scores exceeded the threshold.

Overall, RNN is still likely to fail to learn long-term temporal patterns. In addition, training an RNN is time consuming because the RNN structure is too complicated and it cannot make use of the parallel computing capacity of the GPU but takes words in sequence.

CNN: CNN model performs well in computer vision because whether an image or voice is Euclidean data with a regular spatial structure. However, it is not suitable for capturing temporal sequence patterns, particularly time series. In addition, it requires a fixed-size feature matrix as input. Saaudi et al. ba proposed a solution consist of CNN and LSTM. The CNN was composed of two layers: a convolutional layer with ReLU and a max-pooling layer. Each log sequence (logon, file, http, device, email logs) was combined and then transformed from a list of strings to a list of the character indexes with the same length through embedding and padded by 0. These feature map matrices were then fed into the CNN. Next, they applied LSTM to model the temporal patterns between the extracted features. Finally, a regular dense neural network layer was utilized to classify the users as normal or malicious. The experimental results showed that the CNN-LSTM model outperformed the CNN and LSTM models. This approach takes advantage of the merits of CNN for capturing the local context and LSTM for modeling the temporal pattern. It does not require handcrafted features and implements automatic and fast feature representation learning.

However, the CNN-based model still easily loses some valuable information because of the pooling layer and requires a large dataset with labels for training. In addition, it is not suitable for modeling features without spatial relationships.

GCN: All the above models only care about users' property information, which may result in a high false-positive rate. However, GCN applies a graph structure to capture the correlation between users and entities. GCN is an extension of CNN, which is a class of neural networks for learning graphic representations of data, such as node features and structural information. Jiang et al. [27] used a graph structure to model the correlations among users, for example, communicating by email or working on the same host. They also combined user behaviors, user profile information, and entity properties into feature vectors of nodes. They then input them to the convolutional network and chose cross-entropy as the objective function to predict anomaly users (nodes) in the graph and their related malicious groups. Li et al. [51] presented Dual-Domain Graph Convolutional Network (DD-GCN) to fuse user behavior information from feature domain and topology domain into a high-level representation. They applied a weighted feature similarity mechanism function to develop heterogeneous graphs based on the original users' relationship information and their activity feature similarity. They then extracted specific graph embeddings from the original topology domain and graph concurrently. Next, the two types of embeddings were fed into an attention mechanism to learn the importance weights of the user's embeddings in the two domains and fused together. Finally, the output module classified the output to complete insider threat detection tasks.

The GCN is not required to label all the nodes in the graph, and thus, partly labeled data can be used to train the model. Only the connection relationships between the users captures the structural information of the network. Thus, structural information seems to carry less weight on improving the accuracy of insider threat detection. As a result, the GCN's performance lags behind that of the RNN when it is fed with a large imbalanced dataset. Additionally, it is not easy to implement because it requires prior knowledge and significant effort to construct a complex graph structure to learn its topology as the input of the model.

From Table 2, we can see that the deep FNN cannot handle time-series data to model temporal patterns. The CNN model easily loses valuable information because of the pooling layer and requires a large dataset with labels. Thus, it is not suitable for modeling time sequence patterns. The GCN is able to model the behaviors of the users and their relationship, but its accuracy needs improvement and requires prior knowledge and efforts to construct the graph. RNN is capable of handling sequential data but is limited by the slow training time and capturing the long-term sequence dependencies. The LSTM provided a fair solution to the long-range sequence dependency problem but disregarded parallel computations; thus, the training is even slower than that of the RNN.

The **digital twin (DT)** concept was introduced by Grieves et al. [28] in 2002 as an integrated probabilistic simulation of a physical asset or system to emulate the lifetime of its matching physical twin by using the best existing physical model, history information, latest sensor data updates, etc. DT was renewed by El Saddik and Abdulmotaleb [29] in 2018 as a digital replica of a living or non-living physical entity. DT synchronizes with the physic entity through smooth data communication. In contrast to many industrial or manufacturing DT applications that often mimic

TABLE 2. Comparison of the deep learning methods of insider threat detection.

Approaches	Description	Advantages	Disadvantages
Deep Belief Network[20]	The output of stacked multilayer RBMs is fed into the OCSVM model for threat detection	It is suitable for multidomain feature processing	It cannot handle temporal data and generate lots of false alarms
Autoencoder[21]	4 autoencoders detect anomalous on each corresponding data set and then combine their outputs to form an ensemble to list top N malicious insiders	Autonomous organization level offering few false alarms	It cannot handle temporal data; late data fusion loses the early chance of modeling correlations across domains
RNN[22][23][24][25][50]	RNN especially LSTM or GRU takes user activity time sequence as input and output; the output are fed into prediction model to calculate anomaly score	It can efficiently capture temporal pattern in data	Slow and expensive training; tend to forget long term dependency; it can find suspicious event but cannot ensure if it is insider threat
CNN-LSTM[26]	CNN is used to extract features and LSTM is for capturing temporal dependency. A dense layer is used to classify whether a user is malicious	It can efficiently capture temporal pattern in data; it also enable faster processing of matrices by using CNN	Slow and expensive training; easily loses much valuable information due to pooling layer and need a large labeled dataset
GCN[27][51]	It applies a graph structure to model the correlation between users and users' properties. Then input them with other data into a CNN to predict anomaly users and their community	It can model the correlation between users and users' behavior	It is difficult to implement; it requires prior expert knowledge and much efforts to construct the graph

a component of the industry system or the whole industry system, a digital twin in the context of this study is a digital counterpart of an actual real-world physical insider, or an entity such as a computer or device, and its relationship with other digital twins. This is used to simulate the behavior of a specific user in an organization.

III. PROPOSED FRAMEWORK

This paper proposes an intelligent insider threat detection framework, DTITD, based on Digital Twin and self-attention-based deep learning models. To the best of my knowledge, this paper is one of the first papers proposing and implementing such an idea.

A. DTITD FRAMEWORK

Insider threat detection is a continuous process that requires frequent data processing, data augmentation, model training, model retraining and fine-tuning, model validation and testing, and model inference throughout its lifecycle. In addition, it often involves huge amounts of user activity and user profile information that are sensitive and private. This information cannot be accessed by unauthorized people. Therefore, this

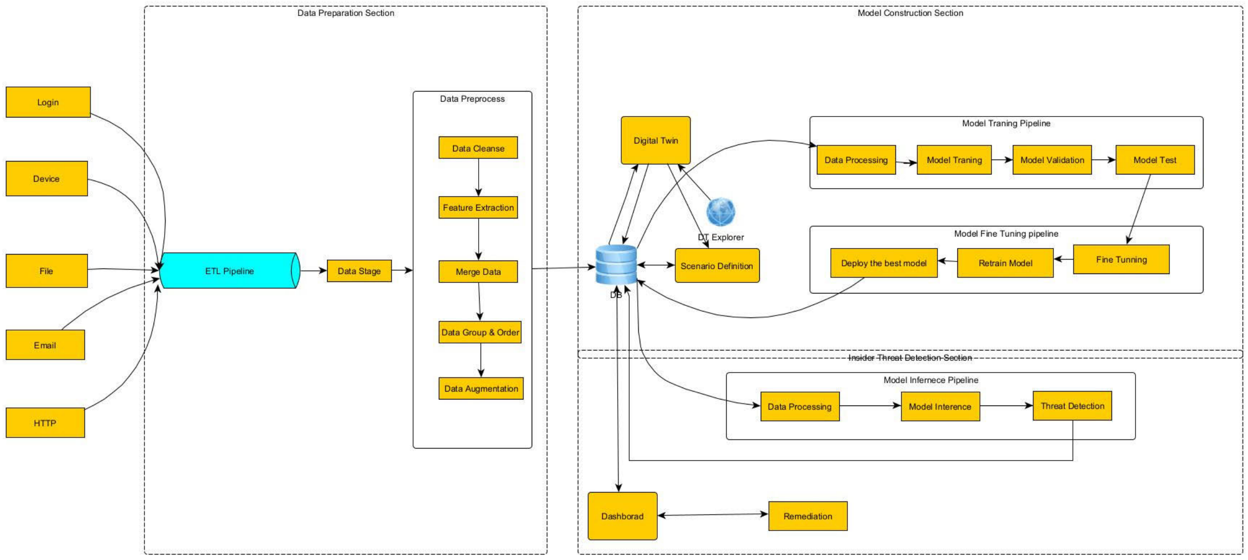

FIGURE 1. **The framework of DTIDT system.**

study applied the digital twin technique to create a DTITD framework to timely and continuous monitor the insider risk profile of an organization and take remediation actions. Adopting DT brings about the following benefits:

1) The application of digital twin technique in insider threat detection allows the corresponding physical users and entity's entire lifecycle and their correlations to be modeled, simulated, and regularly synchronized with the corresponding physical twins. With the power of digital twin graphs and digital twin's properties updated by user activity and user profile information in real time, users' current behaviors and their changes over time are mimicked. Thus, we can easily obtain an accurate broad overview and in-depth views of the insider risk profile at the organizational level and quickly take measures to address the risks.

2) DTITD is also an advanced AI solution for detecting insider threats in near real-time. It allows heterogeneous data from various sources to be quickly integrated, processed, and fed into downstream machine learning or deep learning models to detect insider threats. DTITD is a good implementation of User and Entity Behavior Analytics (UEBA). UEBA is a type of cyber security process that not only models the regular behavior of users and entities but also associates user behavior with entity behavior. The term ''entity'' in the context of cyber security can refer to machines such as PC, networked devices, servers within the network, or IT systems. Thus, the UEBA follows all the users and entities in an organization, rather than passively monitor devices or respond to security events. A normal user profile is the user's activity sequences and frequencies, which are shown on a usual basis. By modeling such baselines, the UEBA can detect any abnormal actions when they are significantly different from normal user profiles. Therefore, it can more accurately detect dubious behavior, likely threats, and attacks that traditional security systems may not detect, especially new or ''unknown unknowns,'' risks that traditional detection methods are neither aware of nor understand.

3) This framework not only enhances cybersecurity but also aligns with broader Environmental, Social, and Governance (ESG) goals, contributing to a more sustainable, responsible, and ethical business environment. It provides transparency and interpretability to insider threat detection and response processes and thus improves corporate governance practices. Meanwhile, it safeguards sensitive data, protects individuals' privacy, and complies with data protection regulations.

Figure 1 provides an overview of the DTITD framework. The DTITD comprises three sections: data preparation section, model construction section, and insider threat detection section. The ETL pipeline collects and filters various real-time log data, including login, device, file, email, and HTTP log data, and then moves them from the data sources on premise to the data stage on the cloud. Raw log data are preprocessed in four phases: data cleaning, data merging, feature selection, data group, and order. Subsequently, preprocessed data will persist in the database and update the properties of the digital twin. The scenario definition defines one scenario by filling the desired user-defined parameters. Then, it triggers the model training pipeline so that data are further processed, and the deep learning model learns the scenario's user behavior from the historic data. Next, the super-parameters of the models are fine-tuned to compare performance and select the best model. After that, the model inference pipeline is triggered and periodically run so that real-time data from logs can be processed in a timely manner, and the trained model will be run to detect insider threats in near real time. The insider threat detection results are sent back to an associated result table in the DB. If there are abnormal users or activities, the system will create an incident to notify the dashboard and update the DT graph. The dashboard will show the detection results. Security analysts can take remediation actions to revert the possible damages done or prevent the incident from happening again.

A DTITD system consists of the following components:

1) MODEL CONSTRUCTION SECTION

Digital Twin Service: The Digital Twins service instance is a platform as a service (PasS) storing digital DT models and DT graphs with their states, and orchestrated event processing on the cloud. DT graph is a graphic representation of the DTs of users and entities and their relationships. Users are represented by circular nodes, whereas entities are represented by rectangular nodes. User DTs' properties contain rich user profile information, such as IP, port, and normal work hours. Various edges capture the structural information and interactions occurring among nodes, for example, assigned PCs, teammate relationships, same role relationships, teammate relationships, connections on the same device, or email communication. The file, Email, and Http web browse features can also be incorporated into the DT graph. DT Explorer is a client application that allows users to quickly create DT models, manage DT models and graphs, and run queries to find existing DTs with simple drag-and-drop actions. Security analysts can also explore insights from existing DTs or share them on the DT service platform using the DT query API. DT services can generate events that trigger external computing resources handling business logic and data processing. This module is achieved by using Azure Digital Twin Services.

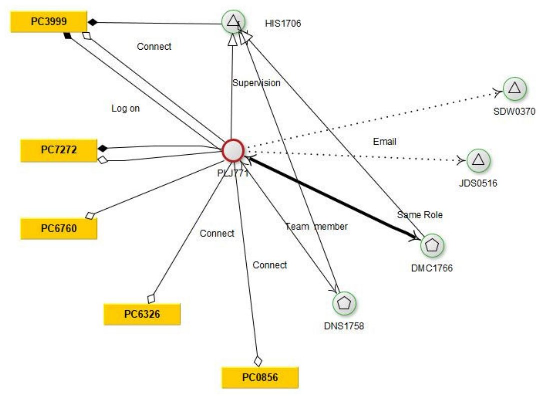

FIGURE 2. **DT graph example.**

For example, Figure 2 shows that IT administrator PLJ1771 has an assigned PC, PC7272. He has team members DNS1578 and DMC1766, who have the same role as administrators. He also has email communications with user SDW0270 and JDS0516 who are managers in other departments. He was connected to PC0856, PC6326, PC6760, and PC7272. This is normal behavior since he performed his daily work on these PCs. However, he connected to his supervisor's machine PC3999 by coping with a keylog file to that machine and logging on to that machine. This behavior is very suspicious, and thus the user's status is changed to red.

Digital Twin's work process is as follows: First, create a client connection to Digital Twin instance using its URL endpoint. Then, create or upload DT models using the Digital Twin Model Definition Language, making sure to include the properties for the time-series and the graph relationships. Next, DTs are built for the nodes in the DT graph by referencing the correct model ids, and initializing the properties. Subsequently, the relationships are added to the DT graph using the defined topology. Finally, the properties are updated using real-time incoming data, including time-series data and user category information

Scenario definition: One scenario is defined by filling in the necessary user-defined parameters. It includes querying the twin graph to obtain a scenario's required context information, for example, Digital Twin ID entries, selecting the target properties in the scenario, for example, the behavior and category features, from the digital twin graph, and selecting data source information in the DB, for example, table name and column mapping.

Data Storage: Through Sink link, the data are fed into the data stage tables in the Azure log analytics workspace. The Azure log analytics workspace is a data stage that has reliable and scalable abilities to persist collected and preprocessed data with strong log management capabilities. An Azure Logic App runs a Kusto Query Language (KQL) every 15 minutes to extract new fed data, converts the data into normalized relational schema, and loads them into the Azure SQL DB for insider threat detection. The DB stores the pre-processed training dataset for model training. Moreover, DB also stores the DT models, trained ML model metadata, and prediction results. User activity data are automatically extracted from the Azure Log Analytics Workspace and archived in the staging tables in the DB. DB runs a scheduled script to update digital twin properties by using Azure Functions through the event hub. Prediction result tables are created for every scenario, and take the name of the scenario. The prediction results includes a boolean value "isInsiderThreat" indicating the prediction result, severity of the anomaly, anomaly score, feature's importance reflecting each feature's contribution to the prediction results, and other context information send back form ML inference pipeline. Furthermore, we can directly upload data to a DB table by importing data from local files.

ML Model Training pipeline: It takes in preprocessed data to train, validate, and deploy the ML model. It often involves the following steps: move the training dataset from Azure SQL DB to Azure Blog Storage; create the Run Configuration defining the environment needed to run the pipelines on the compute target; create a Data Preparation Pipeline Step where we run Python script to retrieve the raw input data from Azure blob storage, process the input data, and output the processed data that will be used in the subsequent steps. In the Model Training Pipeline Step, we use the processed data as the model input to train the models. In the Model Validation Step, we validate the model using validation data sets. In the Model Deployment Step, we register the model and deploy it to the

Web service endpoint. Then, an experiment is created, the pipeline run is submitted, and the pipeline run is monitored. Finally, it often needs fine tune the super-parameters of the model to obtain the best model. This pipeline is developed using Azure Machine Learning Studio.

2) DATA PREPARATION SECTION

ETL pipeline: Our data includes user behavior data (host logs, network flows, etc.) and user profile data (user identities, HR data, psychological information). Initially, Fluentbit collects real log data from data source systems and moves to the Kafka pipeline source topic. Then, the Kafka pipeline filters the original data flow to obtain the data that we are interested in and transforms the data into our desired data format according to the filter rules and transformation rules defined in the configuration files. Next, we send the filtered and converted data to the data stage on the cloud platform (Azure log analytics workspace) through Sink Link Connectors.

Data Preprocessing Module: Data preprocessing module includes data cleansing, feature extraction, data merging, data grouping and sorting, and data augmentation. We perform data cleansing to eliminate dirty data and fill up missed values, extract and generate useful features from cleaned data such as user ID, activity time, activity types, etc., combine data from five domains: login, device, file, email, and http, group the data by each user and each day, and sort each user's behavior by occurrence time to form a user behavior sequence for each day. It can also augment the data for a highly imbalanced dataset. This study applied context embedding insert and substitution by using a pre-trained transformer model BERT and context sentence embedding by using a pre-trained transformer model GPT2. (The details are in Section III, Proposed Framework D. SEMANTIC VECTORIZATION 3) Feature Engineering)

3) INSIDER THREAT DETECTION SECTION

ML Model Inference Pipeline: Once model tuning is done, a trigger will periodically trigger the model inference pipeline to apply trained and registered models to make predictions to detect insider threats. The model inference pipeline consists of two steps: Data Prep Pipeline Step processing of the new batch input data, and Inference Pipeline Step running the trained model to make inferences on the newly processed data. This pipeline is developed using Azure Machine Learning Studio.

Trigger: The ML model training pipeline and the ML model inference pipeline can be triggered using either the Azure Logic App or Azure Machine Learning Pipeline Schedule. For example, when the Logic App detects that a data file has been added to the container, it automatically triggers the ML model inference pipeline.

Python Scripts: Python scripts run in Docker containers in the Azure Container Registry and perform the following functions:

- Preprocess the raw data and convert the data to the format desired by the deep learning model.
- In each step of the model training pipeline or model inference pipeline, Python scripts are run to read the relevant input data, train models, generate predictions, and store the results in the DB.
- Visualize and interpret insider threat results.

The configuration parameters are saved in a JSON file that can be easily reused, modified, or extended by users to meet the requirements of various scenarios.

Dashboard and Remediation Module: This module acts as the control component (actuator) of the DT, conveying physical feedback to control user behavior. It is responsible for alerts, incident management, and anomaly response **:** A scheduled query is run every 15 min on the prediction table in the DB. When it finds abnormal users or activities, it creates an incident, raises an alarm, and updates the DT graph and dashboard. The dashboard presents all the security metrics and shows the alarm and related context information to the security staff. A security analyst can analyze related information, investigate the root cause, and take appropriate actions. One option is to trigger a remediation workflow to revert the damage and prevent the incident from occurring again. For instance, they can block the user so that it does not have access to the system or can revert malicious actions. This module is developed using the Azure Sentinel and Azure Logic Apps.

B. DATASET

The CERT is the most popular insider threat open dataset, which is a set of synthetic scenarios, malicious users, and their behavior data. It was developed by the CERT Division of the Software Engineering Institute at Carnegie Mellon University. It simulates the IT-based activities and personal profiles of all employees in a synthetic organization named DTAA. It consists of five log files: logon.csv archives all employees' logon and logoff actions on PCs; email.csv archives all the email activities of employees; http.csv collects all the web browsing history records of employees; file.csv documents file-related operations (open, write, copy, or delete); and decive.csv maintains the connection information for the removable drive. In addition, the CERT dataset provides HR data for all employees in LDAP.csv and the psychometric score for all employees in psychometric.csv.

There are multiple versions of the CERT datasets based on the generated time. The most commonly used version is r4.2, and the least used version is r6.2. CERT r4.2 mimics a corporation with 1000 employees and 32,770,227 activities, where 70 employees are malicious users and 7233 activities are anomalous in three scenarios of insider risk. The three scenarios are data exfiltration, embezzling intellectual property, and system sabotage. In contrast, CERT r6.2 illustrates a much larger corporation with 4000 employees and 135,117,169 activities, where only five users are malicious and only 470 activities are anomalous. There are five scenarios of insider risk, but only one malicious employee for each scenario. The newly added scenarios 4 and 5 are all about intellectual property theft, but they all occurred in one day. Compared with r4.2, r6.2 is a sporadic dataset, and thus insider threats are much more difficult to detect because it contains much fewer malicious users and their activities, but much more users' normal behaviors. We performed our experiments on both the v4.2 and v6.2 dataset. To overcome the challenges of the v6.2 dataset, we have to find good data augmentation approaches to generate synthetic data to overcome data imbalance and a good deep learning model to model the users' behavior patterns instead of shallow machine learning models and manual feature engineering.

C. DATA ENGINEERING

In contrast to other related studies, this study uses the Natural Language Processing (NPL) approach to detect insider threats. This method considers each activity as a word, a user's daily activity sequence as a sentence, the entire set of users' activity sequences as a book, and user behavior patterns as hidden seminar rules. This study not only uses NPL to process data, encode and tokenize data, and perform an NPL model based on a self-attention mechanism to detect anomalous user behavior, but also uses the NPL method to perform data augmentation to overcome the data imbalance.

1) DWONSMAPLING MAJOROTY CLASS IN CERT R4.2

Compared with CERT r6.2, CERT r4.2 is a dense dataset because it has many more malicious users and activities. However, this dataset is still imbalanced. We must downsample the activity data of normal users to reduce the imbalance. To determine which down-sample algorithm is the best, this study applied RandomUnderSampler, AllKNN, NearMiss, and Tomek Links algorithms to generate downsized samples and then split them into the training dataset and the test dataset. the Gaussian NB classifier was trained on the training dataset. Next, we used the trained model to classify the test dataset and measure the prediction accuracy. The results are shown in Figure 3, where Tomek Links [44] performed the best; therefore, this study chose Tomek Links to down-sample normal users' behavior data.

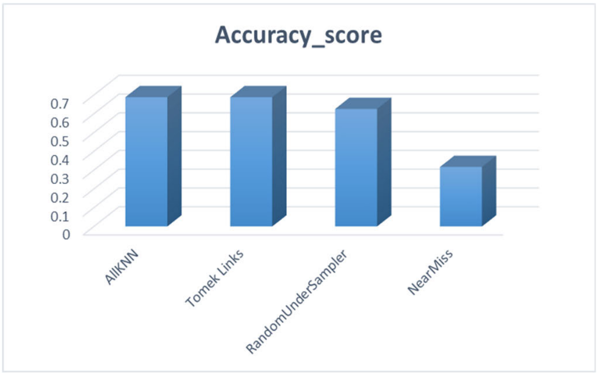

FIGURE 3. **Accuracy score of various down-sampling algorithms.**

2) UPSAMPLING MINORITY CLASS IN CERT R6.2

Because there are very few anomalous user behavior data in the experiment dataset CERT r6.2, the dataset is extremely imbalanced. Whatever the models we use, they will always predict the class with the majority class because we do not have enough minority class data to build robust models to distinguish between the two classes: normal and anomaly. This study has to synthesize new data from the original dataset to up-sample the minority class and thus decrease the imbalance of the data.

Scholars have often used the Synthetic Minority Oversampling Technique (SMOTE) algorithm to address this issue. The Smote algorithm is not a good fit for time-series data but fits scalar features. We can vectorize the text using the Tfidf Vectorizer and then apply the SMOTE algorithm. However, transformer-based models such as BERT and Roberta often have their own contextual tokenization. To optimize the performance of the models, this study adopts text augmentation approaches in the natural language process to upsample minority classes. However, augmenting text data is much more difficult than augmenting images (rotation, flip, sharpening, cropping, etc.). We have to ensure that synthetic data does not change the user's hidden behavior pattern (seminar and grammar) in the temporal sequences (sentence). The simplest method is to insert or replace words using static word embeddings, such as GloVe or Word2vec. However, just as not every word has a synonym, not every activity can be replaced by another one. Even if you find one to replace it, the meaning of this time sequence can be totally changed because the context is different. To solve this issue, contextualized word embeddings are used to insert or replace target words, because they can consider surrounding words to generate different words in different contexts.

a: WORD LEVEL AUGMENTATION

This study used a powerful pre-trained transformer model, BERT, to insert or substitute activities (words) to generate synthetic data to upsample the minority class. The BERT model [31] is a pretrained transformer-based model developed by Google. BERT was pre-trained on unlabeled data from large datasets: Books Corpus (800M words) and English Wikipedia (2,500M words). It was pre-trained on the masked language prediction task and the next sentence forecasting task. There are two models for the original English language, BERT:

(1) BERT-based model:12 encoders with approximately 110 million parameters. There are 12 bidirectional self-attention heads and 768 hidden layers in every block.

(2) BERT-large model:24 encoders with approximately 340 million parameters. There are 24 bidirectional self-attention heads and 768 hidden layers in every block.

We selected a BERT-based model. Instead of randomly selecting one word, this study used the BERT model to predict possible inserted words based on the context it pre-learned. Rather than substituting a word with a static synonym or word embedding, the BERT model takes the surrounding words in the context as input to predict the target word. Because the target can occur in any position of the sentence, as a pre-trained transformer-based model for language understanding, BERT is good at learning both leftward and rightward contexts due to its bi-directional architecture.

b: SENTENCE LEVEL AUGMENTATION

This study also generated a new time series (sentence) using a pre-trained transformer model GPT-2. Unlike the former method, we does not insert or replace a single or few activities (words), but instead generates the whole time series (sentence). This study used a pre-trained transformer-based model named GPT-2, which is well known for language generation. Generative pre-trained transformers (GPT) are a family of pre-trained varied transformer models for generative language learning, which was proposed by OpenAI in 2018. In contrast to large language understanding pre-trained models such as BERT or RoBERTa, the unique features of GPT is its ability to generate coherent and contextually relevant text given a prompt or input because it is designed for autoregressive text generation tasks rather than sequence-to-sequence tasks like machine translation or text classification. GPT-1 is a 12-level transformer decoder with 12 heads, and a linear layer and a softmax layer are on top of those with 117 million parameters in total. It was pre-trained on 4.5 GB text from BookCorpus. GPT-2 [34] is a scaling up of GPT1 that shares the same architecture as GPT-1, but with modified normalization. It is pre-trained on 1.5 billion web text, including 40GB of text and has 1.5 billion parameters. Because we utilized the next sentence ability of GPT-2, the similarity of the generated synthetic data to the original data would be lower than that of the contextual embedding word.

D. SEMANTIC VECTORIZATION

CERT data set is composed of two categories of data:

1) USER'S BEHAVIOR DATA

User behavior is characterized as a sequence of activities that users perform in an organization. User behavior data are obtained from different real-time logging systems, such as logions, devices, files, emails, and HTTP logs. They are often required to be gathered and handled in a timely manner in order to quickly detect and respond to anomalous behaviors. They are saved in http.csv, logon.csv, device.csv, file.csv, and email.csv.

2) USER'S PROFILE DATA

A user's profile information refers to the user's background or context data, which can include the employee's position, role, relationships with other users, permitted access, and psychometric properties. These are saved in LDAP.csv and psychometric.csv.

3) FEATURE ENGINEERING

The feature engineering process is shown as follows:

1) The process starts by loading five domain data files to their corresponding data subset after cleansing the data to remove dirty data and fill up missed values.

2) Extract and generate useful features from the data subsets: As shown in Figure 4, this study extracted features

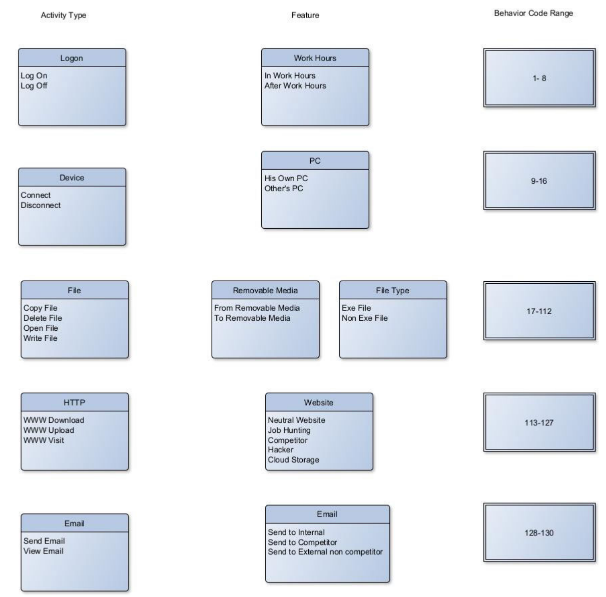

FIGURE 4. **User behavior code.**

such as user IDs, activity time, and activity types including Logon (Log On, Log off), Device(Connect, Disconnect), File(Copy File, Delete File, Open File, Write File), HTTP (WWW Download,, WWW Upload, WWW visit, and Email (Send Email, View Email), and created features such as Work Hours (In workhours, after workhours), PC (user's own PC, other user's PC), removable media (from removable media, to removable media), file type (exe file, non exe file), website (neutral website, job hunting, competitor, hacker, cloud storage), and email (send to internal, send to competitor, send to external non-competitor) from existing data. We takes 8:00 am to 5:00 pm as working hours and the period outside this range as the after-working hours. This paper also regards the PC that a user uses most often as his/her assigned PC.

3)Encodes user behavior to a numeric token according to activity type, time information, and related features.

$$x = B * 24 + h \tag{1}$$

where x is a numeric token of the behavior that can distinctively identify user behavior, B is the behavior code based on activity type and related features (Figure 4), and h denotes the occurrence hour from the beginning of the day.

4) All data subsets are merged into one dataset. Group the data by each user and then by each day, and sort each user's behavior by the occurrence time to generate a user's behavior sequence for each day. Thus, a user's daily activities comprise a long string of activities on that day. For example, log on, visit web site, copy files, connect to device, send email to colleagues..., log off, etc.

5) Combine the dataset with insider.csv to label the user behavior sequence as normal or anomalous.

6) Text data are converted into numeric vectors. We accomplish this by transforming each word into a word embedding. This study takes the textual sequence of the user's daily activities as input, obtains the vocabulary of the word index,

transforms the input from a list of words into a list of indexes in the vocabulary, pads the sequence to have the same length using 0, and outputs a two-dimensional matrix whose every row is a dense vector containing the information of the word's meaning.

E. SELF-ATTENTION BASED MODEL

The transformer model was introduced in 2017 by a team at Google Brain [30] and is increasingly becoming one of the first choices in Natural Language Processing (NLP). We applied this model to insider threat detection to capture the temporal dependency of user behavior. This model is the first sequence model that relies entirely on self-attention to model global temporal patterns between the input and output without any recurrent layers or convolutional layers that are most frequently used in temporal sequence modeling. Self-attention is an attention structure that associates different positions of a single sequence to output a depiction of the sequence. This study regards the input as a set of query matrix (Q) and key matrix (K) value matrix (V). Attention is calculated as a weighted sum of the values, where the weight corresponding to every value is the softmax function of the dot products of the query with keys, divided by the square root of d_k which helps prevent varnishing gradients in training.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2}$$

where d_k is the dimension of keys.

The original transformer has a typical encoder-decoder framework. The encoder converts an input sequence of symbol representations $(x_{1,\ldots,}x_n)$ into a sequence of intermediate representations $z = (z_1, \ldots, z_n)$. The decoder then converts z into an output sequence $(y_1, \ldots, y_m)$. At every step, the model is autoregressive, which means that the output sequence depends linearly on its own previous sequence and on a stochastic term.

The encoder consisted of a stack of six identical blocks. Each block has two sublayers. The first is a multi-head self-attention structure, while the second is a position-wise fully connected feed-forward network composed of two linear layers activated by the ReLu function. Similarly, the decoder consists of six blocks. The first difference from the encoder is that multi-head attention takes keys and values from the output of the encoder and queries from the output of the previous decoder layer. The second difference is that the first self-attention layer in the decoder block is masked to avoid attending to consequent positions, and the output embedding is right shifted by one position to ensure that the predictions are autoregressive.

In contrast to RNNs, transformers simultaneously handle the entire input. The original input text is parsed into tokens by a byte pair encoding tokenizer, and the tokens are converted into equal-length vectors using word embedding and padding. The positional information of the token is then added to the word embedding.

$$PE_{(pos,2i)} = sin(pos/10000^{2i/dm}) \tag{3}$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/dm}) \tag{4}$$

where PE is Positional Embedding, pos is the position, i is the dimension, and d_m is the dimension of the input and output.

Next, encoders and decoders handle these vectors via multi-head self-attention to learn the attention weights of those representation subspaces and integrate such information from diverse positions. In particular, the masked multi-head self-attention sublayer is masked to avoid overfitting.

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O \tag{5}$$

$$Head_i = Attention(QW_i^{Q,}, KW_i^K, VW_i^V) \tag{6}$$

where the parameter matrices are $W_i^Q \in R^{d_{model}*d_k}$, $W_i^K \in R^{d_{model}*d_v}$, $W_i^V \in R^{d_{model}*d_v}$, and $W_i^O \in R^{d_v*d_{model}}$. A residual connection is built between the input's self-attention sublayer and the point-wise feed-forward network (FFN), preventing the vanishing gradient and the shift of the covariate. Subsequently, the computed results are refined by the normalization sublayer and point-wise feed-forward sublayer.

$$FFN(x) = \sigma\ (max(0, xW_1 + b_1)W_2 + b_2) \tag{7}$$

where FFN is the forward feed network, σ is the ReLU activation, W_1 and W_2 are the slopes, and b_1 and b_2 are the biases. Finally, the data go to a linear layer and a softmax layer to obtain the malicious probability of user activities.

Compared with CNN and RNN, including LSTM and GRU, the transformer utilizes self-attention to interactively compute a weighted sum of all input values to capture the global context of every word in the sequence. The complexity of the computation on each layer is much less. Additionally, the transformer has no recursive or convolutional mechanism, and the lengths of the paths traversing the network between positions in the input and output series are shorter. Thus, it is easier to learn long-term temporal patterns without the gradient disappearance. Moreover, RNN, including LSTM and GRU, must process in sequence so that they cannot utilize the parallel computing capability of the GPU. However, multi-head attention combined with positional embedding, the computation of the transformer can be processed in parallel to take all inputs simultaneously. Therefore, the transformer model has a much better training and inference efficiency and saves considerable training time. Furthermore, the transformer model makes fewer assumptions about the schema information of the data. Tokenization makes transformers a common and extensive architecture for processing heterogeneous input data.

F. CUSTOM TRANSFOMER MODEL:DISTILLEDTRANS

To further improve the detection performance and reduce the training time, this study proposes a customized transformer model named DistilledTrans to detect insider threats.

In this study, the architecture of the transformer model was simplified to maintain only the encoder stack. As shown in Figure 5, a one-dimensional GlobalAveragePooling layer, dropout layer, and fully connected layer with sigmoid activation were added to the top of the encoder stacks to obtain the prediction results. In contrast to some scholar's [33] conclusion, this study found that adding or reducing one encoder/decoder layer or putting a normalization layer before or after the multi-head self-attention mechanism or FFN achieved almost the same performance. However, adding too many layers would degrade the detection performance. Therefore, the number of layers should match the size and complexity of the training dataset. Compared with other models, DistilledTrans exhibited excellent detection performance across different datasets, including datasets augmented by different data augmentation approaches, sporadic datasets, and dense datasets.

G. PRE-TRAINED TRANSFORMER MODELS

When building the transformer models, this study used an embedding layer to convert each integer index into a dense vector containing encoding. Because this embedding layer is task independent, it cannot provide context-dependent word embedding. Thus, it may not be possible to learn sophisticated behavior patterns in a time series. In addition, the size of the CERT dataset may be insufficient to train a complicated insider threat architecture. In contrast, transformer pretrained models are pretrained on much larger datasets and have their own learned contextual embeddings for words. Therefore, to verify that there is room to further improve the detection performance, this study explored the fine-tuning approach on pre-trained transformer models (BERT, RoBERTa, etc.) by adding the final layer on the models to detect insider threats. The final layer is composed of a dropout sublayer, linear sublayer with ReLU activation, and sigmoid sublayer.

As a result of the pretraining process, BERT learns contextual embeddings for words owing to its bidirectional ability. It also saves time and efforts because there is no need to design a special architecture or prior knowledge for different applications. We fine-tuned the BERT-based model with fewer resources on relatively smaller datasets to detect insider threats.

Robustly Optimized BERT Approach (RoBERTa) [32] was developed by Facebook AI. It is also able to learn contextualized word embedding from input sentences using a self-attention structure. It is a variant of the BERT model that has almost the same architecture as BERT's but with some changes: the Next Sentence Prediction (NSP) objective is discarded. It is trained on a large dataset (160GB), including the CC-NEWS, OPENWEBTEXT, and STORIES datasets. In addition, the size of the batch, the length of sequences, and learning rates are increased. Moreover, it dynamically alters the masking strategies rather than static masking in the training to learn more solid and general word embeddings. Furthermore, it adopts a byte-level Byte Paired Encoding (BPE), whose vocabulary is 50 K sub-word units, as a tokenizer to replace the character-level BPE, whose vocabulary is 30 K sub-word units. This study also tuned the RoBERTa model to detect insider threats.

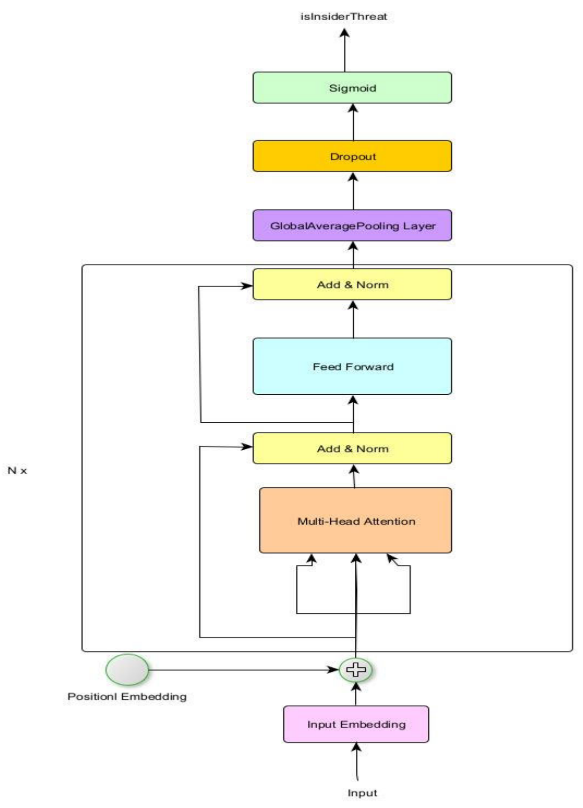

FIGURE 5. The architecture of DistilledTrans model.

H. HYBRID MODELS

Hybrid models using transformer-based models have become an increasingly prevalent approach to accomplish various tasks in different areas. This study also built two hybrid models to perform insider threat detection: combining BERT with CNN and combining BERT with LSTM. In the BERT+ CNN model, BERT is used as the low-level model and connected to the CNN with three convolution kernels, which acts as the high-level model. In the BERT+ LSTM model, BERT works as a low-level model and is connected to LSTM, which acts as a high-level model. BERT is chosen as the low-level model because it showed better performance and higher stability than RoBERTa in insider threat detection. On the other hand, CNN and LSTM are chosen as high-level models because they are among the most popular deep learning models applied in various learning tasks, especially insider threat detection.

IV. EXPERIMENT

A. EXPERIMENT SETTING

Our experiments were accomplished in Google Colab, which is a Virtual Machine with NVIDIA A100-SXM4-40GB GPU, 83.48GB RAM, and 166.77 GB disk. The Pytorch version is 2.0.0, and Python's version is 3.10.11. The main software packages used to develop the models encompass pandas, Matplotlib, transformers, Pandas, Sklearn, and nlpaug. In this study, we trained our model with batch sizes of 32 and 20 epochs and a learning rate of 1e-6.

B. BASELINES

The CERT r4.2 dataset is a dense dataset containing more malicious use cases than the other versions, while CRET r6.2 has the least anomalous users' behavior data. Most researchers worked on CERT r4.2. In this study, experiments were performed on both the datasets. We adopted eleven baseline models on the CERT r4.2 dataset: One Class SVM [20], Hidden Markov model [35], Isolation Forest [36], Decision Tree with under-sampling approach [49], and Hybrid Learning approach [50] are typical applications of machine learning models for insider threat detection. The Deeplog, AutoEncoder, LSTM-AutoEncoder [37], LSTM-based AutoEncoder [38], and LSTM-CNN [24] are representative deep learning models for insider risks. Huang et al. [39] applied the RoBERTa + LSTM model, which is the only study that we found to use a transformer-based model to detect insider threats, but it only ran on CERT r4.2. Our evaluation baselines on the CERT r6.2 dataset include classic machine learning algorithms, such as Isolation Forest and one-class SVM, and deep learning models, such as LSTM-RNN [40], multistate LSTM + CNN [41], Hierarchical LSTMs [25], DeepMIT [42], log2vec, log2vec++ [43] using graphic embeddings, and DD-GCN [51]. However, most of these studies did not provide complete experimental results. Some gave accuracy, some gave AUC, and the rest gave precision, recall, or F1 scores.

C. EVALUATION METRICS

The main purpose of the experiments is to evaluate the performance of the proposed model in detecting insider's abnormal behavior. To evaluate our model, we used the following performance metrics as evaluation criteria:

- **Accuracy**

$$Accuracy = \frac{TP + TN}{TP + TN + NP + NN} \tag{8}$$

where TP, TN, FP, and FN refer to the true positive, true negative, false positive, and false negative, respectively. Accuracy is the percentage of properly classified instances among all instances. This means that out of the diagnosed positive data, how many percentages we classify correctly.

- **Recall**

Recall is also called Detection Rate, Sensitivity or True Positive Rate (TPR)

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

Recall is the proportion of positives correctly classified among the actual positives. This means that out of the positive data, how may percentage we correctly diagnose as positive.

- **Precision**

Precision is also called Confidence

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

Precision is the percentage of actual positives among the predicted positives. That means out of the diagnosed positive data, how many percentages we categorize correctly.

- **F1 Score**

F1 Score is the harmnonic mean of precision and sensitivity.
F-score is defined as the following:

$$F1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

where precision is calculated as $\frac{TP}{TP+FP}$ and recall is defined as $\frac{TP}{TP+FN}$ increasing precision by trading off recall and vice versa.

- **Area under ROC curve (AUC)**

The ROC curve is obtained from the coverage curve by normalizing the axes to the range [1, 0]. The area under the ROC curve (AUC) is the ranking accuracy.

V. EVALUATION AND ANALYSIS

A. EVALUATION RESULTS

On the dataset CERT r4.2, Table 3 shows that BERT + Final layer and RoBERTa + Final layer prove that they are reliable models that outperform all baselines in terms of almost all evaluation metrics except that the precision is slightly lower than that of LSTM-AutoEncoder and Isolation Forest. The original transformer demonstrated excellent performance with an AUC of 97.37%, an accuracy of 93.53%, and a precision of 98.13%, outperforming all the baseline models. However, its recall is 80.71% and thus F1-score is only 88.61% which is lower than that of RoBERTa-LSTM and LSTM-AutoEncoder. In comparison, DistilledTrans not only outperforms all baselines in terms of accuracy, precision, and AUC, but also increases recall to 84.62%; thus, the F1 score increases to 90.53%. However, the recall and F1 score still lag behind slightly with the LSTM-AutoEncoder, although its training time is more than 10 times faster.

As for the CERT r6.2 augmented by contextual word embedding, Table 4 shows that DistilledTrans outperforms other transformer-based models and all baselines in terms of almost all evaluation metrics except that the recall is slightly lower than LSTM-RNN and DeepMIT. The original transformer, BERT + FL, and BERT + CNN also outperform all the baselines in terms of AUC and accuracy. Additionally, BERT+FL and BERT+ CNN performed the best in recall, but their precision and F1-score need improvement. In contrast, the performance of RoBERTa + FL is so poor that it cannot effectively detect anomaly instances after training on this dataset.

As for the CERT r6.2 augmented by contextual embedding sentences, Table 5 shows that DistilledTrans performs the best among all models. The original transformer ranked

TABLE 3. Performance comparison of the models trained by CERT r4.2.

Algorithms	Accuracy	Precision	Recall	F1-Score	AUC
Original Transformer	**93.53%**	**98.13%**	80.77%	88.61%	**97.37%**
DistilledTrans	**94.48%**	**97.35%**	84.62%	90.53%	**97.18%**
BERT + FL	**96.40%**	93.23%	**95.38%**	**96.40%**	**96.12%**
Roberta+ FL	**96.64%**	94.62%	**94.62%**	**94.62%**	**96.09%**
ITDBERT: Roberta + LSTM	X	93.00%	91.87%	92.43%	95.56%
LSTM - CNN	X	X	X	X	94.49%
Deeplog	X	72.59%	90.26%	80.47%	X
AutoEncoder	X	50.18%	94.10%	65.46%	X
LSTM based Autoencoder	90.00%	X	X	X	X
LSTM-AutoEncoder	90.00%	97.00%	X	94.00%	X
Hidden Markov Model	X	X	X	X	83.00%
Isolation Forest	79.00%	95.00%	35.19%	51.35%	X
One class SVM	87.79%	X	X	X	X
DT + Under-sampling	X	X	88.00%	X	94.00%
Hybrid Learning Approach	88.00%	85.70%	84.40%	84.85%	84.00%

NOTE: BOLD DIGITAL INDICATES THAT THE METRIC'S VALUE IS HIGHER THAN ALL BASELINE VALUES. X INDICATES THAT THE PERFORMANCE METRICS WERE NOT PROVIDED.

TABLE 4. Performance comparison of the models trained by CERT r6.2 augmented by contextual embedding words.

Algorithms	Accuracy	Precision	Recall	F1-Score	AUC
Original Transformer	**99.35%**	**100.00%**	76.67%	86.79%	**99.50%**
DistilledTrans	**99.72%**	**100.00%**	90.00%	**94.74%**	**97.52%**
BERT+ FL	**99.35%**	84.85%	**93.33%**	88.89%	**96.42%**
BERT + CNN	**99.08%**	77.78%	**93.33%**	84.85%	**96.29%**
BERT + LSTM	96.58%	44.26%	90.00%	59.34%	93.38%
LSTM-RNN	93.85%	95.12%	92.46%	X	X
multistate LSTM + CNN	85.00%	X	X	X	90.47%
Iforest	X	X	X	X	76.19%
OCSVM	X	X	X	X	73.67%
Hierararchical LSTMs	X	X	X	X	94.96%
DeepMIT	X	91.60%	93.20%	93.20%	X
log2vec	X	X	X	X	86.00%
log2vec++	X	X	X	X	93%
LSTM	X	X	X	X	71.00%
DD-GCN	98.65%	X	X	85.69%	X

NOTE: BOLD DIGITAL INDICATES THAT THE METRIC'S VALUE IS HIGHER THAN ALL BASELINE VALUES. X INDICATES THAT THE PERFORMANCE METRICS WERE NOT PROVIDED.

second. Their metric values are so excellent that they outperform other transformer-based models and all baselines in all metrics, including AUC, F1-score, recall, precision, accuracy, and training time. However, neither BERT + FL nor RoBERTa + FL can detect insider threats. We put CNN or LSTM on BERT to train again; they showed some improvement, but their performance, especially recall, F1 score, and AUC, still could not compete with DistilledTrans and Original Transformer.

B. COMPARISON ANALYSIS

From Tables 3, 4, and 5, we can see that whether trained on dense dataset CERT r4.2, or sporadic dataset CERT r6.2 augmented by contextual word embedding and contextual sentence, both the original Transformer and DistilledTrans performed very well. They take the lead of other models in almost all performance metrics, meanwhile their structure is more concise, and the training time is much shorter. The self-attention mechanism, positional embedding,

TABLE 5. Performance comparison of the models trained by cert r6.2 augmented by contextual embedding sentence.

Algorithms	Accuracy	Precision	Recall	F1-Score	AUC
Original Transformer	**99.82%**	**100.00%**	**93.33%**	**96.55%**	**98.52%**
DistilledTrans	**99.82%**	**100.00%**	**93.33%**	**96.55%**	**99.63%**
BERT + CNN	97.97%	**100.00%**	26.67%	42.11%	63.33%
BERT + LSTM	**98.80%**	**100.00%**	56.67%	72.34%	78.33%
LSTM-RNN	93.85%	95.12%	92.46%	X	X
multistate LSTM + CNN	85.00%	X	X	X	90.47%
Iforest	X	X	X	X	76.19%
OCSVM	X	X	X	X	73.67%
Hierararchical LSTMs	X	X	X	X	94.96%
DeepMIT	X	91.60%	93.20%	93.20%	X
log2vec	X	X	X	X	86.00%
log2vec++	X	X	X	X	93%
LSTM	X	X	X	X	71.00%
DD-GCN	98.65%	X	X	85.69%	X

NOTE: BOLD DIGITAL INDICATES THAT THE METRIC'S VALUE IS HIGHER THAN ALL BASELINE VALUES. X INDICATES THAT THE PERFORMANCE METRICS WERE NOT PROVIDED.

and multiple heads enable them have strong ability to learn long temporal relationships, compute in parallel, and easily integrate data without knowing the data schema. Contextual-independent word embedding results in excellent extensivity when trained on different datasets, including augmented data sets. Compared with the original transformer, DistilledTrans performs better because it only utilizes encoder blocks, and thus avoids the over-processing of information. This also demonstrates that our data augmentation methods are feasible for training an excellent model for detecting insider threats.

For the dense dataset CERT r4.2, fine-tuning the pre-trained PERT or RoBERTa plus a final layer can further improve recall with a slight sacrifice of precision. Because they are pre-trained on a large dataset, we can take advantage of the pre-trained knowledge to perform the downstream task of insider threat detection. As for the sporadic dataset CERT r6.2 augmented by contextual word embedding, BERT plus a final layer can further improve recall with a slight decrease in precision owing to pre-trained word embedding and model structure. In contrast, the performance of BERT + CNN and BERT+ LSTM is worse, especially in terms of precision and F1-score. This proves that the self-attention structure is a substitution for convolutional and recursive mechanisms rather than a supplement. Putting another neural network on BERT makes the structure too deep and complex so that information would be lost, thus damaging the performance. Therefore, a hybrid model may not be a suitable choice for insider threat detection.

For the sparse dataset CERT r6.2, augmented by contextual embedding sentences, both the original transformer and DistilledTrans perform very well. In contrast, neither BERT + CNN nor BERT+ LSTM perform satisfactorily. Their recalls are much worse than their performance in CERT r6.2 augmented by the contextual word embedding method. BERT plus a final layer cannot detect insider threats at all. This is because the dataset was augmented by another model, GPT-2. Thus, BERT's pretrained knowledge, contextual embedding, and network structure are no longer effective.

Even if the neural network is added to the BERT model, the model's performance still cannot match that of two transformers. In addition, RoBERTa plus a final layer cannot detect insider risks after training using both augmented CERT r6.2 datasets. This is because its pretrained knowledge, contextual embedding, or network structure mismatch with the insider threat detection task and the input of the augmented dataset. For instance, RoBERTa uses a large byte-level BPE vocabulary to build its own contextual-dependent tokens, so it may not be able to understand our input at all. Moreover, RoBERTa removed the NSP loss. This could also weaken its performance.

VI. CONCLUSION

In this paper, we proposed DTITD, an innovative intelligent insider threat detection framework based on digital twins and self-attention based deep learning models. This solution is not only able to timely and continuously monitor the insider risk profile of an organization but also easily perform in-depth analysis to find root causes and quickly take appropriate remediation actions against insider threats. We then perform novel language data augmentation approaches, including contextual word embedding insert and substitution predicted by the BERT model and context embedding sentence predicted by the GPT-2 model, to overcome the high data imbalance of the sporadic dataset. Our experiment demonstrates that the context embedding sentences predicted by the GPT-2 model outperform the contextual word embedding predicted by the BERT model. Next, this paper presents a custom transformer model named DistilledTrans and conducts extensive experiments to compare this model with the original transformer model, pre-trained transformer model (BERT+ Final layer, RoBERTa + Final layer), hybrid models (BERT + CNN, BERT + LSTM), and the state-of-the-art models. Our experimental results show that

1) When training models with the sporadic dataset, DistilledTrans trained with the dataset augmented by contextual embedding sentences performs the best in terms of all evaluation metrics, including accuracy, precision, recall, F1-score, and AUC. In addition, this model is lightweight and can significantly reduce the training time and costs.

2) When training models with the dense dataset, pre-trained models BERT plus a final layer or RoBERTa plus a final layer can achieve significantly higher performance than all current models, including various hybrid models and two transformer models, with very little sacrifice of precision. Additionally, these models are much more concise and simpler than hybrid models.

Considering that there are email and website contents in the CERT dataset that have not been utilized yet, our future work would be to perform sentimental analytics on the contents and take advantage of user profile data to provide early alarms. Combined with this study that we have already done, it will compose a complete, highly accurate, and timely insider risk solution at the enterprise level. We will also research efficient transfer learning for various learning tasks or different user behavior representations to further elevate the efficiency of insider threat detection and protect privacy.

ACKNOWLEDGMENT

The author would like to thank his supervisor Prof. Abdulmotaleb El Saddik for his guidance, encouragement, valuable comments, and support during his studies. He also like to thank all MCRLab mates for the good times he had worked in such a friendly environment, especially for Haopeng Wang's comments and encouragement.

REFERENCES

[1] *Insider Threat Report 2019*, CA Technol., San Jose, CA, USA, 2019.

[2] "2018 U.S. state of Bercrime," CSO, U.S. State of Cybercrime D. of SRI-CMU, and ForcePoint, Tech. Rep., 2018.

[3] (2022). *Market Guide for Insider Risk Management Solutions, Jonathan Care, Paul Furtado, Brent Predovich, USA*. Accessed: Mar. 5, 2022. [Online]. Available: https://www.gartner.com/document/4013691?ref=solrAll&refval=363320574

[4] (2020). *Market Guide for Insider Risk Management Solutions, Jonathan Care, Brent Predovich, Paul Furtadoh, USA*. Accessed: Mar. 5, 2022. [Online]. Available: https://www.gartner.com/document/3994931?ref=solrAll&refval=363319695

[5] (2020). *Insider Risk Report, DTEX Systems, USA*. Accessed: Mar. 5, 2022. [Online]. Available: https://www2.dtexsystems.com/2022-insider-risk-report

[6] N. Nguyen, P. Reiher, and G. H. Kuenning, "Detecting insider threats by monitoring system call activity," in *Proc. IEEE Syst., Man Cybern. SocietyInformation Assurance Workshop*, Jun. 2003, pp. 45–52.

[7] M. Hanley and J. Montelibano, "Insider threat control: Using centralized logging to detect data exfiltration near insider termination," DTIC, Fort Belvoir, VA, USA, Tech. Rep. 24, 2011.

[8] M. A. Maloof and G. D. Stephens, "ELICIT: A system for detecting siders who violate need-to-know," in *Proc. Int. Workshop Recent Adv. Intrusion Detect.*, 2007, pp. 146–166.

[9] R. A. Maxion and T. N. Townsend, "Masquerade detection using truncated command lines," in *Proc. Int. Conf. Dependable Syst. Netw.*, Jun. 2002, pp. 219–228.

[10] M. B. Salem and S. J. Stolfo, "Detecting masqueraders: A comparison of one-class bag-of-words user behavior modeling techniques," *J. Wireless Mobile Netw. Ubiquitous Comput. Depend. Appl.*, vol. 1, no. 1, pp. 3–13, 2010.

[11] M. B. Salem and S. J. Stolfo, "A comparison of one-class bag-of-words user behavior modeling techniques for masquerade detection," *Secur. Commun. Netw.*, vol. 5, no. 8, pp. 863–872, Aug. 2012.

[12] P. Kudłacik, P. Porwik, and T. Wesołowski, "Fuzzy approach for intrusion detection based on user's commands," *Soft Comput.*, vol. 20, no. 7, pp. 2705–2719, Jul. 2016.

[13] Y. Song, M. B. Salem, S. Hershkop, and S. J. Stolfo, "System level user behavior biometrics using Fisher features and Gaussian mixture models," in *Proc. IEEE Secur. Privacy Workshops*, May 2013, pp. 52–59.

[14] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "Disclosure: Detecting botnet command and control servers through large-scale NetFlow analysis," in *Proc. 28th Annu. Comput. Secur. Appl. Conf.*, Dec. 2012, pp. 129–138.

[15] W. T. Young, H. G. Goldberg, A. Memory, J. F. Sartain, and T. E. Senator, "Use of domain knowledge to detect insider threats in computer activities," in *Proc. IEEE Secur. Privacy Workshops*, May 2013, pp. 60–67.

[16] T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels, and E. Kirda, "Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks," in *Proc. 29th Annu. Comput. Secur. Appl. Conf.*, Dec. 2013, pp. 199–208.

[17] O. Brdiczka, J. Liu, B. Price, J. Shen, A. Patil, R. Chow, E. Bart, and N. Ducheneaut, "Proactive insider threat detection through graph learning and psychological context," in *Proc. IEEE Symp. Secur. Privacy Workshops*, May 2012, pp. 142–149.

[18] T. E. Senator, "Detecting insider threats in a real corporate database of computer usage activity," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2013, pp. 1393–1401.

[19] A. Gamachchi, L. Sun, and S. Boztas, "Graph based framework for licious insider threat detection," in *Proc. 50th Hawaii Int. Conf. St. Sci.*, 2017, pp. 2638–2647.
[20] L. Lin, S. Zhong, C. Jia, and K. Chen, "Insider threat detection based on deep belief network feature representation," in *Proc. Int. Conf. Green Informat. (ICGI)*, Aug. 2017, pp. 54–59, doi: 10.1109/ICGI.2017.37.
[21] L. Liu, O. De Vel, C. Chen, J. Zhang, and Y. Xiang, "Anomaly-based insider threat detection using deep autoencoders," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2018, pp. 39–48, doi: 10.1109/ICDMW.2018.00014.
[22] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," in *Proc. AI Cyber Secur. Workshop (AAAI)*, 2017.
[23] J. Lu and R. K. Wong, "Insider threat detection with long short-term memory," in *Proc. Australas. Comput. Sci. Week Multiconf.*, Jan. 2019, pp. 1–10, doi: 10.1145/3290688.3290692.
[24] F. Yuan, Y. Cao, Y. Shang, Y. Liu, J. Tan, and B. Fang, "Insider threat detection with deep neural network," in *Computational Science ICCS 2018* (Lecture Notes in Computer Science), Y. Shi, H. Fu, Y. Tian, V. V. Krzhizhanovskaya, M. H. Lees, J. Dongarra, and P. M. A. Sloot, Eds. Cham, Switzerland: Springer, 2018, pp. 43–54.
[25] S. Yuan, P. Zheng, X. Wu, and Q. Li, "Insider threat detection via hierarchical neural temporal point processes," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 1343–1350.
[26] A. Saaudi, Z. Al-Ibadi, Y. Tong, and C. Farkas, "Insider threats detection using CNN-LSTM model," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Dec. 2018, pp. 94–99, doi: 10.1109/CSCI46756.2018.00025.
[27] J. Jiang, J. Chen, T. Gu, K.-K.-R. Choo, C. Liu, M. Yu, W. Huang, and P. Mohapatra, "Anomaly detection with graph convolutional networks for insider threat and fraud detection," in *Proc. MILCOM IEEE Mil. Commun. Conf. (MILCOM)*, Nov. 2019, pp. 109–114, doi: 10.1109/MILCOM47813.2019.9020760.
[28] M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems," in *Transdisciplinary Perspectives On Complex Systems: New Findings and Approaches*, 2017, pp. 85–113.
[29] A. El Saddik, "Digital twins: The convergence of multimedia technologies," *IEEE Multimedia Mag.*, vol. 25, no. 2, pp. 87–92, Apr. 2018.
[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017.
[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.
[32] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*.
[33] Z. Wu, H. Zhang, P. Wang, and Z. Sun, "RTIDS: A robust transformer-based approach for intrusion detection system," *IEEE Access*, vol. 10, pp. 64375–64387, 2022.
[34] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," Tech. Rep., 2018.
[35] T. Rashid, I. Agrafiotis, and J. R. C. Nurse, "A new take on detecting insider threats: Exploring the use of hidden Markov models," in *Proc. 8th ACM CCS Int. Workshop Manag. Insider Secur. Threats*, Oct. 2016, pp. 47–56.
[36] B. Lv, D. Wang, Y. Wang, Q. Lv, and D. Lu, "A hybrid model based on multi-dimensional features for insider threat detection," in *Wireless Algorithms, Systems, and Applications*, vol. 13. Tianjin, China: Springer, Jun. 2018, pp. 333–344.
[37] R. Nasir, M. Afzal, R. Latif, and W. Iqbal, "Behavioral based insider threat detection using deep learning," *IEEE Access*, vol. 9, pp. 143266–143274, 2021.
[38] B. Sharma, P. Pokharel, and B. Joshi, "User behavior analytics for anomaly detection using LSTM autoencoder–insider threat detection," in *Proc. 11th Int. Conf. Adv. Inf. Technol.*, Jul. 2020, pp. 1–9.
[39] W. Huang, H. Zhu, C. Li, Q. Lv, Y. Wang, and H. Yang, "ITDBERT: Temporal-semantic representation for insider threat detection," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Sep. 2021, pp. 1–7.
[40] F. Meng, F. Lou, Y. Fu, and Z. Tian, "Deep learning based attribute classification insider threat detection for data security," in *Proc. IEEE 3rd Int. Conf. Data Sci. Cyberspace (DSC)*, Jun. 2018, pp. 576–581.
[41] M. Singh, B. M. Mehtre, and S. Sangeetha, "User behavior profiling using ensemble approach for insider threat detection," in *Proc. IEEE 5th Int. Conf. Identity, Secur., Behav. Anal. (ISBA)*, Jan. 2019, pp. 1–8.
[42] D. Sun, M. Liu, M. Li, Z. Shi, P. Liu, and X. Wang, "DeepMIT: A novel malicious insider threat detection framework based on recurrent neural network," in *Proc. IEEE 24th Int. Conf. Comput. Supported Cooperat. Work Design (CSCWD)*, May 2021, pp. 335–341.
[43] F. Liu, Y. Wen, D. Zhang, X. Jiang, X. Xing, and D. Meng, "Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 1777–1794.
[44] T. Elhassan and M. Aljurf, "Classification of imbalance data using tomek link (T-link) combined with random under-sampling (RUS) as a data reduction method," *Global J. Technol. Optim.*, vol. 1, no. S1, p. 2016, 2016.
[45] M. A. Haq, M. A. R. Khan, and M. Alshehri, "Insider threat detection based on NLP word embedding and machine learning," *Intell. Autom. Soft Comput.*, vol. 33, no. 1, pp. 619–635, 2022.
[46] O. A. Alzubi, J. A. Alzubi, A. M. Al-Zoubi, M. A. Hassonah, and U. Kose, "An efficient malware detection approach with feature weighting based on Harris hawks optimization," *Cluster Comput.*, vol. 25, no. 4, pp. 2369–2387, Aug. 2022.
[47] O. A. Alzubi, J. A. Alzubi, M. Alazab, A. Alrabea, A. Awajan, and I. Qiqieh, "Optimized machine learning-based intrusion detection system for fog and edge computing environment," *Electronics*, vol. 11, no. 19, p. 3007, Sep. 2022.
[48] O. A. Alzubi, I. Qiqieh, and J. A. Alzubi, "Fusion of deep learning based cyberattack detection and classification model for intelligent systems," *Cluster Comput.*, vol. 26, no. 2, pp. 1363–1374, Apr. 2023.
[49] T. Al-Shehari and R. A. Alsowail, "Random resampling algorithms for addressing the imbalanced dataset classes in insider threat detection," *Int. J. Inf. Secur.*, vol. 22, no. 3, pp. 611–629, Jun. 2023.
[50] M. Singh, B. M. Mehtre, S. Sangeetha, and V. Govindaraju, "User behaviour based insider threat detection using a hybrid learning approach," *J. Ambient Intell. Humanized Comput.*, vol. 14, no. 4, pp. 4573–4593, Apr. 2023.
[51] X. Li, X. Li, J. Jia, L. Li, J. Yuan, Y. Gao, and S. Yu, "A high accuracy and adaptive anomaly detection model with dual-domain graph convolutional network for insider threat detection," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 1638–1652, 2023.

ZHI QIANG WANG (Member, IEEE) received the B.C.S. degree from the Anhui University of Technology, and the M.C.S. degree from Southeast University. He is currently pursuing the Ph.D. degree in computer science with the School of Electrical and Computer Engineering, University of Ottawa. He has played important roles in writing the book "*Digital Twin for Healthcare: Design, Challenges, and Solutions*" (Elsevier), in 2022. His research interests include cybersecurity, deep fraud, digital twins, and mixed reality.

ABDULMOTALEB EL SADDIK (Fellow, IEEE) is currently a Distinguished Professor and the University Research Chair with the School of Electronical and Engineering and the Computer Science, University of Ottawa, and MBZUAI. He is an ACM Distinguished Scientist. He has authored and coauthored ten books and more than 600 publications and chaired more than 50 conferences and workshops. He received research grants and contracts totaling more than U.S. $20 million. His research interests include digital twin, AR/VR, and tactile internet. He has supervised more than 120 researchers and received several international awards. He is a fellow of the Royal Society of Canada, the Engineering Institute of Canada, and the Canadian Academy of Engineers. He received the IEEE I&M Technical Achievement Award, the IEEE Canada C.C. Gotlieb (Computer) Medal, and the A.G.L. McNaughton Gold Medal for important contributions to the field of computer engineering and science.