IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 22 August 2024, accepted 24 September 2024, date of publication 17 October 2024, date of current version 11 November 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3482988

RESEARCH ARTICLE

FedITD: A Federated Parameter-Efficient Tuning With Pre-Trained Large Language Models and Transfer Learning Framework for Insider Threat Detection

ZHI QIANG WANG[1], (Member, IEEE), HAOPENG 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 Insider threats cause greater losses than external attacks, prompting organizations to invest in detection systems. However, there exist challenges: 1) Security and privacy concerns prevent data sharing, making it difficult to train robust models and identify new attacks. 2) The diversity and uniqueness of organizations require localized models, as a universal solution could be more effective. 3) High resource costs, delays, and data security concerns complicate building effective detection systems. This paper introduces FedITD, a flexible, hierarchy, and federated framework with local real-time detection systems, combining Large Language Models (LLM), Federated Learning (FL), Parameter Efficient Tuning (PETuning), and Transfer Learning (TF) for insider threat detection. FedITD uses FL to protect privacy while indirect integrating client information and employs PETuning methods (Adapter, BitFit, LoRA) with LLMs (BERT, RoBERTa, XLNet, DistilBERT) to reduce resource use and time delay. FedITD customizes client models and optimizes performance via transfer learning without central data transfer, further enhancing the detection of new attacks. FedITD outperforms other federated learning methods and its performance is very close to the best centrally trained method. Extensive experiment results show FedITD's superior performance, adaptability to varied data, and reduction of resource costs, achieving an optimal balance in detection capabilities across source data, unlabeled local data, and global data. Alternative PETuning implementations are also explored in this paper.

INDEX TERMS Cybersecurity, insider threat, deep learning, transformer, BERT, RoBERTa, XLNet, DistilBERT, GPT, data augmentation, artificial intelligence, machine learning, pre-trained LLM, PETuning, adapter, LoRA, BitFit, LLM, NLP.

I. INTRODUCTION

According to 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.'' [1]

When it comes to malicious actions by insiders, insider threats can be categorized into three main groups:

1. Fraud involves unauthorized actions such as inserting, deleting, or altering an organization's data.
2. Data theft encompasses unauthorized access to and exfiltration of data, such as intellectual property embezzlement from the organization.
3. System sabotage involves the direct use of information technology to disrupt or harm an organization's system,

The associate editor coordinating the review of this manuscript and approving it for publication was Jenny Mahoney.

such as actions that compromise data integrity or disrupt service availability.

In contrast, External Intrusion refers to unauthorized access or attempts to access a computer system, network, or data from outside an organization's perimeter. Examples of external intrusion encompass a hacker using malware to infiltrate a company's network, phishing attacks aimed at stealing login credentials, or Distributed Denial of Service (DDoS) attacks to disrupt services.

Insider Threat Detection Systems (ITDS) and Intrusion/Extrusion Detection Systems (IDS/EDS) focus on different types of threats and use distinct approaches:

- **Source of Threats**: ITDS detects threats from insiders with legitimate access, while IDS/EDS targets external threats like hackers and malware.
- **Detection Method**: ITDS often uses deep learning and user behavior analytics to identify subtle deviations in insider activity, whereas IDS/EDS focuses on known attack signatures or unusual network traffic or system patterns to detect external threats.
- **Data Sources and Monitoring**: ITDS monitors a wide range of user activities and logs within the organization, while IDS/EDS primarily analyzes network traffic and system logs for external intrusion signs.
- **Response and Prevention**: ITDS involves detailed investigations, automated responses, and preventive measures like user education and access controls. IDS/EDS emphasizes immediate alerts, system patching, and securing systems against exploitation.

Due to the rising prevalence of networks and the swift evolution of modern hacker tactics, an escalating number of organizations are encountering cybersecurity risks stemming from insiders rather than external hackers. Insider users often have legal privileges in accessing an organization's system and thus it often causes huge losses compared with external attacks. Preventing insider threats requires a multifaceted approach combining technical, procedural, and cultural measures. Key strategies include 1) Implementing strict access controls, such as the Principle of Least Privilege and Role-Based Access Control; 2) Monitoring user activities with software tools to detect anomaly behavior; 3) Security awareness training and clear communication of internal policies help educate employees;4)Background checks and Employee Assistance Programs (EAPs) address psychological and behavioral risks; 5) A well-defined incident response plan; 6) Regular audits of access controls and security measures are essential for early detection and response to potential threats. This paper focuses on insider threat detection. However, how to timely and efficiently detect insider threats faces the following challenges:

1. **Poor Detection Performance:** Compared with external intrusions, insider threats are often much harder to detect. Insiders often have legitimate privileges within an organization's system, making it easier for them to bypass security measures. Thus malicious activities are hidden within a large number of normal user behaviors and might appear legitimate or routine. Insider threat models have to identify subtle anomalies that are small deviations from normal user behavior rather than obvious anomalies which are often unusual patterns in network traffic or system behavior suggesting an external attack.
2. **Inefficient Detection Model:** Insider threat detection often requires a deeper understanding of internal users' and entities' behaviors and related correlations. It requires considering the context of user actions, such as time of access, action temporal patterns, and relation to other users and entities. These patterns and correlations are so complex, deep, and hidden that it requires more powerful models to handle them. For example, most existing modeling methods focus on behavior category features such as various frequencies or simple sequences but ignore temporal patterns. Although some models attempt to capture temporal relationships, they are unable to handle long temporal sequences. The remaining models often incorporate multiple deep learning models, leading to excessively complex, resource-intensive, and time-consuming training.
3. **Highly Imbalanced Datasets:** due to isolated data islands and concerns about security and privacy, organizations are more reluctant to share insider threat data, resulting in a lack of anomaly data to effectively train robust insider threat detection models. Clients often have a large number of records of normal user behaviors but few available anomalous data instances. This leads to a biased model with notably lower sensitivity and a high false positive rate. Additionally, it is difficult for them to detect new or unknown threats because these threats never happen locally.
4. **Lack of Personalized Model:** the existing intrusion detection systems lack the capability of personalization. Most methods rely on a global model for nearly all clients. Once enough client data is gathered to train a satisfactory global model, it is for all clients to detect intrusion. However, various organizations may have different user daily behavior patterns and face different security threats. They are also concerned about their data privacy and security. Therefore, a global model is not practical and custom models should be considered for insider threats.
5. **Long Latency and High Bandwidth Usage:** traditional intrusion detection methods often involve aggregating data in a central location for analysis, leading to long latency and high bandwidth usage. However, customers increasingly demand local real-time detection systems with locally trained models and prompt alert generation for insider threats.

Addressing the existing problems that insider threat detection faces requires a combination of multiple advanced technologies: To improve poor detection performance and

accurately identify anomalies, we adopt pre-trained Large Language Models (LLMs). To address the inefficient model, we use federated Parameter-Efficient Tuning (PETuning) methods to tune models. To break the barriers between isolated data islands and detect unknown attacks, we utilize Federated Learning (FL) and transfer learning to learn from other clients. To overcome highly imbalanced dataset issues, Natural Language Processing (NLP) data augmentation methods are applied to upsample minorities. To handle lacking personalized models and labeled data, we incorporate the Transfer Learning (TL) unsupervised domain adaption method. To solve long latency and high resource costs, we design a flexible, hierarchical, and federated framework with local real-time detection systems, locally trained models, and federated PETuning methods. Therefore, we propose a novel system called FedITD combining pre-trained LLMs, PETuning, federated learning, and transfer learning.

To overcome this obstacle of data integration, FL has surfaced as a privacy-conscious technique. FL aims to collectively train models without the need to transmit substantial client data to a centralized location. In FL, decentralized clients only simply compute and send a small part of model information, like parameters or gradients, to a server at intervals, where they are then aggregated to generate a global model. It disrupts data islands by enabling learning across distributed data sources. It accelerates training and reduces communication requirements compared to centralized learning methods. Additionally, it facilitates collaboration without compromising the security of each participant by maintaining data locally. Thus, FL is chosen as our system's foundation.

Pre-trained LLMs like BERT [2], RoBERTa [3], XLNet [4], and DistllBert [5] have showcased remarkable performance across various natural language processing (NLP) tasks, such as GLUE [6]. Wang et al. [7] demonstrated exceptional insider threat detection performance by centrally fine full tuning the pPe-trained LLMs on the CERT dataset. Therefore, we adopt Pre-trained LLMs as the backbone models of our FL system. However, LLMs introduce challenges such as significant communication overhead, storage cost, and high costs associated with adapting local models after FL and security concerns.

PETuning methods, including Adapter [8], LoRA [9], and BitFit [10], typically involve locking most of the parameters in Pre-trained LLMs and only tuning a small subset of extra parameters or a small portion of the initial model parameters for downstream jobs. These methods were originally trained centrally. We introduce them to FL to effectively reduce communication costs and resource usage while maintaining satisfactory performance. It also can significantly improve security e.g., counter gradient inversion attacks, which can result in an average reduction of 40.7% in the accuracy of recovered words compared to Federated Parameter Full Tuning [11].

However, the global model may not be a good fit for a specific client due to local data heterogeneity and diversity. Using TL, this study can rapidly develop a custom local model tailored to specific organizations or cases, leveraging insights gained from previous experiences shared by other organizations. The central concept is to minimize the distribution divergence between various organizations. It significantly enhances adaptability because this approach could offer multiple variations of the same global models customized for different organizations or use cases with their limited local data, thus yielding superior local results. We combine federated PETuing and TL to address the aforementioned challenges and further improve unknown attack detection.

To sum up, this paper contributes in the following ways:

1) **Novel Framework**: To the best of our knowledge, FedITD is the first framework to leverage the FL of transformer variant models, PETuning, and TF for insider threat detection. This paper proposed a flexible, hierarchical, and federated framework with local real-time detection systems, locally trained models, and federated PETuning methods, effectively leading to low latency and prompt alerts. We also explored alternative PETuning implementations like Adapter and LoRA.
2) **Excellent Detection Performance**: FedITD outperforms other federated learning methods and almost all central training-based insider threat detection methods. It closely matches the best centrally trained method, DistilledTrans. Additionaly, it demonstrated FedITD's strong adaptability to both highly and slightly heterogeneous data and effectiveness in detecting new or unknown attacks by indirect learning from other clients without compromising privacy.
3) **High Resource Efficiency:** Demonstrated significant reductions in communication, memory, and storage costs using Federated PETuning compared to full-tuned original LLMs.
4) **Optimal Equilibrium**: Explored the system configuration to attain the optimal TL performance across source data domain, unlabeled local data domain, and global data, which other methods lack.

II. RELATED WORK

Initially, researchers employed rule-based methods combined with signature matching to identify insider threats [12], [13], [14]. However, these approaches required extensive domain knowledge to perform thorough analysis and recognize threat signatures. The response to insider risks was relatively slow, as these methods were often reactive, addressing issues only after damage had occurred. With the rapid emergence of new insider threats, even minor variations of known threats could go undetected if their signatures were not pre-identified.

Over the past decade, similar to the work of Sandikkaya et al. [15] on the use of machine learning models in cloud web application security, many researchers have explored the application of machine learning models for detecting insider threats. Maxion and Townsend [16] utilized naive classification to assign posterior probabilities to test commands based

on historical user data. Salem and Stolfo [17], [18] introduced a one-class Support Vector Machine (SVM) trained to detect anomalous command sequences by analyzing frequency patterns. Kudłacik et al. [19] presented a solution for building a fuzzy user profile based on the frequency vector of a user command. Song et al. [20] experimented with Gaussian Mixture Models (GMM), SVMs, and Kernel Density Estimation (KDE), ultimately finding that GMM performed best. Gamachchi et al. [21] used a bipartite graph approach to model user interactions with host-based data from logs, file systems, and psychological surveys, employing an Isolation Forest (IF) to identify malicious users. After performing complex feature engineering to create many enhanced features, such as frequencies under different time frames that the original dataset does not have, Le et al. [22] employed machine learning algorithms including Logistic Regression, Neural Network, Random Forest, and XGBoost to detect insider threat. Random Forest performs best at a detection rate of 62%, precision of 5.96%, and F1 score of 10.10% on the CERT r6.2 dataset. They also could not detect scenario 5. Al-Shehari and Alsowail [23] applied various sampling techniques, including under-sampling, over-sampling, and hybrid sampling, to address class imbalance in the CERT r4.2 dataset, then trained several machine learning models, such as Extreme Gradient Boosting (XGB), Decision Trees (DT), Random Forest (RF), and K-Nearest Neighbors (KNN), to detect insider data leakage. Brdiczka et al. [24] utilized Structural Anomaly Detection (SAD) to identify anomalies in social networks, combining user connections and psychological features to generate an anomaly score for each user.

However, these machine learning-based model structures are too shallow to capture complex, hidden, and nonlinear internal user behavior patterns, resulting in high false positive rates and low detection accuracy. They also rely heavily on manual feature engineering, which is time-consuming and demands specialized expertise. As a result, there has been a growing shift towards deep learning models for addressing insider threats. Deep learning models offer several advantages over traditional machine learning approaches. They are adept at identifying deep and significant nonlinear correlations, making them well-suited for understanding complex user behaviors and accurately identifying anomalous activities. Additionally, deep learning models can seamlessly integrate heterogeneous data sources from multiple domains. They also reduce the need for extensive feature engineering, requiring less prior knowledge and human efforts. Furthermore, some unsupervised deep learning methods possess strong learning capabilities without the need for labeled data.

Deep learning has been introduced in various applications, i.e., flight control [25], virtual decision-making [26], etc., for many years. Lin et al. [27] proposed an unsupervised hybrid Deep Belief Network (DBN) model for detecting insider threats. This approach involved stacking multilayer Restricted Boltzmann Machines (RBMs) and feeding the resulting output into a One-Class Support Vector Machine (OCSVM) for threat detection. Liu et al. [28] utilized four autoencoders, each designed to detect anomalies in different datasets. The outputs were then combined into an ensemble model to identify the top N malicious insiders. These feedforward neural network (FNN) approaches can enhance accuracy and reduce false alarms. However, most features used in these models are simple aggregated counts of activities, focusing on frequency, which limits their ability to capture temporal patterns. Moreover, late data fusion misses the opportunity to model correlations across different data domains early.

Recurrent Neural Networks (RNNs) often struggle with long-term temporal dependencies, but Long Short-Term Memory networks (LSTMs) [29], [30], [31], [32], [33] and Gated Recurrent Units (GRUs) provide solutions to this problem by maintaining information over extended sequences. These models use user activity sequences as input to generate anomaly scores for prediction. While LSTMs and GRUs can effectively capture temporal dependencies, their complex architectures and lack of parallel processing capabilities result in slower training compared to traditional RNNs. Although these models can identify suspicious behaviors, they cannot always guarantee that the behaviors represent true threats.

Saaudi et al. [34] proposed a combined solution using Convolutional Neural Networks (CNNs) and LSTMs, where CNNs extract features and LSTMs model temporal patterns. Their experimental results demonstrated that the CNN-LSTM model outperformed standalone CNN and LSTM models, effectively capturing temporal patterns in data. Additionally, this approach does not require manual feature engineering, allowing for automatic and faster feature processing and learning. However, the CNN-based model can sometimes miss valuable information due to the pooling layer and typically requires a large labeled dataset for training.

Graph Convolutional Networks (GCNs) [35], [36] offer another approach by using a graph structure to model the relationships between users and their attributes. The graph, along with other contextual data, is input into a CNN to predict anomalous users and their communities. While this method can model complex correlations between users and their behaviors, it is challenging to implement, requiring substantial expert knowledge and effort to construct the graph.

Transformers and their derived models have garnered significant interest in recent times due to their effectiveness and versatility. A transformer model, a deep learning architecture rooted in the multi-head attention mechanism and developed by Google, was introduced in 2017 [37]. It employs self-attention to dynamically calculate a weighted sum of all input values, enabling it to capture the global context of each word in a sequence. Unlike traditional deep learning models, the transformer does not employ recursive or convolutional mechanisms, resulting in shorter paths between input and output positions within the network. This characteristic facilitates learning long-term temporal patterns without encountering the issue of gradient disappearance.

Additionally, the combination of multi-head attention and positional embedding enables parallel processing of all inputs, enhancing both training and inference efficiency and significantly reducing training time. Moreover, the transformer model requires fewer assumptions about the data's schema information, making it a versatile architecture for processing diverse input data types. Tokenization further contributes to the widespread adoption of transformers for processing heterogeneous input data.

Transformer models can undergo pre-training on extensive datasets and then be fine-tuned for various downstream tasks. Wang et al [7] integrate Digital Twin technology and self-attention mechanisms to analyze user behavior and entities in 2023, utilizing advanced data augmentation techniques such as contextual word embedding using the BERT model and contextual sentence embedding using the GPT-2 model to address data imbalances. The paper proposes DistilledTrans, a simplified transformer model, and evaluates its performance alongside other transformer models and hybrid models, including BERT, RoBERTa, BERT+ CNN, RoBERTa + CNN, BERT + LSTM, and RoBERTa + LSTM networks. Experimental results demonstrate the superior performance of DistilledTrans on sporadic datasets CERT r6.2. In contrast, pre-trained models like BERT and RoBERTa exhibit high performance on dense datasets CERT r4.2, suggesting that complex hybrid methods combining transformer and other deep learning models are not necessary. Overall, the proposed framework demonstrates significant advancements in insider threat detection with improved accuracy, efficiency, and interpretability compared to existing models. However, that solution is based on centralized learning and does not involve FL, PETuning, or TL.

In 2016, Google pioneered the introduction of Federated Learning (FL) [38] to facilitate efficient learning across multiple participants while ensuring data privacy and security during data exchange. Over the past few years, FL has experienced rapid growth across various industries and applications. In FL, each participant conducts local training using private data and subsequently uploads the gradient or weight update to the server. These updates from all users are then aggregated to refine the global model, which is subsequently distributed back to all clients for the next iteration. FL effectively solves isolated data island issues to construct a robust global model while safeguarding personal data privacy and information security. However, FL encounters challenges such as high communication costs and diverse client data distributions. FedSGD is introduced alongside the FL concept. It entails the computation of a weighted average of the local gradient. While Stochastic Gradient Descent (SGD) serves as a commonly employed optimization function in Neural Networks (NNs), the process of aggregating gradients at each epoch results in considerable costs concerning bandwidth consumption and computing power. To tackle this challenge, FedAvg is proposed to aggregate model parameters rather than gradients, thereby permitting each client to train the model locally and share updates after several epochs. Our study adopts this approach.

In the realm of insider threat detection, Amiri-Zarandi et al. [39] introduce a federated learning system that employs the AutoEncoder model to improve privacy and Shapley value to enhance explainability. However, it lacks implementation details and in-depth analysis. It neglects the requirements for localizing models meanwhile maintaining global performance, reducing communication cost and storage overhead, overcoming data imbalance, and detecting new or unknown attacks. The solution is not an end-to-end solution but needs human efforts to investigate at least 20% of the data. The performance is not objective as it relies on the threshold set by humans. Additionally, it does not involve any data processing, powerful Pre-trained LLMs, PETuning methods, and effective TL. Moreover, the performance metrics are incomplete and the published performance cannot compete with transformer models [7]. Furthermore, the Shapley value is not suitable for temporal sequence data that are often the input data of learning models capturing temporal patterns.

Qu et al. [40] are pioneers in applying Transformers to FL, demonstrating that Transformers exhibit greater effectiveness and robustness in handling client heterogeneity when compared to conventional architectures like Convolutional Neural Networks (CNNs). Weller et al. [41] provide experimental evidence demonstrating that Pre-trained LLMs can mitigate adverse effects from non-IID and diminish the gap in accuracy between them and centralized learning. Nevertheless, the substantial communication cost in FL systems results in slow and impractical FL for real jobs. Moreover, LLMs may present challenges for local clients with constrained hardware abilities for computation, memory, and storage.

The PETuning method aims to retain the majority of parameters in LLMs frozen while fine-tuning only lightweight extra parameters or a small subset of the parameters for specific downstream tasks. This approach allows Pre-trained LLMs to leverage existing knowledge, minimize resource requirements (computation, memory, storage, etc.), and alleviate communication overhead in FL, which is predominantly influenced by the size of model update parameters. Houlsby et al. [8] developed a down-up projection module named Adapter within each transformer layer for BERT in 2019. Another approach involving fine-tuning the backbone of the pre-trained model is Bitfit [10] that focuses solely on adapting the bias term of the network. Hu et al. [9] demonstrate the feasibility of training large glossary language models while maintaining accuracy and privacy through the introduction of low-rank adaptation (LoRA) in 2021. Nevertheless, it's worth noting that all of these methods were trained centrally on the server without taking into account user privacy concerns, under the assumption of having access to the source data.

Sun et al. [42] introduced the FedPEFT framework in 2022, which incorporates three PETuning methods (Bias, Adapter, Prompt) from pre-trained visual models into FL.

Their findings indicate that employing featherweight PETuning methods in FL can notably alleviate the communication burden while preserving performance, showing superior performance across various FL settings. Additionally, Chen et al. [43] expanded PETuning techniques to visual language models within FL in 2022, demonstrating that PETuning can accelerate convergence rates. Through extensive experiments, they examine three fine-tuning methods (prompt, Adapter, and Bitfit) across two types of pre-trained models (vision-language models and vision models) for FL. Key findings from their experiments include: Fine-tuning the bias term of the backbone performs optimally when utilizing a robust pre-trained model; vision-language models (e.g., CLIP) surpass pure vision models (e.g., ViT) and exhibit greater resilience in few-shot settings; FL with pre-trained models achieves higher accuracy compared to pure local training, as it mitigates overfitting issues. However, these studies overlook the significant issue of privacy attacks in FL. Zhang et al. [11] address the critical issue of privacy attacks inherent in FL. They introduced FedPETuning to evaluate the effectiveness of Pre-trained LLMs in natural language understanding, focusing on the GLUE datasets and using RoBERTa as the base model. FedPETuning maintains satisfactory performance, achieving over 95% of FedFT's performance, while significantly reducing communication overhead. This approach effectively defends against data reconstruction attacks, which aim to recover text from gradients or weight updates uploaded by clients. Since FedPETuning only communicates a small part of the entire model parameters with the server during FL, it becomes impractical for attackers to reconstruct the original text from such lightweight model parameters. However, it is apparent that the samples stored on the server exhibit significantly varied probability distributions compared to the data generated by each client. As a result, the global model fails to achieve localization on each client's site. However, these studies overlook the critical issue of how to build the custom model for each client after FL. Furthermore, none of these studies examine the impact of PETuning methods using various foundation models in FL that are crucial for deploying pre-trained models in FL scenarios.

Even when utilizing the global model trained by the FL system directly, it often performs inadequately on specific client data. This discrepancy arises from the distribution disparity between the client and the globally trained data. Continuing full-tuning or PETuning on client data fails to yield satisfactory results. Instead of involving more parameters of the model in training, transfer learning emerges as a solution. Transfer learning facilitates the transfer of knowledge from existing domains to a new domain without necessitating model reconstruction. It effectively addresses the challenge of training models when data is limited, such as in the case of small data islands. Chen et al. introduce FedHealth [44], the pioneering federated transfer learning framework for wearable healthcare, aimed at classifying human activities. Moreover, IoTDefender [45] employs deep domain adaptation to develop personalized Intrusion Detection System (IDS) models for attack detection in 5G IoT networks. This approach tackles distribution differences between traditional networks and IoT environments. However, these papers do not involve any PETuning method of LLMs in the FL environment and how to apply it in insider threat detection.

Sun et al. [46] introduced an easy and effective method for unsupervised domain adaptation called CORrelation ALignment (CORAL) in 2016. The CORAL method aligns the second-order statistics of the target and source's data distributions through a linear transformation without requiring any label in the target domain. Extensive testing on standard benchmark datasets showcases CORAL's exceptional performance. However, it is worth noting that CORAL relies on a linear transformation and thus it is not end-to-end. It involves a multi-step process wherein features are initially extracted, followed by the transformation, and subsequently training a machine learning model. Sun et al. [47] extended CORAL in 2016 by incorporating it directly into deep neural networks by introducing a new differentiable loss function named CORAL loss. This loss function minimizes the disparity between the learned feature covariance of the target and source domains. Compared to the original CORAL approach, the Deep CORAL method learns a more potent non-linear transformation that is much simpler to optimize. Furthermore, it seamlessly integrates with deep Convolutional Neural Networks (CNNs), exhibiting stronger performance on standard benchmark datasets. However, it has not been integrated into Pre-trained LLMs and applied to temporal sequences yet. Additionally, they do not disclose how to get the best performance across the source domain, target domain, and global domain. Moreover, they do not verify its extensive capability on highly heterogeneous target domain.

III. PROPOSED FRAMEWORK

A. PROBLEM STATEMENT

We have data from K different clients, i.e., organizations, which is indicated by $\{C_1, C_2, \ldots, C_K\}$, and the data collected by each client is symbolized by $\{D_1, D_2, \ldots, D_K\}$. Traditional methods typically train a unified model M_u by aggregating all the clients' data, $D = D_1 \cup D_2 \cup \ldots \cup D_K$. However, this method has to share all the clients' data and thus compromises clients' privacy. Denote a customized client model locally trained based only on the local dataset D_k k $\in$ [1, K] as M_c.

In our problem, our objective is to collaboratively use all clients' information to train a federated transfer learning model M_f, while ensuring each client C_k k $\in$ [1, K] does not disclose its data D_k and keeps their privacy. The performance of M_c, M_u, and M_f are symbolized as P_c, P_u, and P_f respectively. Additionally, another goal is to build an effective framework to enhance P_f to be better than P_c and as close as possible to P_u. Moreover, this framework should be highly efficient in tuning Pre-trained LLMs so we can save training time, communication cost, and resource consumption. Furthermore, M_f is required to be highly adaptable to the C_k's

local data distribution and possesse the ability to generalize well to detect unknown attacks.

$$|P_u - P_f|/P_u < \triangle \tag{1}$$

where $\triangle$ is a very small nonnegative percentage, e.g., 6%.

B. FRAMEWORK OVERVIEW

As Figure 1 shows, the FedITD framework is composed of three systems: the supervised system, the client security system, and the federated system.

1) **The supervised system** encompasses various aspects of the monitored system, such as PCs, file systems, emails, web browsing, devices, etc. The collected data can pertain to either user behavior or user profile. User behavior information is derived from various real-time logging systems, including host logs, network logs, VPNs, firewalls, etc. User profile data, on the other hand, refers to users' background details or contextual information such as user identities, positions, relationships, permitted access, and psychometric attributes. It's worth noting that different client's supervised systems are separated at various locations and operate independently without sharing information.

2) The middle layer is **the client security system** serving as the core of the framework. It is specifically designed to accommodate insider threat detection constituents and act as local access points to federal systems. Consequently, each client security system is empowered to gather data from the respective supervised systems for training the client models and detecting potential attacks. This layer encompasses the entire workflow of insider threat detection, including data processing, model training, pre-trained model tuning, detection, analysis, mitigation, and alert generation. It is assumed that this subsystem possesses sufficient computing power and network bandwidth to preprocess data, execute federated learning, perform transfer learning, and conduct real-time anomaly detection using deep learning models. Meanwhile, it can centrally train its client model based on locally collected data.

3) **The federal system** is situated on the top and facilitates collaborating federated learning, particularly in terms of parameter aggregation and distribution. This study employs a centralized architecture within this system. This architecture comprises a central server located either on the cloud or on-premise, responsible for aggregating updated parameters from agents and training the global model. Additionally, it encompasses agents, which act as sub-servers facilitating the transmission of updated parameters from clients or optional training segmented models tailored to client groups based on their performance or characteristics. These agents also serve as access gateways to the central server for clients.

This system can also be implemented using a **decentralized architecture.** This involves establishing an agent peer-to-peer network, wherein clients collaboratively train deep learning models with high performance, using consensus and incentive mechanisms embedded in the blockchain [48].

Security Interface: Homomorphic encryption is employed to aggregate the update parameters without the server knowing the generated model. The Paillier cryptosystem [49] supports addition and can disseminate the resulting aggregate back to the clients. Subsequently, each client can decrypt the updated parameters by dividing biases and weights by the number of participants to obtain the averaged values.

The main learning process is as follows:

(1) Initially, the global model residing on the central server undergoes training using public datasets or existing collected data. The initial model is then distributed to all clients.

(2) Subsequently, Clients assemble the global model using downloaded lightweight trainable parameters W_t^0 and the backbone Pre-trained LLM's frozen parameters $W_f^{k,0}$, k $\in$ [1, K]. Perform transfer learning to overcome the difference in data distribution between server data and client data.

(3) Following this, clients train the assembled model with their respective private local data D_k, PETuning methods are employed to streamline parameter communication, exchanging only lightweight trainable parameters instead of all the cumbersome Pre-trained LLM parameters.

(4) Post local training, clients transmit their updated efficient parameters, e.g., the k-th client's $W_t^{k,1}$, to the central server for federated aggregation.

(5) The server conducts federated aggregation based on the gathered parameters from clients and sends the updated parameter values, e.g., W_t^2 back to the clients.

(6) Each client trains their client model again on their local dataset for a certain number of local epochs to get a better client model.

Repeat 3), 4), 5), and 6) until convergence of the global model or the maximum number of global training rounds R is reached.

(7) Clients Perform transfer learning to overcome the difference in data distribution between server data and client data

Importantly, it's worth noting that FL serves the purpose of knowledge sharing among all agents through model parameter aggregation. Therefore, all parameter-sharing processes are conducted without risks of user data or sensitive information leakage. This is achieved through techniques such as homomorphic encryption, Parameter Efficient Tuning, local model updates, federated averaging, and Federated Transfer Learning, ensuring the confidentiality and privacy of user data throughout the entire system.

Figure 2 illustrates the workflow of insider threat detection running in the FedITD framework. Data stream/data store, data processing, detection, analysis, execution, actuator, and knowledge form a complete feedback loop. Knowledge is shared among all steps and plays a central role in this workflow. Deep learning models, defined rules, and experts' expertise play the role of knowledge. The client security system's data input is from the supervised system and external data sources (open dataset, shared data from partners, etc.). Supervised system collects data from either real-time data

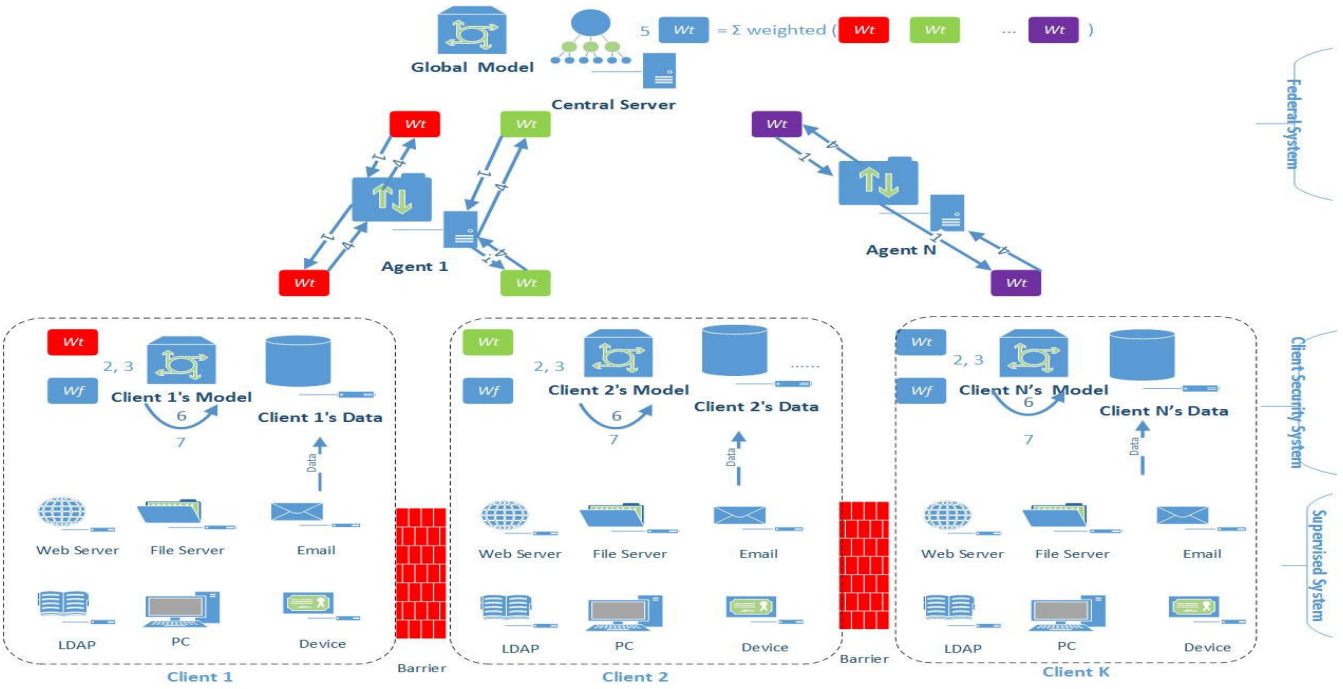

FIGURE 1. FedITD framework and the learning process.

streams from sensors and devices or data stores (database, data warehouse, or distributed file system, etc.) persisting various hosts, devices, network, and system logs. Collected data are preprocessed in data processing and then fed into the rule engine or trigger the deep learning model inference pipeline running fine-tuned models for detection. If there is an abnormal user or activity, the security system will create an incident, raise an alert, and send all related information to the security analyst for analysis. Next, after in-depth analysis, the security analyst can label it as normal behavior or launch an automatic remediation workflow to take action. For instance, he can block the user so that it does not have access to the system or revert the damages done by the user's malicious actions. The actuator executes remediation actions and thus generates new data. So the cycle repeats.

Insider threat mitigation strategies often require detailed investigation, correlation with contextual data, and automated responses. They may include:

- **Containment:** Take immediate steps to contain the threat, such as revoking access, isolating affected systems, or restricting network connectivity. Implement both short-term measures to stop ongoing malicious activities and long-term strategies to prevent recurrence
- **Eradication**: Perform root cause analysis to investigate the incident to determine the root cause and eliminate it. This might involve removing malware, cleaning affected systems, closing vulnerabilities, or addressing policy violations.
- **Recovery**: Restore systems to normal operation, ensuring they are free from vulnerabilities or threats. Increase monitoring of the environment to detect any signs of remaining threats or new attacks.
- **Lessons Learned**: Conduct a thorough review of the incident, including what happened, how it was handled, and what could be improved. Document all findings, actions taken, and lessons learned to enhance future insider risk management. Update policies, procedures, and training programs based on the insights gained from the incident.

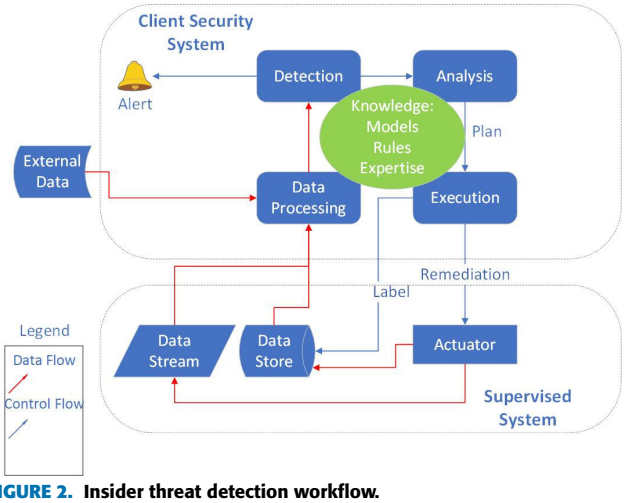

FIGURE 2. Insider threat detection workflow.

C. PRE-TRAINED LANGUAGE MODELS

We select the following pre-trained language model as the foundation model: The BERT model [2], developed by Google, is a pre-trained transformer-based model. It is

trained on a huge amount of unlabeled data, including the Books Corpus which has 800 million words, and English Wikipedia which has 2,500 million words. BERT undergoes pre-training on tasks such as masked language prediction and next-sentence forecasting. We chose the BERT-based model as one of the foundational models. This model consists of 12 encoders with approximately 110 million parameters. Each block contains 12 bidirectional self-attention heads and 768 hidden layers.

By relying on corrupting the input with masks, BERT overlooks the reliance between the masked positions, resulting in a divergence between the pre-trained model and the fine-tuned model. To address these shortcomings, XLNent [4] was proposed as a generalized autoregressive pre-training method. XLNet has two key mechanisms: (1) **Bidirectional Context Learning**: XLNet maximizes the expected probability over all permutations of the factoring order, enabling it to learn bidirectional contexts. It employs a permutation-based approach named **Permutation Language Modeling**, allowing the model to learn from all possible combinations of input tokens rather than a fixed order. This is achieved by training the model to predict the probability of a token given all other tokens in the input sequence, irrespective of their position. (2) **Long-Term Dependency**: XLNet overcomes BERT's shortcomings by using its autoregressive attribute. It integrates ideas from Transformer-XL [50], an advanced autoregressive model, into pre-training to get long-term reliance in the input sequence. This is accomplished through a segment-level recurrence mechanism, enabling the model to preserve the memory of the previous segment while handling the current segment. Under equivalent experimental settings, XLNet consistently beats BERT across a range of 20 tasks, achieving notably superior results. These tasks encompass text classification, question answering, natural language inference, sentiment analysis, and document ranking.

The Robustly Optimized BERT Approach (RoBERTa) [3], developed by Facebook AI, shares similarities with the BERT model but introduces several modifications. Like BERT, RoBERTa learns contextualized word embeddings using a self-attention structure. However, it eliminates the Next Sentence Prediction (NSP) objective. RoBERTa is trained on a larger dataset (160GB), including the CC-NEWS, OPENWEBTEXT, and STORIES datasets. Additionally, RoBERTa increases the batch size, sequence length, and learning rates. It also employs dynamic masking strategies during training to enhance the robustness and generality of word embeddings. Furthermore, RoBERTa adopts byte-level Byte Paired Encoding (BPE) with a vocabulary of 50K sub-word units as a tokenizer, replacing the character-level BPE with a vocabulary of 30K sub-word units used by BERT.

The above-mentioned models are too large, slow, and heavy to fit edge applications. DistilBERT [5] was proposed to pre-train a small, fast, and light general-purpose language representation model. On the other hand, this model can then be fine-tuned to achieve strong performance across a wide array of tasks, akin to its larger equivalents. The knowledge of distillation is applied to simply the model during the pre-training phase. The concept suggests that once a large language model has undergone training, it's possible to approximate its full output distributions using a smaller network. The experiments demonstrated the potential of reducing the size of a BERT model by 40%, computing 60% quicker while keeping 97% of its language understanding abilities. To harness the inductive biases learned by larger models during pre-training, a triple loss integrating language modeling, distillation, and cosine-distance losses is introduced. It is more cost-effective to be pre-trained, and suited for edge device computations or resource-constrained environments.

TABLE 1. The comparison of four foundation models.

	BERT	RoBERTa	DistilBERT	XLNet
Size (MB)	Base: 130 Large: 340	Base: 110 Large: 340	Base: 66	Base: about 110 Large: about 340
Training Time	Base:4xV100 12days Large:280xV100 1 days	Large: 4-5 times more than BERT	Base: 4 times less than BERT	Large: 5 times more than BERT
Claimed Performance		2-20% improvement over BERT	5% degradation from BERT	2-15% improvement over BERT
Training Data	16GB data 3.3 billion words	160 GB data (16GB BERT data + 144GB additional)	16 GB BERT data 3.3 billion words	Base: 16 GB BERT data Large:113GB(16GB BERT data + 97GB additional) 33 billion words
Method	Bidirectional Transformer with MLM and NSP	BERT without NSP	Simplified BERT	Bidirectional Transformer with Permutational Based Modeling

Table 1 compares the size, training time, claimed performance, pre-training data, and methodologies of four models. To sum up, the BERT model is the de facto baseline of Pre-trained LLM in deep learning research. The RoBERTa model is well-known for its prediction performance. The XLNet model's structure is more complex. It removes the limitation of input and its permutation-based training can learn long-term dependencies well, and thus might work well in classification. The RoBERTa model has a simpler structure and a faster inference. These are the reasons that this study chooses the four models as foundation models of insider threat detection.

D. FEDERATED PETUNING

This paper denotes a client as k and K clients are assumed in the Federated Learning system. Each client can only access their private dataset $D_k := \{(x_i, y_i)\}$. (x_i, y_i) is the i-th example where x_i and y_i represent the i-th *model input* and its related label. n_k is the number of examples in D_k while n is the number of examples of all clients' datasets. The client's model parameters are composed of two components: the fixed pre-trained backbone with parameter W_f and the lightweight trainable component with parameter W_t. Therefore, the classification loss of the prediction on example (x_i, y_i) can be represented as $L(x_i, y_i; W_t, W_f)$. The cross-entropy loss is adopted in this study to train client models in FL. The average

loss value of client k can be computed as the below:

$$L_k(W_t, W_f) = \frac{1}{n_k} \sum_{i=1}^{n_k} \mathrm{L}(x_i, y_i; W_t, W_f) \tag{2}$$

Hence, the overarching objective of the FL system can be articulated as minimizing the weighted average loss of all participated clients' insider threat detection:

$$\underset{W_t, W_f}{argmin} L(W_t, W_f) = \sum_{k=1}^{K} \frac{n_k}{n} L_k(W_t, W_f) \tag{3}$$

Our research focuses on a classic FL system, which involves a central server tasked with coordinating participating clients for their local training and disseminating shared model parameters. Rather than transmitting all bulky Pre-trained LLM parameters, this paper adopts Parameter Efficient Tuning Methods to interchange only lightweight trainable parameters. Before the training begins, our system initializes the backbone of the Pre-trained LLM with frozen parameter $\mathrm{W_f}$ and the Parameter Efficient Tuning module named Delta model with trainable parameters $\mathrm{W_t}$. Subsequently, each participant client's local tuning and then the global aggregation on the central server are executed in succession:

Client Local Tuning: upon receiving the trainable parameters from the central server, the participant clients build a complete client model using both lightweight trainable parameters $\mathrm{W_t}$ and local Pre-trained LLM's frozen parameters $\mathrm{W_f}$. Subsequently, these client models are trained on their local data $\mathrm{D_k}$. When local training is completed, each client k $\in$ [1, K] sends its modified trainable parameters $W_t^{k,r}$ to the central server for federated accumulation after dropout.

Server Global Aggregation: The central server disseminates the trainable parameter W_t to the selected K clients. When clients' local tunings are completed, the central server will receive updated trainable parameters $W_t^{k,r}$ from clients. Subsequently, the server executes federated aggregation of these received clients' updated trainable parameters, updating the W_t^r by incorporating these updates:

$$W_t^r = \sum_{k=1}^{K} \frac{|D_k|}{\sum_{k=1}^{K} |D_k|} W_t^{k,r} \tag{4}$$

where $|\mathrm{D_k}|$ is the size of the client k's local dataset and r is the global epoch number. After that, the server returns the updated trainable parameters W_t^r to the clients. Apart from FedAvg, FedITD is also open to other fusion algorithms, such as Adaptive Federated Averaging [51], Shuffle Iterative Average [52], etc., against data inversion attacks or data poisoning.

The training process outlined above iterates until a specific criterion is satisfied, such as meeting the convergence condition or reaching the maximum number of global communication rounds R. Homomorphic encryption is applied in data exchange to avoid any sensitive or privacy information leakage. This entire process is outlined in Algorithm 1 by using pseudo code.

E. PETUNING METHODS OF PRE-TRAINED LLMS

1) Addition-based approaches introduce additional trainable neural modules or parameters that are not present in the original model or process into the Pre-trained LLM. This study chooses the adapter-based tuning method as the representation of addition-based approaches in our experiment. Houlsby et al. [8] proposed inserting small modules, known as adapters, between transformer layers in 2019. Typically, the adapter layer employs a down-projection with $\mathrm{W_d} \in \mathrm{R^{dm}}$ to map the input $h \in \mathrm{R^d}$ into a smaller-dimensional space identified by a bottleneck dimension m. Following this, a nonlinear activation function f(·), and an up-projection with $\mathrm{W_u} \in \mathrm{R^{md}}$ map it back to the input size d. As Figure 3 illustrates, these adapters are enveloped by a residual connection, resulting in the final formula:

$$h = f(hW_d)W_u + h \tag{5}$$

Typically, two adapters are inserted in sequence: one behind the multi-head attention while the other one after the Feed Forward Network (FFN) sub-layer. During fine-tuning, only adapter layers $W_t = \{W_u, W_d\}$, the normalization layer's parameters, and the final classification layer's parameters are trainable, constituting around 0.5% to 8% of the parameters of the complete model in the tuning process, meanwhile retaining the parameters of the Pre-trained LLM frozen.

Algorithm 1 Training Procedure of Federated Parameter Efficient Tuning

Input: Client set C; Global epoch number R; Local epoch number E; Pre-trained LLM's original parameters W_f; Local trainable and efficient parameters W_t; The K clients are indexed by k and the local dataset D_k belongs to the k-th client; Local PETuning Method T.
Output:a fine-tuned global model W_{R+1}
Before Training: Initialize W_t^0 and W_f on the central server and every client in C.
Server Global Aggregation:
For each global round $r = 1$ to R do
  The global model W_t^{r-1} is shared with each client.
  For each client $k \in C^t$ in parallel do
  $W_t^{k,r} \leftarrow$ Client Local Tuning (k, W_t^{r-1})
  End
Get local updated parameters $W_t^{k,r}$
Conduct global aggregation $W_t^r = \sum_{k=1}^{K} \frac{|D_k|}{\sum_{k=1}^{K} |D_k|} W_t^{k,r}$
End
Return W^R
Client Local Tuning (k, W_t^r):
$W^r \leftarrow$ Load and assemble W_t^r and W_f
For each epoch e= 1 to E do
$W_t^{k,r+1} \leftarrow T(D_k, W_t^r)$
End
Transmit $W_t^{k,r+1}$ to the server

As per the original paper [9] (Figure 4), we inserted two adapters sequentially into the foundation model: one following the multi-head attention while the other after the FFN sub-layer. The experiment results show that Macro Average F1 is 0.9286 and accuracy is 99.17%. As Figure5. Shows,

this study also explored another alternative method: only one adapter is added after the FFN "add & layer norm"

2) Specification-based methods entail designating a small part of parameters within the original model or process as trainable, while the remainder is set to remain frozen. This paper selects BitFit as the representation of the specification-based method in our experiment. Zaken et al. [10] introduced BitFit in 2021 and made the experiment by freezing all other parameters and solely tuning the bias terms of the Pre-trained LLM, i.e., $W_t = \{b_{(\cdot)}^{l,(\cdot)}\}$, which constitute only 0.08% and 0.09% of the total number of parameters in BERT base and BERT large models respectively, while competitive performance can still be achieved compared to full-tuning the entire model.

$$f(X) = f(X) + B \tag{6}$$

where f is all functions and B is the bit term. As per the original paper [10], the bias term is either added or unfreezed in the following modules: Attention layer encompassing all positions q, k, and v, along with the output linear layer, FFN, normalization layer, and Classifier. Figure 6 illustrates how to implement the Bitfit tuning method on the RoBERTa model.

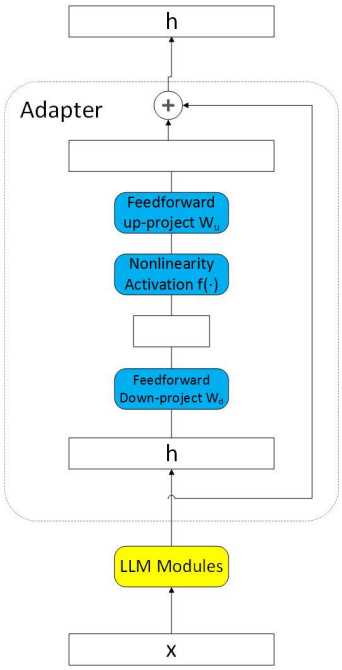

FIGURE 3. The architecture of the adapter module.

3) Reparameterization-based methods convert current parameters into a more parameter-efficient form through reparameterization. This study chooses LoRA as the representation of the reparameterization-based method in our experiment. Hu et al. [9] introduced trainable low-rank matrices alongside multi-head attention (LORA) in 2021, aiming to convert original parameters into a more parameter-efficient style. For a pre-trained weight matrix $W_0 \in \mathrm{R}^{d\times k}$, LoRA expresses its update using a low-rank decomposition:

$$h = W_0 x + \triangle W x = W_0 x + sBAx \tag{7}$$

where s is a tunable scalar hyperparameter that is larger or equal to 1. As Figure 7 shows, LoRA employs this modification to the query and value projection matrices (Wq; Wv) within the multi-head attention sub-layer. During training, W_0 remains fixed and doesn't get gradient updates, whereas A and B encompass trainable parameters. This approach enables LoRA to enhance training efficiency with less than 1% of trainable parameters $W_t = \{B, A\}$, while still achieving performance comparable to fine-tuning on the GLUE benchmark. They testify to the efficacy of their approach across pre-trained language models of different scales and architectures.

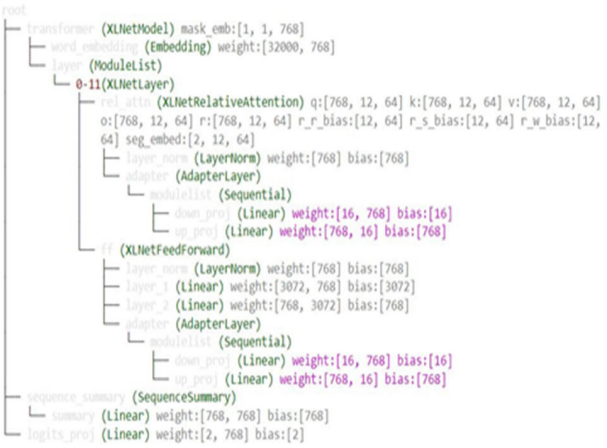

FIGURE 4. The original architecture of XLNet with adapter tuning.

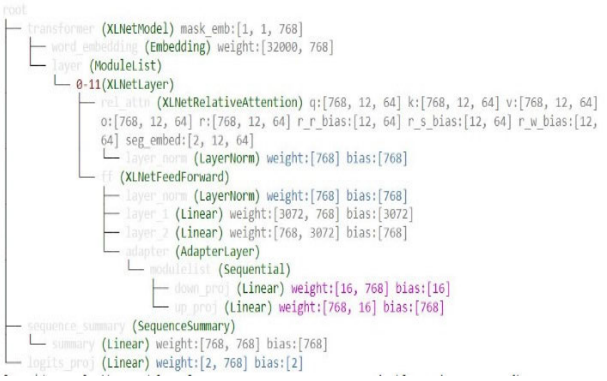

FIGURE 5. The alternative architecture of XLNet model with adapter tuning. Note: the purple part is the delta parameters inserted into the backbone model; the blue part is the tunable parameters of the backbone model.

Initially, this study adopted the original paper's method [9]: For example, when the foundation model is DistilBERT, as Figure 8 shows, q and v matrixes in the attention layer are changed with LoRA matrixes. However, we found that revising other linear layers can result in better performance. As Figure 9 illustrates, FFN's linear layers, pre_classifier layer, and classifier layer are modified with LoRA matrixes. In our implementation, we opted not to substitute the linear layer of the backbone model with the linear layer from Loralib. Instead, we introduced a parallel module into the backbone architecture. In essence, we regard (W_0 + sBA)x

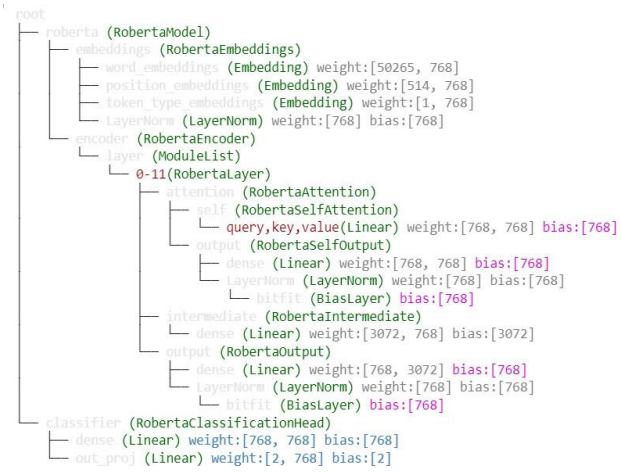

FIGURE 6. The architecture of roberta model with BitFit tuning. Note: the purple part is the delta parameters inserted into the backbone model; the blue part is the tunable parameters of the backbone model.

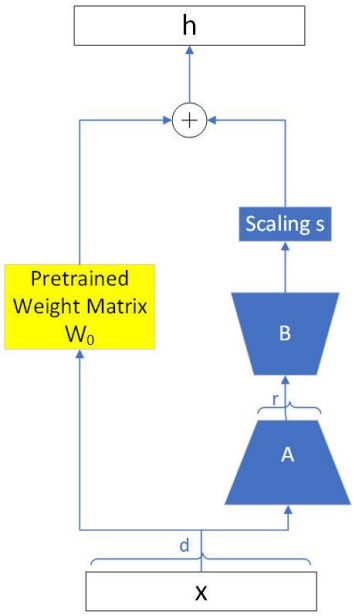

FIGURE 7. The Architecture of the LoRA module.

as $W_0x + sBAx$, incorporating sBAx as a parallel insertion module. LoRA modules are then added after every linear layer. Using the original method, the Macro Average F1 is 0.9390. In comparison, the new method can improve Macro Average F1 to 0.941 while accuracy is the same. Therefore this study adopts the new method.

F. FEDERATED TRANSFER LEARNING

When the federated learning is completed, we can get the global model. However, if we utilize the global model directly on the client's local data, it may still exhibit poor performance for the specific client. The federated learning system operates well under the assumption that the training and test data are independent and identically distributed (i.i.d.). However, this assumption rarely holds in practice due to variations in data among different clients, over time, and across different locations. Consequently, the global model trained by the federated learning system often struggles to effectively handle the distribution disparity between the training data and the client's local data, a phenomenon well-known as **domain shift**. This discrepancy happens from the distribution disparity between the client's data and training data. The global model hosted on the central server typically captures only coarse features from all clients, thus struggling to grasp the nuanced, fine-grained information specific to individual clients. On the other hand, the local data from individual clients typically lacks anomaly samples and thus it is difficult to train a robust model. Therefore, to achieve excellent customized client models, it is necessary to apply transfer learning to fill this gap rather than collecting labeled local data and training a new classifier for every client. Furthermore, the target domain data is usually unlabeled, necessitating unsupervised adaptation techniques rather than supervised adaption techniques to transfer the learning we have already got.

FIGURE 8. The original architecture of the DistilBERT model with LoRA tuning.

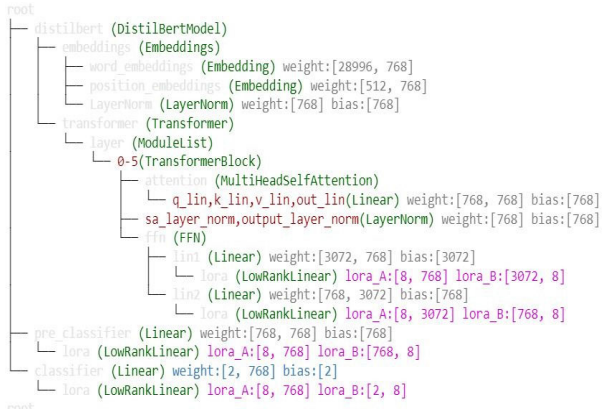

FIGURE 9. The alternative architecture of the DistilBERT model with LoRA tuning.

Deep CORAL [47] is an excellent unsupervised adaption technique to incorporate learning a nonlinear transformation aimed at minimizing the difference in feature covariances between source and target domains. We have source-domain training examples $D_S = \{x_i\}$, $x_i \in \mathrm{R}^{\mathrm{d}}$ with labels $L_S = \{y_i\}$,

$i \in \{1,\ldots, L\}$, and unlabeled target data $D_T = \{u_i\}$, $u_i \in R^d$. Let n_S and n_T stand for the number of source and target data, correspondingly. C_S indicates the feature covariance matrix of the source data while C_T represents the feature covariance matrix of target data.

The differential CORAL loss is designated as the discrepancy between the covariance of the source and target features.

$$L_{CORAL} = \frac{1}{4d^2}||C_S - C_T||_F^2 \tag{8}$$

where $||\cdot||_F^2$ represents the squared matrix Frobenius norm. The covariance matrices for the source and target data are represented respectively as follows:

$$C_S = \frac{1}{n_S - 1}(D_S^T D_S - \frac{1}{n_S}(\mathbf{1}^T D_S)^T(\mathbf{1}^T D_S)) \tag{9}$$

$$C_T = \frac{1}{n_T - 1}(D_T^T D_T - \frac{1}{n_T}(\mathbf{1}^T D_T)^T(\mathbf{1}^T D_T)) \tag{10}$$

Our goal is that the ultimate deep features must possess both discriminative powers to effectively train a robust classifier and constant to differences between the source and target domains. Solely minimizing the classification loss may result in overfitting to the source domain, thereby diminishing performance on the target domain. Conversely, only minimizing the CORAL loss alone could produce deteriorated features and poor detection performance. Training jointly with minimizing both the classification loss and CORAL loss can result in learning features that are effective on the target domain.

$$L = L_{CLASS} + \sum_{i=1}^{t} \lambda_i L_{CORAL} \tag{11}$$

Here, t represents the number of CORAL loss layers in a deep network, and λ is a weight parameter that balances the adaptation with classification accuracy on the source domain. In this study, the covariance matrices for the source and target features are calculated on the temporal sequence feature. CORAL Loss is then seamlessly integrated into the total loss of the Pre-trained LLM instead of adding an alignment layer. The client model is re-trained with the target of minimizing the total loss L including both the classification loss and CORAL loss on the client's local data. By tuning λ, these two losses compete with each other and finally converge to an optimal equilibrium during training, resulting in final features that are anticipated to perform effectively on both the target domain and source domain. The detailed procedure is described in Algorithm 2. Besides Deep CORAL, FedITD offers extensibility by accommodating other transfer learning algorithms.

IV. EXPERIMENT

Two kinds of experiments are carried out to assess the effectiveness of the framework FedITD. The first kind of experiment aims to assess federated PETuning methods including their fundamental capability of insider threat detection, resource costs, and privacy preserving. The other one is to evaluate the effectiveness of transfer learning consisting of unsupervised domain adaptation's performance evaluation, comprehensive ablation analysis of FL and TF's contribution, performance evaluation on the highly heterogeneous and slightly heterogeneous target domain, generalization ability's evaluation, and how to attain the optimal equilibrium that performs well across source domain, target domain (local test), and global test.

Algorithm 2 Training Procedure of Federated Transfer Learning

Input: Client set C; Local epoch number E; a fine-tuned global model f_S; The K clients are indexed by k and the local dataset D_k belongs to the k-th client; the global test data set are original data including all anomaly scenarios; the source and target validation dataset are the augmented data that do not join model training in the source and target domain. λ is a weight parameter to tradeoff between classification performance and domain adaption.
Output: a well-customized client model f_k for client k
Download the well-tuned global model f_S from the central server
Repeat
  Tune the value of λ
For each epoch e= 1 to E **do**
  Calculate classification loss by using cross-entropy
  Calculate CORAL loss by using (8)
  Train client k's model f_k on the private data set D_k using (11)
  Evaluate the client model f_k's performance on the source validation dataset
  Evaluate the client model f_k's performance on target validation dataset
End
Evaluate the client model f_k's performance on the global test dataset
Until Find an optimal λ to achieve optimal equilibrium that the model performs the best across source validation dataset, target validation dataset, and global test dataset

A. EXPERIMENT SETTING

This study conducts our experiments utilizing Google Colab, a virtual environment equipped with an NVIDIA A100-SXM4-40GB GPU, 83.48GB of RAM, and a disk capacity of 166.77 GB. The PyTorch version is 2.0.0 and Python version is 3.10.11. Key software packages utilized for model development include PyTorch, Pandas, Matplotlib, Transformers, and Scikit-learn. This study adopts RoBERTa-Base, BERT-base-cased, XLNet-base-cased, and distilbert-base-cased as the foundation models using the AutoModelForSequence-Classification package released by Huggingface V4.39.3. All parameter-efficient tuning approaches are implemented based on OpenDelta [53], which is an Open-Source toolkit for parameter-efficient tuning. The infrastructure of the Federated Learning system is developed based on FedLab [54], which is a flexible federated learning framework. Through-

out the study, our model was trained with batch sizes of 32. In federated learning, the model is tuned for 1 local epoch in each client's site, while the global communication epoch is 10 rounds. Cross-entropy loss and the Adam optimizer are employed for local training. For each model and tuning approach, we conduct a hyperparameter sweep. The best model hyperparameters are chosen based on the metric performance on the test data set. Specifically, we explore learning rates from {1e-6, 5e-5, 1e-4, 5e-4, 1e-3, 5e-3, 1e-2}. For the Adapter, we search the bottleneck dimensions from {16, 64}, and the ranks from {4, 8, 16} are examined for LoRA.

B. DATASET

Due to security and privacy reasons, it is very hard to find real insider threat datasets. This study chooses the CERT r6.2 dataset as the experimental data source because CERT is the most popular open dataset for insider threat detection adopted by most research literature. Especially version r6.2 is more complex, challenging, and perfectly simulates real scenarios. It comprises synthetic scenarios, malicious users, and their behavioral data. Developed by the CERT Division of the Software Engineering Institute at Carnegie Mellon University, it mimics IT-based activities and personal profiles within a fabricated organization named DTAA. The dataset encompasses five log files:

(1) logon.csv: Records all employees' logon and logoff activities on PCs
(2) email.csv: Keeps all emails of employees
(3) http.csv: Saves web browsing history records of employees
(4) file.csv: Collects file-related operations such as open, write, copy, or delete.
(5) device.csv: Archives the connection information for removable drives.

Additionally, the CERT dataset offers all employees' HR data in LDAP.csv and psychometric grades in psychometric.csv.

CERT r6.2 presents a significantly large corporation with 4000 employees and a vast number of activities totaling 135,117,169. Within this dataset, only five users exhibit malicious behavior, and there are merely 470 related anomalous activities. Despite the presence of five insider risk scenarios, each scenario involves only one malicious employee. Notably, both scenario 3 and scenario 5 occur on one day. Among the five scenarios, one scenario involves data exfiltration, another one pertains to system sabotage, and the remaining three scenarios revolve around embezzling intellectual property. Due to the sporadic nature of CERT r6.2, it contains fewer instances of malicious users and their activities, while significantly more instances of normal user behaviors. Detecting insider threats within such an imbalanced dataset is considerably challenging.

The data processing process proceeds as follows:

1. Initial data cleaning is performed to remove erroneous entries and fill in missing values. Remove or obfuscate sensitive information before participating in federated learning. This includes data anonymization and feature engineering to reduce the exposure of identifiable data.
2. Five domain data files (login, device, file, email, and HTTP) are loaded into their respective individual data subsets
3. Useful features are extracted and generated from the data subsets, including user IDs, activity time, activity types, etc.
4. User behavior is encoded into numeric tokens, incorporating activity type, temporal information, and relevant features.
5. All five data subsets are combined into a unified dataset, with data grouped first by user and then by day. Subsequently, each user's behavior is sorted chronologically to generate their daily behavior sequence.
6. Incorporate the ''insider.csv'' dataset to label the user behavior sequence as normal or anomalous.
7. Text data transformation is performed, converting each data instance into a word embedding based on a trained custom vocabulary.

The details of data processing are described in our previous paper [7]. Additionally, this study addresses highly imbalanced data issues through NLP data augmentation techniques, such as contextual embedding word insert and substitution using associate pre-trained LLMs.

C. EVALUATION METRICS

To assess the effectiveness of our model, we employed the following performance metrics as evaluation criteria:

- **Accuracy**

$$Accuracy = \frac{TP + TN}{TP + TN + NP + NN} \tag{12}$$

where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, correspondingly. Accuracy represents the percentage of correctly identified items among all items. In other words, it indicates the proportion of correctly detected instances among all detected instances.

- **Recall**

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

Recall, also known as **sensitivity** or true **positive rate**, is the proportion of actual positives that are correctly classified as positive. In other words, it measures the percentage of positive instances that are correctly identified as positive.

- **Precision**

$$Precision = \frac{TP}{TP + FP} \tag{14}$$

Precision, also known as **confidence**, is the percentage of correctly identified positives among all instances predicted as positive. In other words, it measures the proportion of predicted positive instances that are truly positive.

- **F1 Score**

$$F1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (15)$$

The F1 Score, also known as the F-score or F-measure, is the harmonic mean of precision and recall. If precision increases, there may be a decrease in recall and vice versa. It balances precision and recall, allowing for a trade-off between the two metrics.

- **The area under ROC curve (AUC)**

The ROC curve, the Receiver Operating Characteristic curve, is derived from the coverage curve through axis normalization to the range [0, 1]. It illustrates the trade-off between true positive rate (sensitivity) and false positive rate (1-specificity) as the discrimination threshold varies. The area under the ROC curve (AUC) quantifies the ranking accuracy of the model.

Because the global test dataset contains only a few anomaly instances and a huge number of normal instances. If the model predicts all instances to be normal, it can still achieve high accuracy. Therefore, accuracy is not a reliable measure metric in this experiment. The F1 score provides a comprehensive measure of both precision and recall. However, if increasing the F1 score of one class, there may be a decrease in the F1 score of the other class. **Macro Average F1-score** takes the F1 scores of each class and averages them, treating all classes equally. Therefore, this study chooses the Macro Average F1-score as the first measure metric and keeps accuracy as the second measure metric.

- **Communication Cost**

The communication cost 'c' can be computed using the following formula:

$$c = rXnXsX2 \quad (16)$$

where r is the global communication round, n is the count of the clients, and s is the size of the trainable parameters. The trainable parameters include tunable parameters in both the tunable modules and the foundation model.

- **Storage Overhead**

Storage overhead is measured by the model size. The size of the model includes the size of the foundation models and the size of the delta model. The size of the delta model is the size of a tunable module, e.g., Adapter, or RoLA.

- **Memory Usage**

Memory usage is the amount of RAM required to run the model during inference. It is a very important performance metric and especially useful for deploying models on devices with limited memory.

V. EVALUATION AND ANALYSIS

A. EVALUATION RESULTS

1) FEDERATED TUNING PERFORMANCE

The experiments are performed in the Cross-silo Setting. Cross-silo Setting caters to multiple users, typically no more than 100 clients. Almost all experiments except for evaluating the impact of a large number of clients are performed in this setting. The total number of clients is set to 10. The central server chooses all clients for training in every communication round. In the experiment of federated learning, this study adopts contextual word embedding augmentation methods using associate LLMs to upsample the anomaly behavior data because this method is proven to make the model perform better [7]. In the augmented dataset, the ratio of original normal user behavior data to augmented anomaly user behavior data is 55%: 45%. 80% of the whole data is split as the training dataset, 10% of the whole data is split as the validation dataset, and the remaining 10% is kept as the test dataset. The ratio of original normal user behavior data to augmented anomaly user behavior data is 1:1 in the both training dataset and the validation dataset. In the test dataset, both normal and anomaly user behavior data are all original data. Then the training dataset and validation dataset are divided among clients as their local data.

TABLE 2. The performance of federated and central tuning of LLMS.

Macro AVG F1 (Accuracy)	RobertTa	BERT	Xlnet	DistilBert	AVG
FedFT	0.9216(0.9917)	0.941(0.9935)	0.941(0.9935)	**0.9534(0.9954)**	0.9393(0.9935)
FedAP	**0.9369(0.9935)**	0.9101(0.9898)	0.9286(0.9917)	0.917(0.9908)	0.9232(0.9915)
FedBF	**0.9216(0.9917)**	0.8906(0.9871)	0.7175(0.9445)	0.9034(0.9889)	0.8583(0.9781)
FedLR	**0.945(0.9945)**	0.9241(0.9917)	0.9286(0.9917)	0.941(0.9935)	**0.9347(0.9929)**
CenFT	**1(1)**	0.9633(0.9963)	0.9739(0.9972)	0.9534(0.9954)	0.9727(0.9972)
CenAP	**0.9739(0.9972)**	0.9633(0.9963)	0.9502(0.9945)	0.9101(0.9898)	0.9494(0.9945)
CenBF	**0.9739(0.9972)**	0.945(0.9945)	0.8451(0.9787)	0.9314(0.9926)	0.9239(0.9908)
CenLR	0.945(0.9945)	0.955(0.9954)	**0.9739(0.9972)**	0.9534(0.9954)	**0.9568(0.9956)**

NOTE: ACCURACY IS IN THE BRACKET WHILE MACRO AVERAGE F1 IS OUTSIDE OF THE BRACKET. THE BOLD FONT IS THE BEST PERFORMER OF THE SAME MODEL USING THE SAME TUNING METHOD. THE BLUE FONT IS UNDERPERFORMERS.

As depicted in Table 2, in terms of federated learning with PETuning methods, the RoBERTa model with LoRA tuning demonstrates superior performance compared to other methods, while the XLNet model with BitFit tuning performs the poorest. Across various base models, RoBERTa consistently exhibits the highest performance, outperforming DistilBert, BERT, and XLNet with all PETuning methods. The performance ranking is as follows: RoBERTa > DistilBert > BERT > XLNet. When comparing different PETuning methods, the performance rankings are as follows: LoRA > Adapter > BitFit. In terms of federated learning with full tuning methods, DistilBERT performs best with Macro Average F1 0.9534 and an accuracy of 99.54%. This is consistent with our past conclusion that the simplified encoder architecture of the transformer model performs best in insider threat detection [7]. Only the Macro Average F1 score of XLNet with BitFit tuning and BERT with BitFit tuning are lower than 0.90. It shows that although the BitFit module is added or unfreeze bit term in many layers of the foundation model, but it carries much less weight in classifying insider behavior sequence.

In terms of central learning with PETuning methods, the RoBERTa model with adapter tuning and BitFit tuning and

TABLE 3. The Relative percentage of macro avg F1 between the federated and central tuning of PLMS.

Relative %	RoBERTa	BERT	XLNet	DistilBERT
FT	**-7.84%**	-3.38%	-3.38%	0.00%
Adapter	-5.70%	-5.52%	-4.65%	0.76%
BitFit	-5.37%	-5.76%	**-15.10%**	-3.01%
LoRA	0.00%	-4.65%	-4.65%	-1.51%

TABLE 4. The Relative macro avg F1 between federated PETuning and federated Full-tuning.

Relative Gap	RobertTa	BERT	XLNet	DistilBert
FedFT	-3.34%	-1.30%	-1.30%	0.00%
FedAdapter	-1.73%	-4.54%	-2.60%	-3.82%
FedBitFit	-3.34%	**-6.59%**	**-24.74%**	-5.24%
FedLora	-0.88%	-3.07%	-2.60%	-1.30%

XLNet with LoRA tuning demonstrate superior performance compared to other methods, while the XLNet model with BitFit tuning performs the poorest. Across various base models, RoBERTa consistently exhibits the highest performance, outperforming DistilBert, BERT, and XLNet in all PETuning methods. The performance ranking is as follows: RoBERTa > BERT > DistilBert > XLNet. When comparing different fine-tuning methods, the performance rankings are the same as federated learning: LoRA > Adapter > BitFit.

Relative Percentage of Macro Average F1 between federated learning and central tuning is the macro average F1 of the federated tuning minus the macro average F1 of central tuning with the same tuning method and then divided by the macro average F1 of the central tuning with the same tuning method. As Table 3 shows, XLNet with BitFit tuning performs the worst as the reduction rate attains 15.10%. Additionally, the relative percentage of macro average F1 of RoBERTa with full-tuning is −7.84%. However, all other models with other tuning methods perform well as their relative macro average F1 percentage reduction are all less than 6%. Especially DistlBERT with full-tuning and RoBERTa with LoRA tuning's macro average F1 have no changes between federal tuning and central tuning.

The relative Percentage of Macro Average F1 between federated PETuning and federated full-tuning is the percentage of performance gap achieved by federated PETuning relative to the best federated full-tuning performer DistilBERTin terms of Macro average F1. As Table 4 shows, XLNet with BitFit tuning performs the worst as the gap is -24.74%. Additionally, the relative macro average F1 percentage of BERT with BitFit tuning is 6.59%. However, all other models with other tuning methods perform well as their relative percentage of macro average F1 are all less than 6%.

From the above results, we can see that the performance of most PETuning methods downgrades by less than 6% in federated learning environments compared to corresponding PETuning methods in central learning environments. Additionally, the performance of most PETuning methods downgrades less than 6% in federated learning environments compared to corresponding full-tuning methods in the same environment. However, FL using Pre-trained LLMs with most PETuning methods can achieve high Performance in terms of Macro Average F1 score (>0.90) and accuracy. RoBERTa demonstrates superior performance compared to other models and exhibits greater robustness in a federated learning environment. LoRA emerges as the most effective PETuning method when leveraging a strong Pre-trained LLM like RoBERTa or DistilBert. Nevertheless, some combinations of Pre-trained LLM and PETuning methods especially XLNet with BitFit tuning method have a considerable degeneration in both federated learning (−24.74%) and central learning environment (−15.49%).

Finally, we evaluate the impact of a large number of clients in Large-scale Cross-device Setting. Large-scale cross-device setting represents the other federated setting intended for deployment across a large number of clients. In this scenario, data held by local clients are more sporadic than in the former setting. In the experiment, the total number of clients in the large-scale settings is 100, and the sampling rate is set as 10%. In each communication round, 10 clients are randomly sampled from all available clients to participate in both local training and federated aggregation processes in the federated learning process. Employing RoBERTa as the base model and LoRA as the tuning approach, we replicate the aforementioned training and testing procedures. The findings indicate that the algorithms' performance remains robust. The Macro Average F1 experiences only a minor decrease of 3.34%, while the accuracy decreases by a mere 1.15%. This suggests that the number of clients has minimal influence on performance.

2) RESOURCE COST

The resources associated with various PETuning methods encompass the storage overhead, the communication cost, and the memory usage.

Table 5 shows the size of trainable parameters and the trainable ratio which is the ratio of the size of trainable parameters to the size of the corresponding models' all parameters. Observations reveal that the size of DistilBERT with the BitFit tuning method is remarkably compact, occupying only 0.04MB, which represents a mere 0.07% of the backbone model's size. In contrast, the BERT model with

Adapter tuning exhibits the largest size at 1.74MB, attributed to the insertion of learnable modules, termed Adapters, into certain transformer layers. According to Table 5, in terms of tuning methods, the sizes of their trainable parameters are ordered as follows: FedFT ≫ FedAP > FedLR > FedBF. Regarding foundation models, the sizes of their trainable parameters are ordered as RoBERTa > BERT > XLNet > DistilBERT. The experimental findings reveal that the size of trainable parameters only accounts for 0.04%

TABLE 5. The size of trainable parameters of fine-tuning methods.

The Size of Trainable Parameters (MB)	RobertTa	BERT	XLNet	DistilBert
FedFT	124.64(100%)	108.31(100%)	117.31(100%)	65.78(100%)
FedAP	1.2(0.96%)	1.74(1.64%)	0.62(0.55%)	0.29(0.46%)
FedBF	0.66(0.53%)	0.08(0.08%)	0.06(0.06%)	0.04(0.07%)
FedLoRA	0.89(0.71%)	0.29(0.27%)	0.70(0.62%)	0.37(0.58%)

NOTE: THE SIZE OF THE TRAINABLE PARAMETER IS OUTSIDE OF THE BRACKET WHILE THE TRAINABLE RATIO IS IN THE BRACKET.

TABLE 6. The communication cost.

The total communication cost in the training(MB)	RobertTa	BERT	XLNet	DistilBert
FedFT	24928	21662	23462	13156
FedAP	240	348	124	58
FedBF	132	16	12	8
FedLoRA	178	58	140	74

to 1.64% of the size of all parameters of the models. That means that PETuning methods can use 62 times to 1955 times fewer trainable parameters than a full-tuning method to attain a performance close to the full-tuning method in FL settings. In addition, by deploying multiple tasks on a local client, PETuning methods can facilitate different tasks to share one Pre-trained LLM. The client is allowed to retrain only a small number of trainable parameters for each task, thereby reducing storage requirements. Moreover, this reduction in the size of trainable parameters results in a significant decrease in communication cost because the amount of data in the communication significantly reduces. Furthermore, it is very helpful to preserve client privacy since the client data exposed to the network is much decreased.

Table 6 illustrates the communication costs for all tuning methods under FL settings. The communication cost of the DistilBERT model with the BitFit tuning method is notably small, occupying just 8MB. Conversely, the communication cost of the BERT model with Adapter tuning is the largest, measuring 348MB. As shown in Table 6, in terms of the PETuning method, the communication costs are ranked as follows: FedFT $\gg$ FedAP $>$ FedLR $>$ FedBF. Regarding foundation models, the communication costs are ranked as RoBERTa $>$ BERT $>$ XLNet $>$ DistilBert. The experimental findings indicate that the Federal PETuning methods greatly enhance training efficiency by substantially decreasing communication costs from 98.39% to 99.94% compared to their FedFT methods on their corresponding models. This reduction in communication costs also leads to a significant decrease in training time during both uploading and downloading parameters. Consequently, Federal PETuning proves to be a more practical choice for real-world applications, especially in networks with communication constraints.

TABLE 7. The memory of the delta model and the total model.

The Memory of the Model (MB)	RobertTa	BERT	XLNet	DistilBert
FedFT	475.50(475.50)	413.18(413.18)	450.31(450.31)	250.95(250.95)
FedAP	2.32(480.14)	6.96(431.58)	2.32(452.15)	1.16(253.27)
FedBF	0.25(475.82)	0.39(413.64)	0.25(447.82)	0.16(251.10)
FedLoRA	1.12(477.75)	1.12(415.43)	2.81(453.13)	1.48(253.90)

NOTE: THE MEMORY OF THE DELTA MODEL IS OUTSIDE OF THE BRACKET WHILE THE MEMORY OF THE TOTAL MODEL IS IN THE BRACKET

Memory usage is also an important aspect of the performance. However, most researchers ignored memory usage. Table 7 illustrates the memory efficiencies of PETuning methods and the Full-tune method across various models in a Federated Setting. RoBERTa model with BitFit tuning and XLNet with BitFit tuning's memory are the smallest at 0.25MB while the BERT model with Adapter tuning has the largest memory usage which is 6.96 MB. PETuning methods' delta model only accounts for 0.05% to 1.61% memory usage compared to the corresponding original foundation models. This advantageous feature is particularly beneficial for local clients in real-world FL systems, especially edge devices.

3) PRIVACY PRESERVING

The experiment of an embedding inversion attack is performed to evaluate the privacy protection capabilities of FedITD. The embedding inversion attack is to recover original input from clients' model weight updates in an FL environment. After removing encryption and federated dropout, this study initializes a tensor of random embeddings and uses a backward optimization process to minimize the Mean Square Error (SME) loss between recovered embeddings and actual embeddings [55]. After optimization, the closest token indices are identified by the Euclidean distance and then decoded back to the recovered text. The F1 score is used to evaluate the performance of the inversion attack. We randomly selected 120 daily user behavior samples from the CERT dataset r6.2 after data processing as an attack dataset and evaluated the performance of the attack using the RoBERTa model with full tuning and LoRA tuning methods. The result shows that the F1 score of the former is 0.1803 while the latter is reduced to 0.0602. It demonstrated that FedITD can more effectively protect clients against data reconstruction attacks even if encryption and federated dropout are removed.

4) TRANSFER LEARNING

We performed three experiments to assess the efficacy of transfer learning in FedITD. The initial experiment involves assessing the fundamental capability of insider threat detection through TL, the performance comparison of various

tuning methods, and an ablation study. The second experiment aims to verify its capacity for generalization, while the third one seeks to validate unsupervised domain adaption's effectiveness and identify its optimal configurations.

(1)Detection Performance Evaluation

Because XLNet net with BitFit tuning has the worst performance after FL. This study chooses the XLNet model tuned using the BitFit method in FL as the starting point. This study compares FedITD's performance with that of other tuning methods: **Full-tuning** method entails adjusting all parameters of the downloaded global model from FL using the client's local data without explicitly employing transfer learning techniques, such as minimizing distribution divergence between domains. **The PETuning** method involves continuing parameter-efficient tuning on the downloaded global model obtained from FL using the client's local data, without explicitly employing transfer learning techniques as well. Furthermore, we perform an ablation study to assess the contributions of the two primary elements: federated learning and transfer learning. The term "**Only Fed**" refers to using the global model trained in FL to apply to client local data without a personalized transfer learning process. In addition, "**Only TL**" means that the global model XLNet undergoes retraining solely using client local data with transfer learning, without involvement of federated learning anymore.

To perform the experiment of transfer learning, we randomly chose 70% of the training data that participated in FL training from each client as the source domain's training dataset. selecting 70% of data from each client's validation dataset that did not participate in FL training as the source domain's validation dataset. The ratio of original normal user behavior data to augmented anomaly user behavior data is 1:1 in the source domain's data. Test datasets in the federated tuning experiments are used for the global test. One client (Client 1) is then chosen from ten clients. Its training dataset is used as the target domain's training dataset while its validation dataset is used as the target domain's validation dataset. Because the target domain's dataset uses the same data augmentation method as the source domain's dataset, data heterogeneity is light. To increase data heterogeneity, we also use the dataset augmented by another method (the contextual sentence embedding method) as the target domain's training and local validation dataset. The former is called the Lightly Heterogeneous (LH) dataset while the latter is called the Highly Heterogeneous (HH) dataset.

From Table 8, we can see that utilizing the global mode trained in FL directly on clients (**Only Fed**) leads to poor performance, primarily due to distribution differences between the source domain and target client domain data. Global models trained via FL typically only grasp common and low-level characteristics from clients, often missing out on fine-grained, specific features unique to each client. Consequently, it becomes essential for clients to engage in transfer learning post-obtaining the global model to attain customized client models. The experiment result also indicates that only applying transfer learning without FL (**Only TL**) can effectively handle distribution shifts to significantly enhance the performance of detection on both LH and HH datasets. However, FedITD combining FL and deep transfer learning achieves the best detection performance whether in terms of Macro Average F1 or accuracy on both LH and HH datasets. In comparison, although full tuning and PETuning achieve a good performance in the local test after 20 epochs of tuning, they still do not attain a satisfactory performance in the global test. FedITD achieves the highest Macro Average F1 score, surpassing the Full-tuning method by 2.62% and 27.08% in the local test and global test on the HH dataset respectively. Additionally, FedITD also achieves the best classification accuracy, outperforming the Full-tuning method by 2.64% and 7.21% in the local test and global test on the HH dataset respectively. It successfully detects all five anomaly scenarios in CERT r6.2, whether scenario 2 which has 252 malicious actions and last for 8 weeks, or scenario 5 that only has 4 malicious actions and last for less than 2 minutes.

TABLE 8. The performance of transfer learning on LH and HH datasets.

	Test	Full-tuning	PETuning	Only Fed	Only TL	FedITD
LH Dataset	Local Test	0.9737(0.9737)	0.9163(0.9165)	0.3391(0.5131)	0.9938(0.9938)	**0.9954(0.9954)**
	Global Test	0.7466(0.9519)	0.5743(0.8327)	0.7175(0.9445)	0.9062(0.9890)	**0.9125(0.9881)**
HH Dataset	Local Test	0.9378(0.9379)	0.8103(0.8105)	0.3571(0.5556)	0.9479(0.9481)	**0.9640(0.9643)**
	Global Test	0.6809(0.9233)	0.5234(0.7717)	0.4930(0.9723)	0.9423(0.9917)	**0.9517(0.9954)**

NOTE: MACRO AVG F1 IS OUTSIDE OF THE BRACKET AND ACCURACY IS IN THE BRACKET

Based on the information provided by Table 8, it is evident that both federated learning and transfer learning play significant roles in the performance of FedITD. Federated learning enables the central server to leverage a broader spectrum of information from multiple clients indirectly, leading to a more generalized global model. On the other hand, transfer learning empowers clients to further improve the global model to adapt to the client's local data, thereby obtaining a more customized client model. On the contrary, due to the diversity of datasets, only federated learning models are more prone to experiencing overfit rising false positive rates or declining sensitivity. Similarly, only transfer learning models may struggle to detect unknown attacks that are not present in the source domain, as it is hard to get knowledge about these attacks from other clients. FedITD can address these shortcomings by constructing a custom model for individual clients using both transfer learning and federated learning. Each client's model is refined to capture their unique behavior traits and detect unknown attacks that do not show up in the local dataset, leading to a more precise, tailored, and general model.

(2) Model Generalization Evaluation

To demonstrate the superior generalization ability of FedITD, we conducted data processing before the experiment

described in the previous subsection to validate its capacity to not only detect insider attacks present in a client's local dataset but also identify unknown attacks. Initially, we removed some specific users' anomaly behavior data from the client's local training and local test datasets while retaining them in the global test set. Specifically, we excluded user CDE1846's (scenario 4) and user MBG3183's (scenario 5) anomaly instances from client 1's target domain training and local test datasets while leaving the source domain's datasets unaffected and also keeping them in the global test dataset. Subsequently, we conducted the aforementioned experiment outlined in the previous subsection. Notably, since user CDE1846 and user MBG3183 are absent from client 1's local dataset, models solely trained on local data without any federated learning or transfer learning process struggled to identify user CDE1846 and user MBG3183's anomaly behavior, resulting in poor detection performance in the global test. Conversely, Only TF method, having access to the source domain dataset, could effectively learn user CDE1846 and user MBG3183's anomaly behavior, thus exhibiting superior performance whose Macro Average F1 is 0.9423 and accuracy attains 99.17% in the global test. Despite client 1's limited local data, it successfully detected user CDE1846's anomaly behavior, which had not previously appeared in client 1's local data. In contrast, as Table 8 shows, FedITD outperformed OnlyTF in terms of both macro average F1 (0.9517) and accuracy (99.54%). This improvement is attributed to FedITD's excellent ability to facilitate client 1 to learn the knowledge of user CDE1846 and user MBG3183's anomaly behavior from other clients through not only transfer learning but also federated learning. These robust results demonstrate the strong generalization ability of FedITD.

(3) Model Localization Analysis

Let's dive into transfer learning using the deep CORAL method: As shown in Figure 10, compared with the TL training and local test performance without CORALL loss, it's evident that incorporating the CORAL loss significantly enhances the performance in the target domain while preserving robust classification accuracy in the source domain. Fine-tuning without domain adaptation may lead to overfitting the model to the source domain, potentially diminishing its performance in the target domain. Integrating the CORAL loss into the system's loss mitigates the dissimilarities of data distribution between the source and target domains during the fine-tuning process. It helps maintain a balance, ensuring the model performs effectively in both source and target domains.

In Figure 11, both the classification loss and the CORAL loss are depicted for training with the CORAL loss when λ is equal to 0.75 and the learning rate is 0.01. Initially, the CORAL loss is minimal, contrasting with the considerable classification loss. However, after 20 epochs of training, these two losses converge to a similar magnitude.

Next, this study will explore how to set lambda values to optimize the performance on both local test and global test. When the target dataset is the LH dataset, as Figure 12 shows, in the local test, with the increasing of λ value, the macro

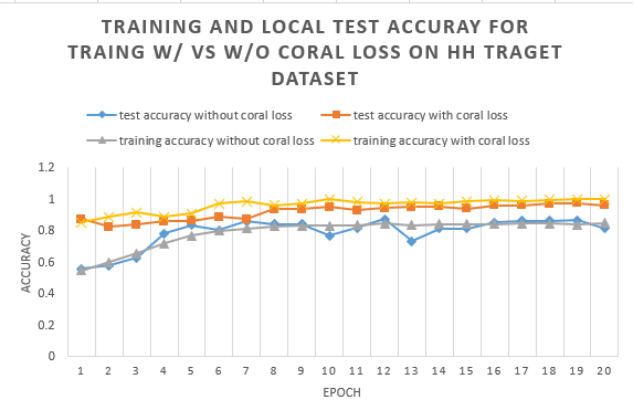

FIGURE 10. Training and Local Test Accuracy for Training w/ vs w/o CORAL Loss on HH Datasets.

average F1 score increases till it attains the maximum value at 0.9640 and the accuracy also rises till it achieves the top value at 96.43% when λ is equal to 0.75. After that, both macro average F1 score and accuracy decline with the increase of λ. The global test shows the same tendency: with the increase of

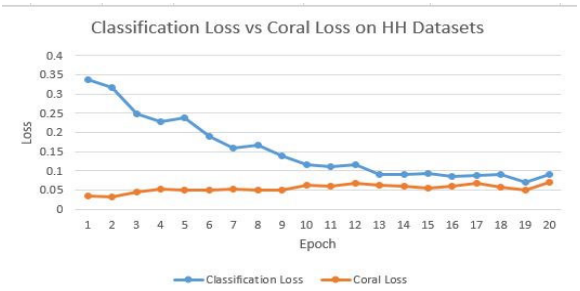

FIGURE 11. Classification Loss vs CORAL Loss on HH Datasets.

λ value, the macro average F1 score increases till it attains the maximum value at 0.9517 and the accuracy also rises till it achieves the top value at 99.54% when λ is equal to 0.75.

When the target domain data is the HH dataset, the case becomes different. As Figure 13 shows, when λ is equal to 0.75, the macro average F1 attains the maximum value at 0.9954 and accuracy achieves the top value at 99.54% as well in the local test. However, in the global test, when λ is 0.5, the macro average F1 attains the maximum value at 0.9240 and accuracy achieves the top value at 98.90%.

B. COMPARATIVE STUDY

XLNet with BitFit tuning method that performed the worst in the federated learning and then is improved by transfer learning is selected as the representative of FedITD to compare with other methods of insider threat detection. Only one insider threat detection solution based on federated learning is found to compare. This solution uses AutoEncoder to perform the detection [39]. This study also compares FedITD with insider threat detection methods based on central training on the CERT R6.2 dataset. These methods include traditional machine learning algorithms, such as Isolation Forest [56] and one-class SVM [27], and recent deep learning models,

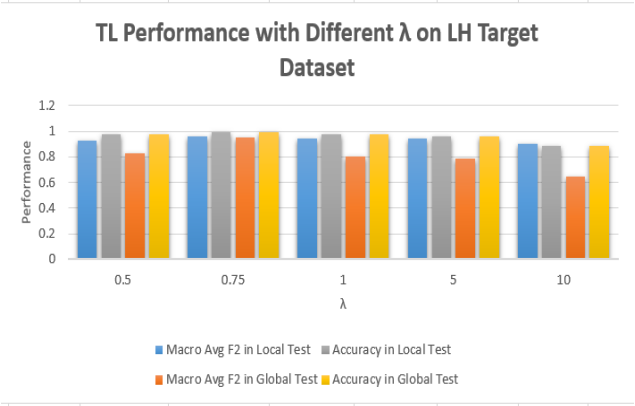

FIGURE 12. TL Performance with different λ on LH Target Dataset.

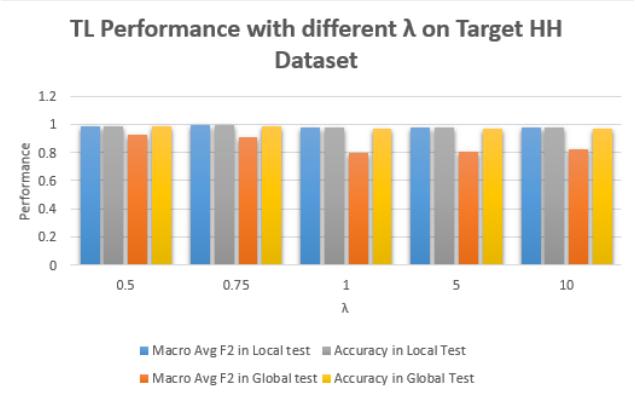

FIGURE 13. TL Performance with different λ on HH Target Dataset.

TABLE 9. The Performance comparison of insider threat methods.

Category	Method	Accuracy	Precision	Recall	F1	AUC
Federated Learning	FedITD	99.54%	92.81%	97.81%	95.17%	97.81%
	Federated AutoEncoder	X	X	97.00%	X	93%
Central Learning	DistilledTrans	99.82%	100.00%	93.33%	96.55%	99.63%
	LSTM-RNN	93.85%	95.12%	92.46%	X	X
	DeepMIT	X	91.60%	93.20%	93.20%	X
	DD-GCN	98.65%	X	X	85.69%	X
	Multistage LSTM +CNN	85.00%	X	X	X	90.47%
	Hierarchical LSTM	X	X	X	X	94.96%
	Simultaneous Neural Learning	X	X	X	X	95.60%
	Unsupervised Ensembles	X	X	X	X	97.70%
	Log2vec	X	X	X	X	86.00%
	Log2vec++	X	X	X	X	93.00%
	Peer Group Metadata-Informed LSTM Ensembles	X	X	X	X	81.40%
	LSTM	X	X	X	X	71.00%
	OCSVM	X	X	X	X	73.67%
	IForest	X	X	X	X	76.19%

such as DistilledTrans [7], LSTM-RNN [57], DeepMIT [58], DD-GCN [36], multistate LSTM + CNN [59], Hierarchical LSTMs [32], simultaneous neural learning [60], unsupervised ensembles [61], log2vec [62], log2vec++ [62], and peering group metadata-informed LSTM Ensembles [63].

The comparative study results show that FedITD not only outperforms Federated AutoEncoder but also defeats all insider threat detection methods based on central training except DistilledTrans [7]. However, the gap between FedITD and DistilledTrans is very narrow: only 1.82% lower in AUC and 1.38% lower in F1 but 4.48% higher in recall.

Our proposed FedITD can indirectly combine data from other clients and reduce the discrepancy of data distribution between source and target domain, therefore the anomaly detection rate is even higher than DistiledTrans which utilizes an optimized transformer model and possesses the advantage of centrally accessing all data samples. Compared to almost all other detection methods whether central trained or federated learned, our method has a great improvement on all performance metrics. It is worth noting that other methods often filter or manipulate the dataset or the model setting to achieve satisfying results. For instance, Unsupervised Ensemble [61] and Federated AutoEncoder [39] require human efforts to examine the highest-ranked data to verify if the detection is true positive. The security analysts have to investigate 20% of data instances post-training to get their claimed performance. Additionally, different thresholds set by humans can generate different performance results. For example, log2vec [61] achieves an AUC of 0.93 by evaluating only 6 potential malicious users and 12 normal users from the CERT R6.2 dataset. In comparison, our proposed method is a complete end-to-end solution without any human intervention or manipulation. Additionally, FedITD has no such limitations on the number of users. Our higher AUC is achieved by training and testing using the complete dataset. One benefit of FedITD is the capability to add new clients automatically, eliminating the need for rebuilding the model and incremental algorithms. In the meantime, FedITD has strong detection performance on client local data that other methods do not have. In brief, FedITD proves its effectiveness in identifying insider anomalies.

VI. CONCLUSION

In this paper, we present FedITD, a pioneering framework for insider threat detection that combines FL, PETuning with Pre-trained LLMs, and transfer learning. FedITD enables indirect integration of information across clients through FL. To reduce resource costs, minimize time delays, and protect privacy, this study explores three representative PETuning methods—Adapter, BitFit, and LoRA—applied to four pre-trained large language models: BERT, RoBERTa, XLNet, and DistilBERT. The results show that Federated PETuning methods maintain satisfactory performance while effectively lowering resource usage and protecting against privacy breaches. Subsequently, we construct custom client models through transfer learning unsupervised domain adaptation. Experimental results reveal that FedITD addresses the existing issues effectively: **1) Detection Performance**: FedITD's detection performance surpasses that of other federated methods and performs better than nearly all centrally trained methods, closely approaching the best central training method DistilledTrans. **2) Efficiency**: It significantly reduces communication costs, storage overhead, memory usage, and time delays. **3) Adaptability and Generalization:** It demonstrates excellent adaptability to both slightly and highly heterogeneous data and has a strong ability to generalize, detecting unknown attacks that have not appeared

in the client's local data. **4) Optimal Equilibrium:** By tuning system parameters, FedITD can achieve an optimal balance in TL performance across source domain data, target domain data (unlabeled local data), and global testing. In the future, incorporating differential privacy will further enhance the system's privacy and security protections. Additionally, leveraging our system to train data online can further improve the system by allowing continuous adaptation and learning from new data.

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.

REFERENCES

[1] (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

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[3] 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*.

[4] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.

[5] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," 2019, *arXiv:1910.01108*.

[6] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," 2018, *arXiv:1804.07461*.

[7] Z. Q. Wang and A. E. Saddik, "DTITD: An intelligent insider threat detection framework based on digital twin and self-attention based deep learning models," *IEEE Access*, vol. 11, pp. 114013–114030, 2023.

[8] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for NLP," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2790–2799.

[9] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," 2021, *arXiv:2106.09685*.

[10] E. B. Zaken, S. Ravfogel, and Y. Goldberg, "BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," 2021, *arXiv:2106.10199*.

[11] Z. Zhang, Y. Yang, Y. Dai, Q. Wang, Y. Yu, L. Qu, and Z. Xu, "FedPETuning: When federated learning meets the parameter-efficient tuning methods of pre-trained language models," in *Proc. Findings Assoc. Comput. Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2023, pp. 9963–9977.

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

[13] 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.

[14] 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.

[15] M. T. Sandıkkaya, Y. Yaslan, and C. D. Özdemir, "DeMETER in clouds: Detection of malicious external thread execution in runtime with machine learning in PaaS clouds," *Cluster Comput.*, vol. 23, no. 4, pp. 2565–2578, Dec. 2020.

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

[17] 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. Dependable Appl.*, vol. 1, no. 1, pp. 3–13, 2010.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] D. C. Le, N. Zincir-Heywood, and M. I. Heywood, "Analyzing data granularity levels for insider threat detection using machine learning," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 1, pp. 30–44, Mar. 2020.

[23] 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.

[24] 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.

[25] P. Zhang, X. Yang, and Z. Chen, "Neural network gain scheduling design for large envelope curve flight control law," *J. Beijing Univ. Aeronaut. Astronaut.*, vol. 31, no. 6, pp. 604–608, 2005.

[26] Y. Xinying, G. Guanghong, T. Yuan, and Y. Xiaoxia, "Generalized optimal game theory in virtual decision-makings," in *Proc. Chin. Control Decis. Conf.*, vol. 2001, Jul. 2008, pp. 1960–1964.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] F. Yuan, Y. Cao, Y. Shang, Y. Liu, J. Tan, and B. Fang, "Insider threat detection with deep neural network," in *Proc. 18th Int. Conf. Comput. Sci. (ICCS)*, in 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.

[32] 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.

[33] 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.

[34] 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.

[35] J. Jiang, J. Chen, T. Gu, 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. IEEE Mil. Commun. Conf. (MILCOM)*, Nov. 2019, pp. 109–114, doi: 10.1109/MILCOM47813.2019.9020760.

[36] 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.

[37] 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.*, vol. 30, 2017.

[38] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, vol. 54, 2017, pp. 1273–1282.

[39] M. Amiri-Zarandi, H. Karimipour, and R. A. Dara, "A federated and explainable approach for insider threat detection in IoT," *Internet Things*, vol. 24, Dec. 2023, Art. no. 100965.

[40] L. Qu, Y. Zhou, P. P. Liang, Y. Xia, F. Wang, E. Adeli, L. Fei-Fei, and D. Rubin, "Rethinking architecture design for tackling data heterogeneity in federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 10051–10061.

[41] O. Weller, M. Marone, V. Braverman, D. Lawrie, and B. Van Durme, "Pretrained models for multilingual federated learning," 2022, *arXiv:2206.02291*.

[42] G. Sun, M. Mendieta, T. Yang, and C. Chen, "Exploring parameter-efficient fine-tuning for improving communication efficiency in federated learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2022.

[43] J. Chen, W. Xu, S. Guo, J. Wang, J. Zhang, and H. Wang, "FedTune: A deep dive into efficient federated fine-tuning with pre-trained transformers," 2022, *arXiv:2211.08025*.

[44] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao, "FedHealth: A federated transfer learning framework for wearable healthcare," *IEEE Intell. Syst.*, vol. 35, no. 4, pp. 83–93, Jul. 2020.

[45] Y. Fan, Y. Li, M. Zhan, H. Cui, and Y. Zhang, "IoTDefender: A federated transfer learning intrusion detection framework for 5G IoT," in *Proc. IEEE 14th Int. Conf. Big Data Sci. Eng. (BigDataSE)*, Dec. 2020, pp. 88–95.

[46] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *Proc. AAAI Conf. Artif. Intell.*, 2016, vol. 30, no. 1.

[47] B. Sun and K. Saenko, "Deep CORAL: Correlation alignment for deep domain adaptation," in *Proc. Eur. Conf. Comput. Vis. (ECCV) Workshops*, Amsterdam, The Netherlands. Cham, Switzerland: Springer, Oct. 2016, pp. 443–450.

[48] H. Liu, S. Zhang, P. Zhang, X. Zhou, X. Shao, G. Pu, and Y. Zhang, "Blockchain and federated learning for collaborative intrusion detection in vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 6073–6084, Jun. 2021.

[49] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 1999, pp. 223–238.

[50] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," 2019, *arXiv:1901.02860*.

[51] L. Muñoz-González, K. T. Co, and E. C. Lupu, "Byzantine-robust federated machine learning through adaptive model averaging," 2019, *arXiv:1909.05125*.

[52] P.-C. Cheng, K. Eykholt, Z. Gu, H. Jamjoom, K. R. Jayaram, E. Valdez, and A. Verma, "Separation of powers in federated learning (poster paper)," in *Proc. 1st Workshop Syst. Challenges Reliable Secure Federated Learn.*, vol. 13, Oct. 2021, pp. 16–18.

[53] *OpenDelta*. Accessed: Apr. 25, 2024. [Online]. Available: https://github.com/thunlp/OpenDelta

[54] *FedLab*. Accessed: Apr. 25, 2024. [Online]. Available: https://github.com/SMILELab-FL/FedLab

[55] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[56] B. Lv, D. Wang, Y. Wang, Q. Lv, and D. Lu, "A hybrid model based on multi-dimensional features for insider threat detection," in *Proc. Wireless Algorithms, Syst., Appl.*, vol. 13. Tianjin, China: Springer, Jun. 2018, pp. 333–344.

[57] 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.

[58] 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.

[59] 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.

[60] L. Liu, C. Chen, J. Zhang, O. De Vel, and Y. Xiang, "Insider threat identification using the simultaneous neural learning of multi-source logs," *IEEE Access*, vol. 7, pp. 183162–183176, 2019.

[61] D. C. Le and N. Zincir-Heywood, "Anomaly detection for insider threats using unsupervised ensembles," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 1152–1164, Jun. 2021.

[62] 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.

[63] J. Matterer and D. LeJeune, "Peer group metadata-informed LSTM ensembles for insider threat detection," in *Proc. 31st Int. Flairs Conf.*, 2018, pp. 1–6.

ZHI QIANG WANG (Member, IEEE) received the B.C.S. degree from 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 AI, cybersecurity, deep fraud, natural language processing, digital twins, and AR/VR.

HAOPENG WANG (Member, IEEE) received the B.Eng. degree in information and electronics and the M.Eng. degree in electronic and communication engineering from Beijing Institute of Technology, Beijing, China, in 2015 and 2017, respectively. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Ottawa. His research interests include AI, computer networks, extended reality, and multimedia.

ABDULMOTALEB EL SADDIK (Fellow, IEEE) is currently a Distinguished Professor and the University Research Chair with the School of Electrical and Engineering and the Computer Science, University of Ottawa, and MBZUAI. He is an ACM Distinguished Scientist. He has authored and co-authored 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. He has supervised more than 120 researchers and received several international awards. His research interests include digital twin, AR/VR, and tactile internet. 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.