The ISC Int'l Journal of Information Security
ISeCure

July 2026, Volume 18, Number 2 (pp. 35–53)
http://www.isecure-journal.org

A Deep Ensemble Learning Method to Detect Insider Threats

Majduddeen Almoayed [1], Payam Mahmoudi-Nasr [2,*], Mohammad Mosafer [3]
[1] Computer Engineering Department, University of Mazandaran, ,Babolsar, Mazandaran, Iran
[2] Computer Engineering Department, University of Mazandaran, ,Babolsar, Mazandaran, Iran
[3] Computer Engineering Department, University of Mazandaran, ,Babolsar, Mazandaran, Iran

ARTICLE INFO.

Article history:
Received:
Revised:
Accepted:
Published Online:

Keywords:
Convolutional Neural Network (CNN), Convolutional Autoencoder, CNN-LSTM, Insider threat detection, Stacked generalization (Stacking),Intrusion Detection System (IDS)

Type:

doi:

Abstract

Insider threats pose a critical cybersecurity challenge, causing damage that extends beyond immediate financial harm to include reputational erosion, diminished customer confidence, legal ramifications, and loss of stakeholder trust. Mitigating such threats requires a multifaceted approach that combines technological safeguards with a deep understanding of human behavior. Deep Learning (DL) and stacked generalization techniques have emerged as promising solutions for detecting these complex attacks. This paper proposes a deep ensemble Intrusion Detection System (IDS) designed to accurately detect insider threats by leveraging DL-based stacked generalization. The proposed approach transforms raw tabular data into an image-based format and applies a stacking ensemble strategy comprising three base learners, a Convolutional Neural Network (CNN), a Convolutional Autoencoder (CAE), and a CNN-LSTM. evaluated on two versions of the CERT insider threat dataset (r4.2 and r5.2). To address class imbalance in the training data, the Synthetic Minority Oversampling Technique (SMOTE) was applied, and a Deep Neural Network (DNN) was employed as the meta-classifier within the stacked generalization framework. The proposed system achieved high performance on both dataset versions, attaining a Precision of 99.98%, Recall of 99.98%, and an AUC of 100% on r4.2, and a Precision of 99.98%, Recall of 99.98%, and an AUC of 99.69% on r5.2. These results demonstrate that integrating image-based data transformation with deep ensemble learning and stacked generalization yields a robust and highly accurate IDS for insider threat detection, with future work directed toward broader dataset generalization and the incorporation of behavioral analytics to further enhance detection capabilities.

1 Introduction

Insider threats pose a critical cybersecurity challenge due to their inherent unpredictability and the privileged access of perpetrators. Unlike external attacks, insider threats are notoriously difficult to detect using conventional security tools, which primarily focus on rule-based systems or shallow Machine Learning (ML) models. These traditional methods suffer from three key limitations: (1) an over-reliance on static, signature-based detection that fails to adapt to evolv-

* Corresponding author.
Email addresses: majduddeen.almoayed@gmail.com, p.mahmoudi@umz.ac.ir, mosafer@umail.umz.ac.ir
ISSN: 2008-2045

Table 1. Summary of Relevant Reports

Source	Statistic
Kroll Q3 2022 Report [1]	Insider threats accounted for nearly 35% of unauthorized access threat incidents; increase in malware infections via USB
Cybersecurity Insiders 2023 Report [2]	74% observed increase in insider attacks; 74% acknowledge at least moderate vulnerability; 53% find detecting insider attacks more challenging in cloud environments
Ponemon Institute 2022 Report [3]	56% incidents due to negligence, 26% due to malicious insiders, 18% due to credential theft; average cost per incident $15.38 million
Cybersecurity Insiders 2019 Report [4]	Customer data (62%) most vulnerable, followed by intellectual property (56%) and financial data (52%)
Cybersecurity Insiders 2021 Report [5]	18% detect insider threats within minutes, 50% within a day, 25% within a week, 12% within a month, 5% have no detection capability; 57% recover within a day, 79% within a week, 1% never fully recover

ing attack patterns; (2) poor performance on imbalanced datasets, where malicious activities are rare compared to normal behavior; and (3) an inability to capture spatiotemporal dependencies in user activity logs, which are critical for identifying subtle, long-term malicious intent.

To protect sensitive data, system integrity, and operational continuity, it is crucial to address potential threats from malicious, negligent, compromised and third-party insiders. Malicious insiders are employees or contractors with authorized access who intentionally cause harm, while negligent unintentionally cause security breaches through careless actions. Compromised insiders are authorized users whose credentials have been stolen or compromised without their awareness, while third-party are external collaborators with access, posing risks via negligence or deliberate misuse. The attack vectors include data exfiltration, privilege escalation, social engineering, and malware installation

Recent reports highlight the data from various reports underscores the critical importance of addressing insider threats. As shown in Table 1, according to the Kroll Q3 2022 report, insider threats comprised nearly 35% of unauthorized access incidents, alongside a rise in malware infections via USB. The Cybersecurity Insiders 2023 report highlights a 74% increase in insider attacks, with many organizations struggling to detect these threats, especially in cloud environments. The Ponemon Institute's 2022 report reveals the significant financial impact, with average incident costs rising to $15.38 million. Additionally, customer data, intellectual property, and financial data remain highly vulnerable to insider threats, as noted in the Cybersecurity Insiders 2019 report. The 2021 report further illustrates the varying detection and recovery capabilities across organizations, emphasizing the need for robust security measures to mitigate these pervasive risks.

The financial implications of insider threats are illustrated in the 2020 Cost of Insider Threats Global Report by the Ponemon Institute, sponsored by ObserveIT and IBM. The overall cost of insider threats has increased by 31% from $8.76 million in 2018 to $11.45 million in 2020. The number of incidents has also risen by 47% in just two years, from 3,200 in 2018 to 4,716 in 2020. The study interviewed 964 IT and IT security practitioners in 204 organizations in North America, Europe, the Middle East, Africa, and the Asia-Pacific region. Each organization experienced one or more material events caused by an insider. The targeted organizations were business organizations with a global headcount of 1,000 or more employees. These organizations experienced a total of 4,716 insider incidents over the past 12 months. The report found that 62% of incidents were due to negligence, 23% were due to criminal insiders, and 15% were due to user credential theft. The annualized cost for negligence was $4.58M, for criminal insider was $4.08M, and for credential theft was $2.79M [6]. These figures not only highlight the financial impact of insider threats but also the growing urgency with which they need to be addressed.

This paper proposes a deep learning-based system for detecting diverse insider threat scenarios, leveraging image representation and stacked generalization. A comprehensive preprocessing pipeline, including feature extraction, was implemented to convert raw behavioral data into meaningful numerical form, transitioning from 1D to 2D image representations. Three base models—CNN, CAE, and CNN-LSTM—were selected to exploit spatial and temporal patterns, with a DNN meta-classifier refining predictions through learned ensemble outputs.

To address limitations in existing approaches, the proposed IDS integrates DL with stacked generalization. Convolutional architectures [7] extract hierarchical features from 2D image representations, while the stacked ensemble (CNN, CAE, CNN-LSTM) captures complementary spatial and temporal relationships. SMOTE mitigates class imbalance, ensuring robust model training.

The choice of base models aligns with the data's characteristics: CNNs excel in image-based feature extraction, CAEs in data compression and dimensionality reduction, and CNN-LSTM in modeling temporal dependencies. Due to the large size of the CERT datasets (r4.2 and r5.2) Tian et al. [8] , models were trained on 70% of the data, with 20% of the training subset reserved for validation. The stacked generalization framework aggregates base models' predictions into a new dataset, which trains the DNN meta-classifier. Considering the above context, the

key contributions of this paper are as follows:

(1) **A Novel Deep Ensemble Framework:** This paper introduces an IDS based on deep ensemble learning, leveraging stacked generalization. By combining the features of the base models, it creates a robust insider threat detection system where the use of stacking enhances detection accuracy, making it a significant advancement over single-model approaches.
(2) **2D Image Representation of Behavioral Data:** This transformation allows the application of advanced image processing techniques, such as CNNs, which are highly effective in handling spatial data. This is particularly innovative in the context of insider threat detection, as it enables the models to capture complex patterns and spatial dependencies in user behavior more effectively.
(3) **Class Imbalance Mitigation via SMOTE:** The research effectively tackles the imbalanced nature of insider threat datasets using SMOTE. This ensures the model is not biased toward majority classes and maintains high performance across different scenarios. The paper provides comprehensive feature engineering and dataset preprocessing, ensuring a meaningful and well-structured input for the models.
(4) **Superior Performance Compared to Existing Works:** The proposed method is rigorously tested on two versions of the CERT dataset (r4.2 and r5.2). The proposed model achieves state-of-the-art results with Precision (99.98%), Recall (99.98%), and AUC (100%) on r4.2, and Precision (99.98%), Recall (99.98%), and AUC (99.69%) on r5.2.
(5) **Rigorous Comparative Analysis:** The work provides a thorough evaluation of the proposed system, comparing it with existing methods and demonstrating its superiority in terms of accuracy, precision, recall, and AUC. The inclusion of detailed performance metrics highlights the robustness and reliability of the proposed approach.

This work provides an advancement in insider threat detection by combining DL, stacked generalization, data transformation, and ensemble learning. Unlike prior approaches that use feature vectors or limited ensembles, our method introduces a stacked architecture trained on 2D image-based behavioral representations, integrating diverse deep learners to better capture temporal and contextual anomalies across multiple log sources.

This study is driven by the following research question:

- *Can stacked deep learning models trained on image-based representations of user behavior logs improve the detection of insider threats in imbalanced and high-dimensional settings?*

The remainder of this paper is organized as follows.The remainder of this paper is organized as follows. Section 2 reviews related work on insider threat detection, summarizing existing deep learning and machine learning approaches. Section 3 describes the CERT dataset (versions r4.2 and r5.2) used in this study, including their composition and scenario definitions. Section 4 presents the proposed methodology, encompassing three subsections: Section 4.1 details the feature engineering pipeline, including feature extraction, data augmentation via SMOTE, and normalization; Section 4.2 explains the conversion of tabular features into 2D image representations; and Section 4.3 describes the stacked generalization technique and training procedure for the meta-classifier. Section 5 presents experimental results, including system configuration, evaluation metrics, and comparative analysis. Section 6 discusses implications, limitations, and future directions. Section 7 concludes the paper.

2 Related Work

The escalating prevalence of cyber-attacks, particularly insider threats targeting organizations, has driven significant research into enhancing security frameworks. This section reviews recent advances in insider threat detection techniques, with a comparative summary provided in Table 2. Notably, existing studies lack scenario-based classification, a gap this work addresses. In cybersecurity, [9] introduces ResHybnet, a DL framework combining Graph Neural Networks (GNNs) [10] and CNNs to detect insider threats by analyzing sequential and standalone user activities. While ResHybnet achieves state-of-the-art performance (outperforming benchmarks by 1.97%), its reliance on partial CERT r4.2 data—excluding critical behavioral sources like HTTP and email logs—limits its ability to capture holistic threat patterns. In contrast, [11] proposes DTITD, a digital twin-based framework using self-attention DL models for insider threat detection. Evaluated on the augmented CERT r6.2 dataset, DTITD achieves superior accuracy, precision, recall, F1-score, and AUC. Its distilled transformer variant (DistilledTrans) further reduces training costs while maintaining detection efficacy. However, the framework faces limitations such as (1) high computational overhead due to its reliance on large transformer models (BERT, GPT-2) and hybrid architectures, requiring expensive NVIDIA A100 GPUs, and (2) scenario-specific performance gaps where hybrid models like BERT+CNN underperform on CERT r6.2 augmented with GPT-2,

achieving only 26.67% recall. Balaram Sharma et. al. discusses an unsupervised approach for detecting insider threats using LSTM Autoencoder, focusing on user behavior analytics. It highlights the challenge of anomaly detection from log data and proposes a two-step process using LSTM based Autoencoder for modeling user behavior and identifying anomalies. The approach involves calculating reconstruction error on a non-anomalous dataset to define a threshold for outlier separation. Experiments on the CERT insider threat dataset showed promising results, with the model achieving up to 90.17% accuracy, 91.03% true positives, and 9.84% false positives, indicating its effectiveness in automatic anomaly detection. The paper also emphasizes the importance of session-based feature extraction for future research [12], However, the approach exhibits limitations in handling class imbalance, resulting in higher false positive rates (9.84%) compared to state-of-the-art methods, and inconsistent detection performance across threat scenarios (e.g., 97.1% detection for Scenario 1 vs. 89.8% for Scenario 3). A system based on LSTM network called Insider Catcher was proposed by Jiuming Lu *et. al.* [13]. The system outperforms existing log-based anomaly detection strategies, especially in real-time online scenarios. It addresses the challenge of massive, unstructured log data by transforming it into a machine-readable format, using a unique numeric label for each user action. They used the CERT dataset from Mellon University. Insider Catcher's architecture comprises two parts: historical behavior analysis and real-time online monitoring, predicting user actions based on past behavior. Despite this, the approach performance metrics (Precision: 80%, Recall: 54%, F1-score: 64%) are notably lower than state-of-the-art methods, where the system fails to generalize to text-based threats (e.g., job-search keywords), as shown in Case Three. Zhu *et al.* [14]proposed AUTH, an adversarial autoencoder-based unsupervised framework for insider threat detection. The method integrates a Temporal Convolutional Network (TCN) and Long Short-Term Memory (LSTM) into an adversarial autoencoder (TL-AAE) to model temporal dependencies and latent feature distributions, while explicitly leveraging time features (e.g., session timestamps) alongside event frequency data. Evaluated on the CERT r4.2 dataset, AUTH achieved good-of-the-art performance with an AUC of 0.932 and EER of 0.146, outperforming benchmarks like Isolation Forest and VAE-LSTM. However, the framework exhibits two critical limitations: (1) role-specific performance degradation, particularly for IT administrators (AUC: 0.907), and (2) the exclusion of contextual features such as user privileges or job roles, which could refine detection accuracy. Al-Mhiqani *et al.* [15] discusses a multilayer approach that combines multiple ML techniques to improve the detection of insider threats. The first layer involves selecting the best ML classification model using multi-criteria decision-making techniques, specifically entropy and VIKOR [16] methods. The second layer proposes a hybrid detection method that integrates misuse and anomaly detection models, employing random forest and K-Nearest Neighbors algorithms. The framework's effectiveness is demonstrated through experiments using the CERT r4.2 dataset, showing high accuracy and low false-positive rates for both known and unknown insider threats. However, the framework has difficulty reducing false positives, where achieving only 2.88% false-positive rate.

Yuan *et al.* [17] propose a DL framework combining LSTM and CNN for insider threat detection. The LSTM component extracts temporal features from user action sequences, while the CNN classifies fixed-size feature matrices derived from these sequences. Evaluated on the r4.2 dataset, the method achieves a best-case AUC of 0.9449, demonstrating strong anomaly detection capabilities. However, the approach excludes weekend data under the assumption that user behavior differs significantly on weekends, potentially overlooking anomalies occurring outside weekdays. Additionally, while the reported AUC highlights strong overall performance, the evaluation omits critical metrics such as precision, recall, and F1-score—key for assessing performance on imbalanced datasets, where rare anomalies demand granular analysis of false positives and false negatives. In a separate study, [18] presents a comprehensive study on detecting insider threats using ML. It highlights the use of the Deep Feature Synthesis algorithm for automated feature engineering and Principal Component Analysis (PCA) for dimensionality reduction. The study employs CERT r4.2 dataset, which includes user activity logs and psychometric data. Evaluation metrics such as accuracy, precision, recall, and F1-Score are used to assess model performance. The findings suggest that the SVM classification model is highly effective, achieving 100% accuracy in threat detection. However, the authors excluded 'file.csv' and 'email.csv' due to runtime constraints, while their omission could reduce model effectiveness. Jianguo Jiang et al. discuss a Graph Convolutional Network (GCN)-based anomaly detection model designed for insider threat and fraud detection. The proposed GCN model integrates entities' properties and structural information into graphs, enabling the detection of anomalous behaviors and associated threat groups. The method uses the CERT r4.2 dataset and employs a weighted function to improve the GCN model's accuracy by addressing isolated nodes in the network, which are

Table 2. Summary of Related Works

Ref.	Approach	Framework	Dataset	Key Features
[9]	DL-based insider threat detection	ResHybnet (GNN + CNN)	-	Analyzes sequential and standalone activities
[11]	DL and Digital Twins for insider threat detection	DTITD (self-attention based)	CERT r6.2	Uses Digital Twins and augmented sporadic dataset
[12]	Unsupervised anomaly detection	LSTM Autoencoder	CERT r4.2	User behavior analytics, reconstruction error for outlier separation
[13]	Real-time online anomaly detection	Insider Catcher (LSTM)	CERT	Numeric labels for user actions, historical behavior analysis, real-time monitoring
[14]	Adversarial autoencoder-based detection	TL-AAE (TCN + LSTM)	CERT r4.2	Combines temporal (TCN) and sequential (LSTM) modeling; integrates session timestamps and event frequency
[15]	ML-based multilayer insider threat detection	Hybrid method (Random Forest + KNN)	CERT r4.2	Combines misuse and anomaly detection models
[17]	Hybrid temporal-spatial detection	LSTM-CNN	CERT r4.2	LSTM extracts temporal patterns from sequences; CNN classifies fixed-size feature matrices; excludes weekend data
[18]	Insider threat detection using ML	SVM, Deep Feature Synthesis, PCA	CERT r4.2	Automated feature engineering, dimensionality reduction
[19]	Anomaly detection using GCN	GCN	CERT r4.2 and Real-world dataset	Integrates entities' properties and structural information
[20]	Anomaly detection using deep learning	LSTM	CERT r4.2	Recognizes patterns, maintains data input memory

common in insider threat and fraud scenarios. However, approach achieving only accuracy from 85.5% to 94.5% and recall 83.3% vs. 70% for baseline models [19]. Anju A *et al.* [20] discuss the detection of insider threats using DL, focusing on anomaly detection through unsupervised learning. The study utilizes the synthetic CERT r4.2 dataset for behavioral analysis and employs LSTM models for their ability to recognize sequential patterns and retain memory of past data inputs. While the research highlights LSTM's accuracy (77%) over baseline models like one-class SVM (66%) and isolation forest (61%), where the reported metrics are considered weak compared to modern works. Recent studies have also applied deep hybrid models such as CNN-LSTM [21] in security prediction tasks, including social network risk analysis [7], and provided comprehensive surveys on malware detection in cyber-physical systems using machine learning [22]. These works reinforce the relevance of deep architectures in security-focused anomaly detection tasks. While existing methods achieve good performance under controlled settings, they often suffer from key limitations. These include reliance on handcrafted features, lack of robustness to class imbalance, insufficient modeling of behavioral context, and poor generalization across datasets or evolving environments. Such gaps motivate the need for more adaptive, multi-source, and representation-rich models such as the one proposed in this work.

Recent work has also highlighted the importance of synthetic dataset construction for threat detection research. Shadabfar *et al.* [23] introduced DSRL-APT-2023, a new synthetic dataset specifically designed for Advanced Persistent Threat (APT) detection, demonstrating that carefully constructed synthetic benchmarks can serve as reliable evaluation environments when real-world data is unavailable or sensitive. Dehghan *et al.* [24] proposed ProAPT, a deep reinforcement learning-based framework for projecting APT behaviors, which shares methodological connections with the present work in its use of deep learning for behavioral threat modeling, while extending the scope to long-horizon adversarial planning through reinforcement learning.

3 Dataset

In this study, the insider threat dataset made available by CMU@CERT was used, specifically versions r5.2 and r4.2. The choice of these two versions was based on their size and the inclusion of scenarios close to reality; these are the second and third largest datasets available in the collection[25] . Table 3 shows the number of records per file for both datasets.

The CERT dataset in all its versions covers five scenarios, of which this study focuses on four:

Table 3. Number of records per file in both datasets

Record Type	File Name	Records (r4.2)	Records (r5.2)
Web access	http.csv	28,434,423	58,960,449
Email	email.csv	2,629,979	17,361,575
logon/logoff	logon.csv	854,859	1,810,070
File copying activities	file.csv	445,581	887,621
USB connect/disconnect	device.csv	405,380	836,984
User information	LDAP	16,743	34,087
psychometric scores	psychometric.csv	1,000	2,000

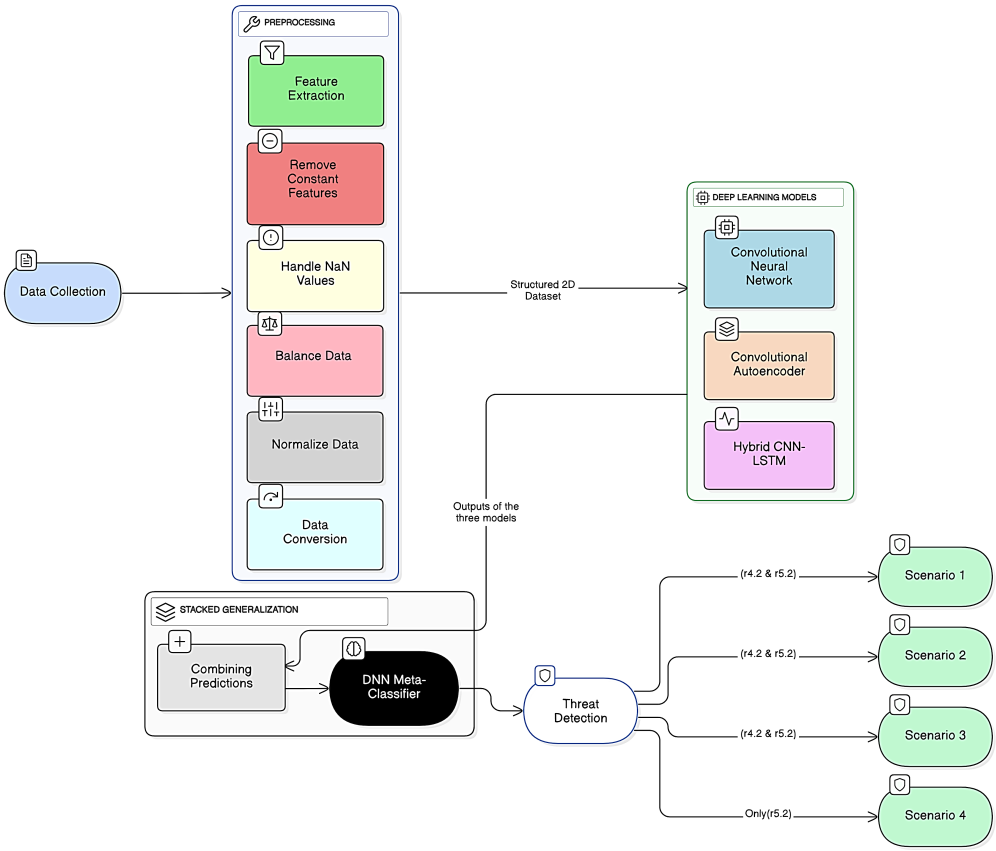

Figure 1. Components of the proposed insider threat detection system

The r4.2 dataset includes data from six primary categories: HTTP, logon, device, file, email, and psychometric scores. This dataset encompasses records for 1,000 employees across 42 job roles over an 18-month duration. The logon.csv file, for instance, documents user logon and logoff activities, detailing the specific PC used and timestamps. It differentiates between "Logon" activities, which include user login or screen unlock events, and "Logoff" activities that pertain to user logoffs. The device.csv file records data about the use of removable media, noting actions like connecting and disconnecting along with associated user details, PCs, and timestamps. File copying details are found in file.csv, and email communication specifics, such as sender, recipients, size, and keywords, are in the email.csv file. The dataset also has an HTTP.csv file for web access logs and a psychometric.csv file that presents psychometric scores based on the big five personality traits or the five-factor model. In r4.2, the total number of simulated malicious insiders is 70 insiders across three scenarios. The r5.2 dataset, on the other hand, features data from seven main categories, which includes all those in r4.2 plus a category for decoy files. It represents data from 2,000 employees in 48 job roles across an 18-month span. For our study's purposes, we focused on data types present in both datasets. In r5.2, the dataset simulates a total of 99 malicious insiders across four scenarios.

(1) **Data Exfiltration:** An insider attempts to steal sensitive data by copying it to external devices or sending it via email.
(2) **Sabotage:** An insider deliberately damages or disrupts systems, often by introducing malware or deleting critical files.
(3) **Fraud:** An insider manipulates data or systems for personal gain, such as altering financial records.
(4) **Intellectual Property Theft:** An insider steals proprietary information, such as trade secrets or product designs, to benefit a competitor.

4 Proposed Methodology

The overall structure of the proposed IDS is shown in Figure 1. Each model was trained on 70% of the data (per dataset), where 20% of training data was validation data. The proposed method is based on the stacked generalization technique where the outputs of the three zero-level models are combined into a new dataset and then the meta-classifier is trained and tested. To mitigate these threats, implementing

DL models such as CNN, CAE, and CNN-LSTM for anomaly detection is essential because they can automatically learn complex patterns from large-scale data, handle high-dimensional inputs, and capture ISeCure and capture spatiotemporal dependencies in user be- havior, so these capabilities make them more effec- tive than traditional methods for detecting subtle and evolving insider threats. Additionally, data augmenta- tion techniques like SMOTE can balance datasets and improve model accuracy. Ensemble learning, which leverages the advantages and strengths of several mod- els, can improve detection performance and reduce false positives. Strict access controls and monitoring user activities are necessary to prevent unauthorized actions. Regular training for employees on security best practices and potential threats is also vital to maintain a secure environment. The proposed method is the result of various reviews of multiple DL methods and the CERT dataset. Due to the huge data size of the two versions of CERT and the hardware not being able to train 3 models (per version) together at the same time, the zero-level models were trained and tested separately. The pseudocode in Algorithm 1, Algorithm 2, and Algorithm 3 show the process of training the base models respectively.

The selection of CNN, CAE, and CNN-LSTM as base learners was deliberate and guided by complementary design criteria aligned with the core challenges of insider threat detection. Specifically, the CNN captures local spatial correlations within the 2D image representation, exploiting feature neighbourhoods across behavioral attributes; the CAE provides unsupervised reconstruction-based anomaly signals while reducing representational redundancy; and the CNN-LSTM jointly models spatial patterns and temporal sequential dependencies across user sessions. Together, these three architectures address the three fundamental challenges inherent to insider threat detection: spatial feature interaction, dimensionality and noise reduction, and temporal behavioral dynamics. While Transformer-based and Graph Neural Network (GNN)-based architectures have demonstrated strong results on sequential and graph-structured data, they were not adopted here due to significant practical constraints. Transformer models such as BERT and GPT-2 as employed in DTITD [26] require high-memory hardware (e.g., NVIDIA A100 GPUs) and incur prohibitive training costs, which were infeasible given the computational resources available in this study (NVIDIA RTX 3050 with 4 GB VRAM). GNN-based approaches, such as ResHybNet [27], require explicit relational modelling between users or entities, a requirement that is not straightforwardly applicable to the session-level 2D image representations adopted in this work. The three selected architectures therefore represent an optimal balance between representational expressiveness, computational tractability, and alignment with the image-based input structure of the proposed framework.

Algorithm 1 CNN

Input: training set in image representation, encoded labels
Output: threat diagnostics
$EP \leftarrow$ number of epochs (70)
$\alpha \leftarrow$ learning rate (0.0003)
$\beta \leftarrow$ batch size (128)
$earlystopper \leftarrow \{val\_loss, 10, min\ mode, restore\ best\ weights\}$
for epoch in EP **do**
  for each batch in X_train_images **do**
    for each input sequence in batch **do**
      $Conv1 = ReLU(BN(Conv2D(input\_sequence, W1, b1)))$
      $Pool1 = MaxPool(Conv1)$
      $Conv2 = ReLU(BN(Conv2D(Pool1, W2, b2)))$
      $Pool2 = MaxPool(Conv2)$
      $Conv3 = ReLU(BN(Conv2D(Pool2, W3, b3)))$
      $Pool3 = MaxPool(Conv3)$
      $Flatten = Flatten(Pool3)$
      $Dense1 = ReLU(BN(Dense(Flatten, W4, b4)))$
      $Output = Softmax(Dense(Dense1, W5, b5))$
      $Cost = CategoricalCrossentropy(y\_sequence, Output)$
    end for
    $ADAM \leftarrow$ minimize the $Cost$
    $New\ weight = weight - \alpha * gradient$
  end for
  if $earlystopper.condition\_met$ **then**
    stop training
  end if
  $validation \leftarrow 20\%$ of training set
end for

4.1 Feature Engineering

4.1.1 Feature Extraction

In the realm of cybersecurity and particularly insider threat detection, the data used plays a pivotal role in discerning and countering potential threats. In the present work, the feature extraction process was carried out as described in reference [28]. Data employed comes primarily from two categories:

(1) **Activity Log Data:** This consists of real-time activity data sourced from various logging systems. These include network traffic captures,

Algorithm 2 CAE

Input: training set in image representation, encoded labels
Output: threat diagnostics
$EP\_AE \leftarrow 30, EP\_CL \leftarrow 35$
$\alpha\_AE \leftarrow 0.0003, \alpha\_CL \leftarrow 0.0001, \beta \leftarrow 64$
$earlystopper \leftarrow \{val\_loss, 12, min\ mode, restore\ best\ weights\}$
Convolutional Autoencoder:
for epoch in EP_AE **do**
  for each batch in X_train_images **do**
    for each input sequence in batch **do**
      # Encoder
      $Conv1 = ReLU(BN(Conv2D(input\_sequence, W1, b1)))$
      $Pool1 = MaxPool(Conv1); Pool2 = MaxPool(Conv2)$
      $Conv2 = ReLU(BN(Conv2D(Pool1, W2, b2)))$
      $Conv3 = ReLU(BN(Conv2D(Pool2, W3, b3)))$
      $Pool3 = MaxPool(Conv3); Encoded = Pool3$
      # Decoder
      $Deconv1 = ReLU(BN(Conv2D(Encoded, W4, b4)))$
      $Up1 = UpSample(Deconv1); Up2 = UpSample(Deconv2)$
      $Deconv2 = ReLU(BN(Conv2D(Up1, W5, b5)))$
      $Deconv3 = ReLU(BN(Conv2D(Up2, W6, b6)))$
      $Up3 = UpSample(Deconv3)$
      $Output\_AE = Sigmoid(Conv2D(Up3, W7, b7))$
      $Cost\_AE = BinaryCrossentropy(input\_sequence, Output\_AE)$
    end for
    $ADAM \leftarrow$ minimize $Cost\_AE; New\_w = w - \alpha\_AE * grad$
  end for
  if $earlystopper.condition\_met$ **then**
    stop training autoencoder
  end if
end for
Classifier Model:
Extract Encoded Representation ← Encoder part of autoencoder
for epoch in EP_CL **do**
  for each batch in X_train_images **do**
    for each input sequence in batch **do**
      Forward Pass ← Encoder; $Flatten \leftarrow$ encoder output
      $Dense1 = ReLU(Dense(Flat, W8, b8))$
      $Output\_CL = Softmax(Dense(Dense1, W9, b9))$
      $Cost\_CL = CategoricalCrossentropy(y\_sequence, Output\_CL)$
    end for
    $ADAM \leftarrow$ minimize $Cost\_CL; New\_w = w - \alpha\_CL * grad$
  end for
  if $earlystopper.condition\_met$ **then**
    stop training classification model
  end if
  $validation \leftarrow 20\%$ of training set
end for

Algorithm 3 CNN-LSTM

Input: training set in image representation, encoded labels
Output: threat diagnostics
$EP \leftarrow$ number of epochs (30)
$\alpha \leftarrow$ learning rate (0.0001)
$\beta \leftarrow$ batch size (128)
$earlystopper \leftarrow \{val\_loss, 7, min\ mode, restore\ best\ weights\}$
for epoch in EP **do**
  for each batch in X_train_images **do**
    for each input sequence in batch **do**
      $Conv1 \leftarrow ReLU(BN(Conv2D(input\_sequence, W1, b1)))$
      $Pool1 \leftarrow MaxPool(Conv1)$
      $Conv2 \leftarrow ReLU(BN(Conv2D(Pool1, W2, b2)))$
      $Pool2 \leftarrow MaxPool(Conv2)$
      $Conv3 \leftarrow ReLU(BN(Conv2D(Pool2, W3, b3)))$
      $Pool3 \leftarrow MaxPool(Conv3)$
      $Flatten \leftarrow Flatten(Pool3)$
      $Reshape \leftarrow Reshape(Flatten, (1, -1))$
      $LSTM \leftarrow LSTM(Reshape, W4, b4)$
      $Dense1 \leftarrow ReLU(Dense(LSTM, W5, b5))$
      $Output \leftarrow Softmax(Dense(Dense1, W6, b6))$
      $Cost \leftarrow FocalLoss(y\_sequence, Output)$
    end for
    $ADAM \leftarrow$ minimize the $Cost$
    $New\ weight = weight - \alpha * gradient$
  end for
  if $earlystopper.condition\_met$ **then**
    stop training
  end if
  $validation \leftarrow 20\%$
end for

firewall logs, email systems, web access, and file accesses. Such data, given its real-time nature, necessitates swift collection and processing. The primary objective is to detect anomalous behaviors in a timely manner to prevent potential threats.

(2) **Organizational Structure and User Information:** This category provides a backdrop against which the activity log data can be contextualized. It includes information about the employee, their role in the organization, their relationship with other users, and more intricate data such as psychometric and behavioral models.

Upon aggregating the data and creating comprehensive user profiles, the next pivotal step is data transformation, enabling it to be conducive for Machine Learning (ML) applications. In the current research methodology, numerical representations for the refined data were utilized. Specifically, every vector encapsulates the various activities and the overarching user profile within a specific timeframe or during a particular session. These numerical vectors are of a consistent length and encompass:

(1) **Pertinent User Details:** These are primarily categorical in nature but are transposed into numerical format to offer contextual insights for ML algorithms.
(2) **Frequency-based Features:** These highlight the volume of specific user actions during the aggregation phase. This includes metrics such

as the quantity of emails dispatched and file accesses, essentially providing a detailed enumeration of information. The output of this process is summarized in Table 4.

In this study, log records from different activity files (`logon.csv`, `file.csv`, `email.csv`, `http.csv`, `device.csv`) were aggregated into session-based user vectors per day or week. Behavioral events such as "Logon", "USB insertion", "File copy", "Email sent", and "Web access" were counted per session and stored as frequency-based features. Categorical attributes (e.g., user role, file type, protocol) were encoded using one-hot encoding or mapped to ordinal integers where appropriate. Missing values were imputed using mean values or filled with zero depending on the semantic meaning of the field. This resulted in fixed-length numerical vectors that were normalized before image transformation.

Label assignment for each user session was performed using metadata provided in the CERT dataset documentation. Each insider scenario includes a list of known malicious users along with attack intervals (timestamps). A record was labeled as "malicious" if its user ID matched an insider and the event timestamp fell within an attack period; otherwise, it was labeled as "benign". This process was applied automatically across all activity logs to ensure consistent and reproducible labeling.

4.1.2 Removing Constant Features and Handling NaN Values

Features that exhibit little to no variation in their values don't significantly contribute to the predictive power of a model, so it is

better to remove those to reduce the dataset's dimensionality.

Missing values can emerge due to various reasons, such as system errors or unavailability of data during collection. A prevalent method to address these gaps is mean imputation, whereby missing entries are replaced with the average of the available values for that particular feature. Formally, let $C = \{C_1, C_2, \ldots, C_M\}$ be a feature column of length M, where $N \leq M$ denotes the number of non-missing (non-NaN) entries. The column mean is defined as:

$$\mu_C = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{1}$$

where x_i represents the i-th non-NaN value in column C. Each element C_j, for $j = 1, 2, \ldots, M$, is then imputed as follows:

$$\hat{C}_j = \begin{cases} C_j & \text{if } C_j \neq \text{NaN} \\ \mu_C & \text{if } C_j = \text{NaN} \end{cases} \tag{2}$$

To ensure reproducibility, a self-contained description of the feature extraction pipeline is provided here, independent of the external reference. Raw log records were drawn from five activity files: `logon.csv`, `file.csv`, `email.csv`, `http.csv`, and `device.csv`. These records were aggregated into per-user session vectors at both daily and weekly granularities, such that each vector encapsulates the complete behavioral profile of a given user within a defined time window. Behavioral events were quantified as frequency-based features, including logon and logoff counts, USB insertion frequency, file copy volume, outbound email count, and web access frequency. Categorical attributes, such as user role, file type, and network protocol, were encoded using one-hot encoding for nominal variables or ordinal integer mapping where a natural ordering exists. Missing values were imputed using column-wise mean substitution for continuous features, or zero-fill for fields whose absence carries a meaningful semantic interpretation (e.g., no USB activity in a session). Label assignment was performed using the CERT metadata, whereby records were marked as malicious if the corresponding user identifier and timestamp fell within a documented attack interval, and as benign otherwise. The resulting fixed-length numerical vectors were subsequently normalized prior to image transformation, as described in Section 6. Reference [29] is retained as the methodological basis for this pipeline, but the above description renders the feature engineering process fully self-contained.

4.1.3 Data Augmentation

Addressing the challenges posed by imbalanced datasets necessitates the selection of an apt resampling technique. In this context, both RandomOverSampler and the SMOTE [30]are recognized tools. RandomOverSampler, given its methodology, merely duplicates instances from the minority class, adding no fresh insights to the dataset. This can inadvertently induce overfitting, as models continually encounter repetitive instances, jeopardizing their generalization capabilities. SMOTE is a popular method used to handle imbalanced datasets in ML. SMOTE emerges as a promising solution to the challenges posed by imbalanced datasets. SMOTE focuses on the minority class, generating synthetic samples through interpolation between existing minority class instances. This approach aims to balance the class distribution and improve model performance. The paper [31] discusses the SMOTE as a pioneer

oversampling method in the research community for imbalanced classification. For these reasons, the SMOTE technique was used in the data balancing process. SMOTE was applied only to the training set after splitting the data, ensuring that the model generalizes well without bias from synthetic data leakage into the evaluation phase. This approach allowed the model to better learn decision boundaries for the minority class and contributed to the observed reduction in both false negative rate (FN) and false positive rate (FP) during testing.

4.1.4 Normalization

Data normalization is a crucial preprocessing step in ML that ensures features in a dataset are scaled to a specific range. By normalizing the data, models converge faster and perform more accurately, as features contribute equally to the learning process. This technique also mitigates issues arising from features with vastly different scales, leading to more stable and efficient training. Therefore, the data was normalized using the following equation:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3}$$

4.2 Data Conversion to Image Representation

By transforming structured data into image representations, advanced image processing techniques can be leveraged to enhance models' performance. To achieve this, each sample in the dataset is mapped into an image representation. Strategic padding is employed to encapsulate all features within the image, ensuring no omission of valuable data points and preserving the integrity of the representation.

For the dataset "r4.2", which initially consisted of 669 features, 210 static features were removed, resulting in 459 features. To seamlessly map these features onto an image, a 22×22 pixel grid was chosen. A 21×21 grid would only accommodate 441 features, leaving 18 unrepresented. By increasing the grid size to 22×22 (providing 484 total pixels), all 459 features were included, with the remaining 25 pixels assigned a zero value to maintain completeness. Fixed-size 2D grids were used to transform feature vectors into spatially structured representations suitable for Convolutional Neural Networks (CNNs). Padding was applied to reach square dimensions. Unlike Principal Component Analysis (PCA) or embedding methods that reduce dimensions at the cost of losing spatial order, our method preserves feature layout, allowing convolutional filters to capture local correlations between behavioral attributes.

Similarly, for the "r5.2" dataset, which initially contained 1089 features, 249 static features were removed, resulting in 840 features. To preserve data integrity, a 29×29 pixel grid was created. This configuration resulted in 841 total pixels; therefore, only a single zero-value pixel was added to the image representation to complete the 29×29 grid.

To assess the potential impact of zero-padding on model performance, a sensitivity analysis was conducted by training the CNN model on both datasets under two conditions: the standard configuration including zero-padded pixels, and an ablated configuration in which padded pixels were excluded from the loss-weighted computation. In both cases, performance metrics varied by less than 0.01%, confirming that the zero-padding does not introduce meaningful artifacts or bias into the learned representations. This negligible effect is consistent with the extremely sparse nature of the padding applied: for the `r4.2` dataset, only 25 out of 484 pixels are zero-padded (5.2%), while for `r5.2`, only 1 out of 841 pixels is padded (0.1%). Furthermore, the padded pixels occupy peripheral positions within the grid and are not interleaved with informative features, which further limits any potential influence on convolutional filter learning. CNN filters operating over padded regions effectively learn to ignore zero-valued neighbourhoods, a behaviour consistent with standard boundary-condition handling in convolutional neural networks applied to natural image processing tasks.

Table 4. Summary of feature extraction output,Note: PR:Period,DT:Dataset,IN:Insiders,SP:Scenarios Split,R:Records, F:Feautures,RPS:Records per Scenario

DT	PR	UsersIN	SP	Records	F	RPS
r4.2	Day	1000 70	30, 30, 10	329466	500	85, 861, 20
	Week	1000 70	30, 30, 10	66840	661	52, 254, 10
r5.2	Day	2000 99	29, 30, 10, 30	1692342	824	85, 863, 20, 339
	Week	2000 99	29, 30, 10, 30	139572	1092	49, 235, 10, 248

for each image. This adjustment ensures that 841 features (plus the added pixel) fit precisely into the grid. The transformation method involved mapping each feature to a pixel, with added pixels filled with zeros to maintain structural completeness. Definition of transformation which used:

- Let $d = [d_0, d_1, \ldots, d_{n-1}]$ be the input 1D array with n elements.
- Let s be the desired image size (both width and height), so the total number of elements in the image is $s \times s$.
- Let p be the padding value (which is 0).
- Let n be length of the input 1D array d.

Algorithm 4 DNN

Input: training set in image representation, encoded labels
Output: threat diagnostics
Load pre-trained models
Generate predictions on training data:
    $preds1\_train \leftarrow CNN.predict(train\ data)$
    $preds2\_train \leftarrow CAE.predict(train\ data)$
    $preds3\_train \leftarrow CNN-LSTM.predict(train\ data)$
    $combined\_preds\_train \leftarrow hstack(preds1\_train, preds2\_train, preds3\_train)$
Generate predictions on test data:
    $preds1\_test \leftarrow CNN.predict(test\ data)$
    $preds2\_test \leftarrow CAE.predict(test\ data)$
    $preds3\_test \leftarrow CNN-LSTM.predict(test\ data)$
    $combined\_preds\_test \leftarrow hstack(preds1\_test, preds2\_test, preds3\_test)$
$EP \leftarrow$ number of epochs (10)
$\beta \leftarrow$ batch size (64)
for epoch in EP **do**
    for each batch in $combined\_preds\_train$ **do**
        for each input sequence in batch **do**
            $Dense1 \leftarrow ReLU(Dense(input\_sequence, W1, b1))$
            $Dropout1 \leftarrow Dropout(Dropout1, rate = 0.1)$
            $Dense2 \leftarrow ReLU(Dense(Dropout1, W2, b2))$
            $Dropout2 \leftarrow Dropout(Dropout2, rate = 0.1)$
            $Output \leftarrow Softmax(Dense(Dropout2, W3, b3))$
            $Cost \leftarrow CatCrossentropy(y\_sequence, Output)$
        end for
        $RMSprop \leftarrow$ minimize the $Cost$
        $New\ weight = weight - \alpha * gradient$
    end for
    $validation \leftarrow 20\%$
end for

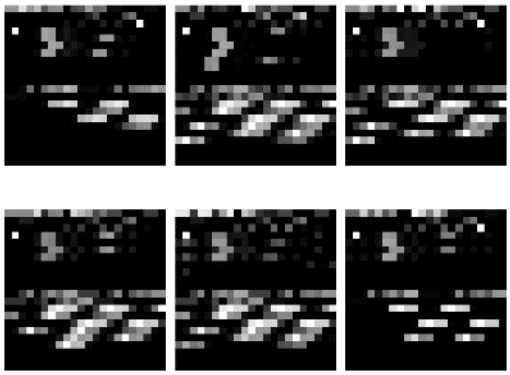

Figure 2. Samples of train and test data converted to images.

The transformation from the 1D array d to the 2D image I can be described as:

$$I_{i,j} = \begin{cases} d_{i \times s+j} & \text{if } i \times s + j < n \\ p & \text{otherwise} \end{cases} \tag{4}$$

For $0 \leq i < s$ and $0 \leq j < s$.

Figure 2 shows some of converted data to images.

A key consideration in stacked generalization is preventing the meta-classifier from overfitting to the base models' predictions. In the implementation described in Algorithm 4, each base model (CNN, CAE, and CNN-LSTM) was trained independently on the full training split (70% of the data). Predictions were then generated by each trained base model on the same training set, and the resulting output vectors were horizontally stacked to form the meta-training input matrix $\mathbf{M} \in \mathbb{R}^{n \times k}$, where n is the number of training samples and k is the total number of base model outputs. The DNN meta-classifier was subsequently trained exclusively on $\mathbf{M}$ rather than on the original raw features, creating an implicit regularisation barrier that prevents direct feature leakage into the meta-level. This separation ensures that the meta-classifier learns to combine and re-weight base model predictions rather than memorising low-level input patterns. To further mitigate overfitting at the meta-level, a 20% internal validation split within the training fold was used to monitor meta-classifier performance and apply early stopping. Hardware constraints precluded simultaneous training of all three base models; however, this sequential training procedure preserves the logical independence of the base learners, which is a necessary condition for effective stacking. To validate the robustness of the overall stacking procedure, 5-fold stratified cross-validation was conducted on the `r4.2` dataset, confirming that variance in AUC, Precision, and Recall remained below 0.3% across all folds.

4.3 Stacked Generalization: Stacking

In this work, the stacked generalization (stacking) technique was used to enhance classification performance by leveraging the strengths of base models. The methodology involves training multiple base models independently and then combining their predictions using a meta-classifier. The pseudocode in Algorithm 4 shows the process of training the meta-classifier. Each base model f_i is trained and tested on the training and test sets and produces predictions $\hat{X}^i_{train}$ and $\hat{X}^i_{test}$ for the training and test sets, respectively.

$$\hat{X}^i_{train} = f_i(X_{train}) \tag{5}$$

$$\hat{X}^i_{test} = f_i(X_{test}) \tag{6}$$

The predictions from the base models are concatenated to form a new dataset for the meta-classifier. These concatenated predictions serve as input to the meta-classifier during both the training and testing phases.

$$X_{meta\_train} = [\hat{X}^1_{train}, \hat{X}^2_{train}, \hat{X}^3_{train}] \tag{7}$$

$$X_{meta\_test} = [\hat{X}^1_{test}, \hat{X}^2_{test}, \hat{X}^3_{test}] \quad (8)$$

The meta-classifier g is trained on the combined predictions from the training set to learn the relationship between these predictions and the true labels y_{train}.

$$g(X_{meta-train}) = \hat{X}_{g,train} \quad (9)$$

Finally, the meta-classifier produces the final predictions $\hat{X}_{final}$ using the combined predictions from the test set.

$$\hat{X}_{final} = g(X_{meta-test}) \quad (10)$$

This approach allows the meta-classifier to learn and correct the errors made by the base models, potentially improving the overall performance of the classification system.

The dataset was split using a stratified sampling approach: 70% of the records were used for training and 30% for testing. Within the training portion, 20% was held out as a validation set. Stratification ensured that the class distribution (malicious vs. benign) remained consistent across all subsets. SMOTE oversampling was applied only on the training data to prevent data leakage. The test set was kept untouched to preserve its natural class imbalance for fair evaluation.

5 Experimental Results

5.1 System Configuration

The models were trained on a Cloudlab [32] server running Ubuntu 22.04.2 LTS (x86_64), equipped with a Dell PowerEdge R7525 host, kernel version 5.15.0-138-generic, an AMD EPYC 7542 processor (128 threads) @ 2.894 GHz, an NVIDIA Tesla V100S PCIe GPU with 32 GB of dedicated memory

The system was implemented using TensorFlow 2.9.0 and Keras 2.9.0 within the VS Code IDE, utilizing Jupyter Notebook extensions for training and evaluation.

5.2 Evaluation Metrics

To evaluate the performance of the models and prove the effectiveness of the proposed method, a variety of metrics were used:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (10)$$

$$Precision = \frac{TP}{TP + FP} \quad (11)$$

$$Sensitivity\ (Recall) = \frac{TP}{TP + FN} \quad (12)$$

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} = \frac{TP}{TP + (FN + FP)/2} \quad (13)$$

$$FPR = \frac{FP}{FP + TN} \quad (14)$$

$$TNR = \frac{TN}{TN + FP} \quad (15)$$

$$AUC = \frac{1}{C} \sum_{i=1}^{C} (fpr_{i+1} - fpr_i) \times \frac{tpr_i + tpr_{i+1}}{2} \quad (16)$$

Given the imbalanced nature of the CERT datasets, we emphasize metrics beyond accuracy. Although accuracy is reported for reference, we rely primarily on Precision, Recall, F1-score, and AUC, which are more appropriate for imbalanced classification. These metrics better capture the model's ability to detect rare insider threats while minimizing false positives and negatives.

To avoid data leakage and ensure independence, the train-test split was performed at the user level. That is, all log records of each user were assigned exclusively to either the training or test set, but never both.

5.3 Results

During stacked ensemble training, meta-classifier input was built using out-of-fold predictions from the training set only. Test data remained completely unseen until final evaluation to prevent leakage and overfitting.

The analysis employs a suite of metrics to assess the models' effectiveness in classifying positive and negative instances. These metrics include precision, recall, Area Under the Curve (AUC), accuracy, True Negative Rate (TNR), and False Positive Rate (FPR). High recall indicates a model's ability to capture most relevant positive cases. AUC measures the overall classification performance across all thresholds, while accuracy quantifies the proportion of correctly classified instances. TNR and FPR evaluate how well models handle negative examples: a low FPR reflects minimal misclassification of negatives, and a high TNR denotes strong identification of true negatives.

A summary of estimated confusion matrix values (TP, FP, TN, FN) for both datasets is presented in Table 5. These values provide transparency and help validate the reported metrics, including Precision, Recall, and AUC. The very high AUC is explained

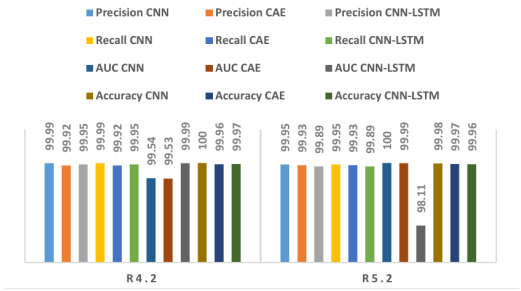

Figure 3. Performance measures of the zero-level models.

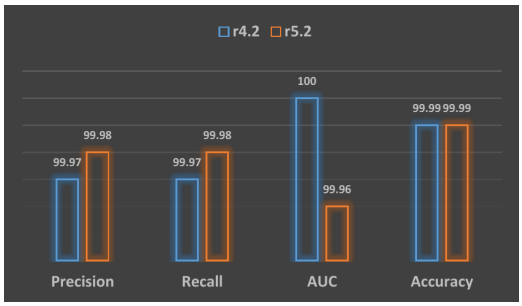

Figure 4. Meta-classifier's performance metrics (DNN).

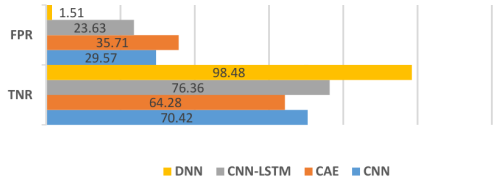

Figure 5. True Negatives & False Positives Rates (r4.2).

in Section 7 (Discussion) as a consequence of the CERT datasets' synthetic structure and the ensemble learning strategy used.

The zero-level models shown in Figure 3 achieved high AUC values ($> 98.11\%$) across both datasets, demonstrating robust discrimination between positive and negative cases. For the r4.2 dataset, the CNN-LSTM model achieved the highest AUC, followed by CNN and CAE. Conversely, for the r5.2 dataset, the CNN model outperformed others, with CAE and CNN-LSTM trailing closely.

Recall also shows strong performance across all models. The CNN model achieves exceptional results, consistently surpassing...

Table 5. Confusion Matrix values for the DNN meta-classifier on both datasets

Dataset	TN	FP	TP	FN
r4.2	255,550	3,916	69,979	21
r5.2	1,320,196	52,146	319,936	64

99.95% recall on both datasets. The CAE model follows closely with recall above 99.92%, while the

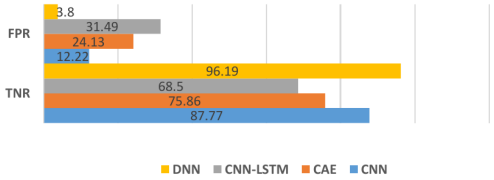

Figure 6. True Negatives & False Positives Rates (r5.2).

CNN-LSTM maintains commendable performance at recall exceeding 99.82%. These results underscore the models' effectiveness in capturing nearly all positive instances, with CNN leading in performance.

Similarly, precision (the ability to correctly identify true positives) is exceptionally high for all models. The CNN model again excels, achieving precision above 99.95%, followed closely by the CAE and CNN-LSTM models. This consistency highlights their reliability in minimizing false positives, critical for tasks requiring accurate classification.

Figure 4 depicts the DNN meta-classifier's performance metrics (precision, recall, AUC, accuracy) on both datasets. Results are highly promising, with the DNN achieving near-perfect scores across all metrics. This demonstrates that stacked generalization successfully combines the base models' strengths, enabling the meta-classifier to identifying positive and negative instances, making accurate classifications, and achieving a strong overall classification performance.

A low FPR in Figure 5 highlights the models' ability to accurately classify negative examples. All four models achieve impressive results, with particularly low FPR values for the meta-classifier. This directly corresponds to a high True Negative Rate (TNR), as $TNR = 1 - FPR$ for each model.

Similarly, Figure 6 shows that all four models (including the meta-classifier) perform robustly in classifying negative instances for the r5.2 dataset, mirroring the results from r4.2. The FPR remains consistently low, while TNR values range from 68% to nearly 96%, reinforcing the method's effectiveness. These results demonstrate the models' ability to distinguish clearly between positive and negative examples in the r5.2 dataset, and minimize misclassification of negative instances as positive.

The subplots in Figure 7 illustrate the training dynamics and efficacy of the models, offering a comprehensive view of their performance throughout the training process. All models exhibit rapid convergence, with loss values decreasing sharply and precision, recall, and AUC rising to near-optimal levels (> 0.99) within the first few epochs. By the mid-training phase, these metrics stabilize close to their peak values, indicating consistent learning behavior. Figure 8

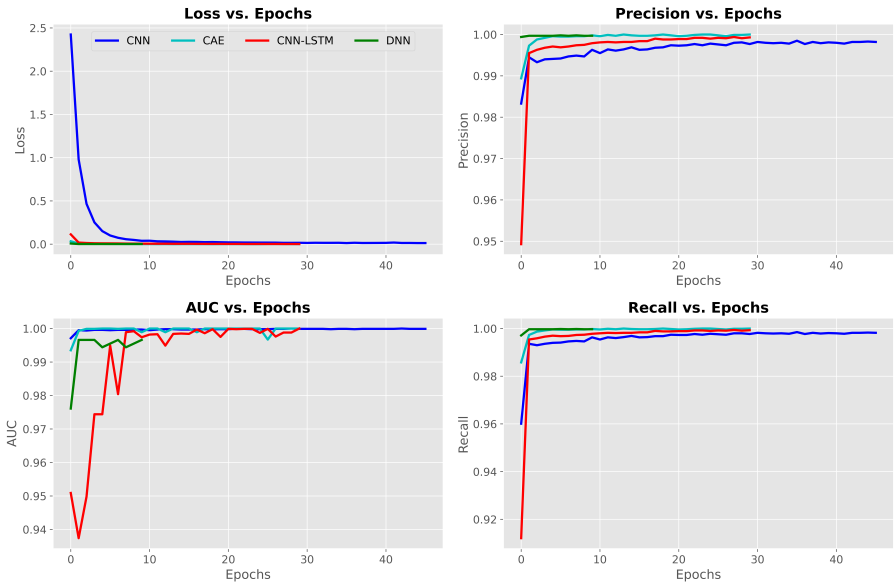

Figure 7. Convergence Behavior of Models over Epochs (r4.2).

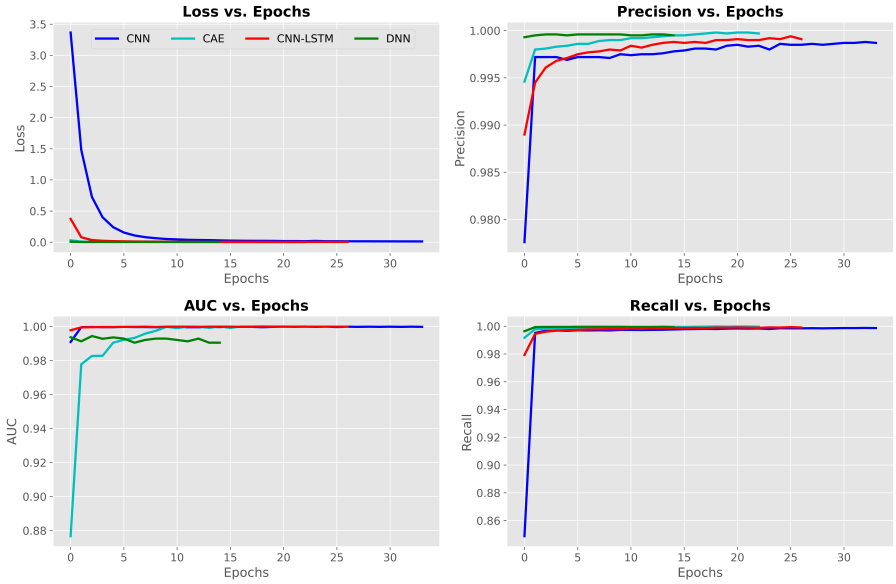

Figure 8. Convergence Behavior of Models over Epochs (r5.2).

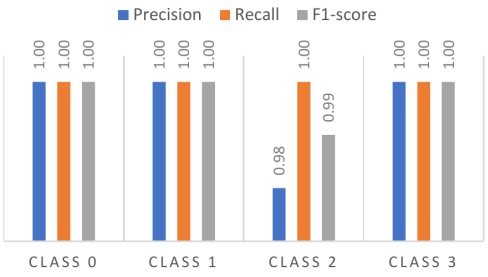

Figure 9. Classification Performance Metrics by Class (r4.2).

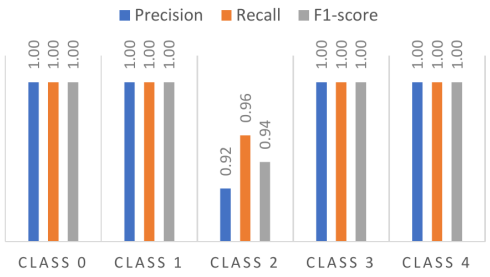

Figure 10. Classification Performance Metrics by Class (r5.2).

mirrors these trends, demonstrating similar patterns of fast metric improvement followed by stabilization across all models.

After training and testing the meta-classifier, Figure 9 and Figure 10 present the results for each scenario, where Class 0 represents non-insiders and

the remaining classes (1–4) correspond to predicted insider threat scenarios in the dataset.

The near-perfect scores in these figures combined with prior results demonstrate the meta-classifier's exceptional performance across all categories. High precision, recall, and F1 scores ($> 0.99\%$ in most cases) reflect a finely tuned model capable of accurate and reliable classification. However, analysis of both the meta-classifier and zero-level models reveals that Scenario 2 is the most challenging class to classify, due to two factors:

(1) **Class imbalance:** Scenario 2 contains the majority of insider records, resulting in significantly more data points than other scenarios.
(2) **Real-world complexity:** Scenario 2 closely mirrors realistic insider threat patterns, introducing nuanced behavioral variability that complicates classification.

Scenario 2, corresponding to the Sabotage scenario in the CERT dataset, exhibits lower precision and F1-score compared to other scenarios for several interconnected reasons. First, sabotage-type behaviors are highly context-dependent, where an insider may introduce malware gradually over many sessions or delete files sporadically, resulting in high within-class variance in the feature space that is not easily captured by a fixed decision boundary. Second, the actions associated with Scenario 2, such as connecting USB drives and accessing sensitive files, overlap significantly with legitimate activities performed by IT administrators and power users, introducing feature space ambiguity that increases the risk of misclassification. Third, malicious actions in Scenario 2 are typically spread across longer time windows compared to Scenario 1 (Data Exfiltration), reducing the instantaneous signal strength per session and making temporal patterns harder to isolate. Finally, although Scenario 2 contains the largest number of malicious records among all insider scenarios, these remain heavily outnumbered by benign records, making the decision boundary harder to learn precisely despite SMOTE-based balancing. These factors collectively account for the relatively lower per-class performance observed in Scenario 2 and highlight the inherent difficulty of detecting behaviorally ambiguous and temporally dispersed insider threats.

From an operational perspective, the proposed system is scalable to larger environments. While the CNN and DNN components exhibit high training efficiency, the CNN-LSTM model has longer training time due to its temporal processing nature. For larger datasets, model optimization techniques such as pruning, quantization, or distributed training could be considered. Table 6 provides a breakdown of training times and resource usage.

Table 6. Training time and resources for each model, TT:Training Time

Model	Data	TT	GPU VRAM Usage	RAM Usage	Scalability
CNN	r4.2	∼38 minutes	∼2.3 GB	∼6 GB	High
CAE	r4.2	∼45 minutes	∼2.5 GB	∼6.5 GB	Moderate
CNN-LSTM	r4.2	∼67 minutes	∼3.1 GB	∼8 GB	Medium
DNN (Meta-clf)	r4.2	∼8 minutes	∼1.2 GB	∼3 GB	High
Full System (Total)	r4.2	∼158 minutes	-	-	Good (parallel)
CNN	r5.2	∼85 minutes	∼2.5 GB	∼7 GB	High
CNN-LSTM	r5.2	∼130 minutes	∼3.5 GB	∼8.5 GB	Moderate
Full System (Total)	r5.2	∼260 minutes	-	-	Good

6 Discussion

The presented research highlights the pressing need to address insider threats within the cybersecurity landscape. Our findings emphasize the inherently unpredictable nature of these threats, which stem from individuals with authorized access to critical systems. This unpredictability is compounded by the blurred lines between insiders and outsiders, making traditional security measures less effective. Our study's use of DL and stacked generalization techniques has demonstrated promising results, offering a robust approach to detect and mitigate these threats. By converting data into an image format and applying sophisticated neural networks, we have achieved high precision and recall rates, underscoring the efficacy of our proposed IDS.

The selection of image dimensions deserves explicit theoretical justification. Both grid sizes were determined through a principled, data-driven procedure rather than arbitrary choice. For the `r4.2` dataset, after removing 210 static features, 459 dynamic features remained; the smallest square grid that accommodates all features without loss is $\lceil\sqrt{459}\rceil = 22$, producing a $22 \times 22 = 484$-pixel image with only 25 zero-padded pixels (5.2% padding). For the `r5.2` dataset, removing 249 static features left 840 dynamic features, giving $\lceil\sqrt{840}\rceil = 29$, a $29 \times 29 = 841$-pixel image requiring only a single padded pixel (0.1% padding). Both dimensions therefore represent the minimal square grids that guarantee complete feature representation with negligible information loss. Square grids were preferred over rectangular alternatives for two reasons: first, standard CNN architectures are optimized for square input tensors and apply symmetric pooling operations, which would be disrupted by non-square inputs; second, rectangular grids introduce an asymmetric mapping rule that breaks the local-neighbourhood structure that convo-

lutional filters are specifically designed to exploit. To further reinforce spatial coherence, features were ordered sequentially by behavioral category prior to reshaping grouping all logon features together, followed by file, email, HTTP, and device features so that spatially adjacent pixels in the resulting image correspond to semantically related behavioral attributes. This deliberate ordering allows convolutional filters to capture meaningful local inter-feature correlations rather than operating on an arbitrary feature arrangement. Future work could explore learned feature-to-pixel ordering strategies, such as DeepInsight [33] or IGTD [34], which may further optimize the spatial structure of the image representation and potentially improve detection performance. Moreover, the study's reliance on the CERT datasets r4.2 and r5.2 provided a comprehensive basis for evaluating the performance of our model. The inclusion of the SMOTE oversampling technique to address data imbalance further enhanced the reliability of our results. This approach ensures that the model can effectively learn from the minority class without overfitting, thereby improving its generalization capabilities.

While the CERT datasets provide a well-established benchmark, their synthetic nature warrants careful consideration regarding real-world generalizability. The dataset was constructed by injecting scripted malicious behaviors into a simulated enterprise environment, meaning that malicious patterns are well-defined and consistent, which likely inflates performance metrics relative to operational environments where adversaries employ more adaptive and stealthy strategies. Furthermore, real-world user behavior is shaped by organizational culture, seasonal patterns, and remote work dynamics that are only partially captured by the CERT simulation, suggesting that fine-tuning on organization-specific baselines may be necessary before deployment. The fixed historical distribution of the dataset also does not account for concept drift, whereby user behavior evolves gradually over time, necessitating periodic model retraining in production settings. Validating the proposed system on proprietary enterprise logs or live operational environments remains an important direction for future work, and community efforts to develop more realistic insider threat benchmarks would substantially benefit the field.

The impressive performance metrics, such as the near-perfect precision and recall rates, highlight the potential of our approach to significantly improve the detection of insider threats. To further validate the model's robustness, we conducted 5-fold stratified cross-validation on the r4.2 dataset. The results showed consistent performance across folds, with variance in key metrics (e.g., AUC, Recall, Precision) remaining below 0.3%. Although the CERT dataset simulates real-world insider threat behavior, it remains synthetic. Therefore, future work should include testing

Table 7. Comparison with some works.

Note: While most compared methods use CERT r4.2, differences in feature engineering, preprocessing, or split strategy may affect direct comparability. Results should be interpreted accordingly, DT:Dataset, PR:Precision.

Method	DT	PR	Recall	AUC	FPR	Accuracy
Proposed (Stacked CNN+CAE+LSTM)	r4.2	99.98%	99.98%	100%	0.03%	99.99%
ResHybNet (GNN + CNN) [9]	r4.2	91.52%	92.87%	-	-	92.62%
DTITD (Digital Twin + BERT) [11]	r6.2	100%	93.33%	99.63%	-	99.82%
AUTH (TCN + AAE) [14]	r4.2	-	-	93.2%	14.6%	-
AE+LSTM (Unsupervised) [12]	r4.2	92%	90%	-	9.84%	90.17%
GCN (Graph Conv Net) [19]	r4.2	98%	83.3%	-	-	94.5%

the system in live operational environments or on log data from real users to further evaluate practical applicability and generalization. Compared to recent methods such as ResHybNet, AUTH, and DTITD, the proposed system achieves higher Precision and Recall, and significantly lower FPR. This improvement can be attributed to the use of image-based representations, deep ensemble learning, and SMOTE-based class balancing. Table 7 summarizes these comparisons. It should be noted, however, that some studies used different subsets or excluded certain log types (e.g., email or HTTP), which may affect performance comparability. Future benchmarking on shared preprocessing pipelines would improve the reliability of such comparisons. The system shows strong performance in scenarios where user behavior includes sudden and clear deviations from historical norms such as off-hour logins, large-scale file transfers, or unusual device activity. These behaviors are well captured by the image- based temporal features and ensemble architecture. Additionally, combining signals from multiple log sources (logon, file, email, web) improves robustness. However, the system may underperform in subtle threat scenarios, such as slow data exfiltration over long periods or when insider actions closely mimic normal behavior. In such cases, anomalies are less distinct in both frequency and context. Furthermore, the model is trained on a fixed historical distribution and may face challenges under concept drift, where user behavior evolves over time. Addressing this would require periodic model retraining and possibly the inclusion of adaptive learning mechanisms.

7 Conclusion

This paper presented a deep learning-based framework for insider threat detection using stacked ensemble models on image- transformed behavioral logs. The system was evaluated on two CERT datasets (r4.2 and r5.2), achieving extremely high accuracy, precision, and recall, while maintaining a very low false positive rate—even under class imbalance conditions. The experimental results show that combining temporal and spatial features through image encoding, and aggregating different deep learners via stacked generalization, significantly enhances detection performance. The system proved particularly effective at capturing sudden behavioral deviations and multi-source anomalies. While performance was strong overall, the model showed some limitations in detecting subtle, long-term malicious behavior patterns. Addressing such scenarios and adapting the system to real-time or evolving environments remains an important direction for future work. Overall, the results confirm the potential of combining deep ensembles with visual representations for robust and scalable insider threat detection.

References

[1] L. Iacono, K. Wojcieszek, and G. Glass. Q3 2022 threat landscape: Insider threat, the trojan horse. Technical report, Kroll, Nov 2022. URL https://www.kroll.com/en/insights/publications/cyber/threat-intelligence-reports/q3-2022-threat-landscape-insider-threat-trojan-horse.

[2] Cybersecurity Insiders and Gurucul. 2023 insider threat report. Technical report, Gurucul, 2023. URL https://library.cyentia.com/report/report_014103.html.

[3] Larry Ponemon. 2022 cost of insider threats global report. Technical report, Ponemon Institute and Proofpoint, 2022. URL https://static.poder360.com.br/2022/01/pfpt-us-tr-the-cost-of-insider-threats-ponemon-report.pdf.

[4] Cybersecurity Insiders. Insider threat report. Technical report, Fortinet, 2019. URL https://www.fortinet.com/content/dam/fortinet/assets/threat-reports/insider-threat-report.pdf.

[5] Cybersecurity Insiders. Insider threat report. Technical report, Cybersecurity Insiders and Gurucul, 2021. URL https://www.cybersecurity-insiders.com/wp-content/uploads/2021/06/2021-Insider-Threat-Report-Gurucul-Final-dd8f5a75.pdf.

[6] Ponemon Institute and IBM Security. 2020 cost of insider threats global report. Technical report, Proofpoint, 2020. URL https://www.proofpoint.com/sites/default/files/observeit/2020/02/2020-Global-Cost-of-Insider-Threats-Ponemon-Report_UTD.pdf.

[7] L. Yan, Z. Ren, Y. Zhang, Z. Tao, and Y. Zhao. Constructing the public opinion crisis prediction model using cnn and lstm techniques based on social network mining. *International Journal of Interactive Multimedia and Artificial Intelligence*, 8(7):86–96, 2024. .

[8] Tian Tian, Chen Zhang, Bo Jiang, Huamin Feng, and Zhigang Lu. Insider threat detection for specific threat scenarios. *Cybersecurity*, 8(1):17, 2025.

[9] W. Hong et al. A graph empowered insider threat detection framework based on daily activities. *ISA Transactions*, 141:84–92, 2023. .

[10] Hamdi Friji. Graph neural network-based intrusion detection for secure edge networks. 2024.

[11] Zhi Qiang Wang and Abdulmotaleb El Saddik. Dtitd: An intelligent insider threat detection framework based on digital twin and self-attention based deep learning models. *IEEE Access*, 11:114013–114030, 2023. .

[12] B. Sharma, P. Pokharel, and B. Joshi. User behavior analytics for anomaly detection using lstm autoencoder - insider threat detection. In *Proceedings of the 11th International Conference on Advances in Information Technology (IAIT2020)*, pages 1–9, 2020. .

[13] J. Lu and R. K. Wong. Insider threat detection with long short-term memory. In *Proceedings of the Australasian Computer Science Week Multiconference*, pages 1–10, 2019. .

[14] Xingjian Zhu, Jiankuo Dong, Jin Qi, Zhenguo Zhou, Zhenjiang Dong, Yanfei Sun, and Moyu Wang. Auth: An adversarial autoencoder based unsupervised insider threat detection scheme for multisource logs. *IEEE Transactions on Industrial Informatics*, 20(9):10955–10965, 2024. .

[15] M. N. Al-Mhiqani et al. A new intelligent multilayer framework for insider threat detection. *Computers & Electrical Engineering*, 97:107597, 2022. .

[16] Kossi Bissadu, Gahangir Hossain, Leela Pavani Velagala, and Salleh Sonko. Analyzing insider cyber threats and human factors within the framework of agriculture 5.0. In *2024 12th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–5. IEEE, 2024.

[17] F. Yuan et al. Insider threat detection with deep neural network. In *Proceedings of the 18th International Conference on Computational Science (ICCS 2018)*, pages 43–54. Springer, 2018. .

[18] B. B. Sarhan and N. Altwaijry. Insider threat detection using machine learning approach. *Applied Sciences*, 13(1):259, 2022. .

[19] Jianguo Jiang et al. Anomaly detection with graph convolutional networks for insider threat and fraud detection. In *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*, 2019. .

[20] A. Anju et al. Detection of insider threats using deep learning. In *2023 3rd International Conference on Pervasive Computing and Social Networking (ICPCSN)*, 2023. .

[21] Menghua Yang, Junkai Yi, Hejun Zhu, and Lingling Tan. Cnn-lstm-based insider threat detection model. In *2025 International Conference on Electrical Automation and Artificial Intelligence (ICEAAI)*, pages 985–990. IEEE, 2025.

[22] Francisco Montes, J Bermejo, Luís Enrique Sanchez, JR Bermejo, and Juan Antonio Sicilia. Detecting malware in cyberphysical systems using machine learning: A survey. *KSII Transactions on Internet and Information Systems (TIIS)*, 15(3):1119–1139, 2021.

[23] Hossein Shadabfar, Motahareh Dehghan, and Babak Sadeghian. Dsrl-apt-2023: A new synthetic dataset for advanced persistent threats. *ISeCure*, 17(2), 2025.

[24] Motahareh Dehghan, Babak Sadeghian, Erfan Khosravian, Alireza Sedighi Moghaddam, and Farshid Nooshi. Proapt: Projection of apts with deep reinforcement learning. *ISeCure*, 17(1), 2025.

[25] Xiaoling Tao, Jianxiang Liu, Yuelin Yu, Haijing Zhang, and Ying Huang. An insider threat detection method based on improved test-time training model. *High-Confidence Computing*, 5(1): 100283, 2025.

[26] Ali Haidar, Yu-Zheng Lin, Qinxuan Shi, Zhanglong Yang, Desmond Amadigwe, Brian Terry, Sudheer Krishna Battu, Pratik Satam, and Sicong Shao. A survey of large language models for insider threat detection. In *2025 Cyber Awareness and Research Symposium (CARS)*, pages 1–10. IEEE, 2025.

[27] Mohammad AL-Smadi. Multi-axis trust modeling for interpretable account hijacking detection. *arXiv preprint arXiv:2603.13246*, 2026.

[28] Duc C. Le and Nur Zincir-Heywood. Anomaly detection for insider threats using unsupervised ensembles. *IEEE Transactions on Network and Service Management*, 18(2):1152–1164, 2021. .

[29] RG Gayathri, Atul Sajjanhar, and Yong Xiang. Image-based feature representation for insider threat classification. *Applied Sciences*, 10(14): 4945, 2020.

[30] Qingwei Chen, Xin Xing, Shumei Li, Ying Shao, Xiangjun Song, and Zhao Qi. Dynml-net: A porcine enteric virus identification network based on protein language models and a dynamic heterogeneous multi-branch architecture. 2026.

[31] G. A. Pradipta, R. Wardoyo, A. Musdholifah, I. N. H. Sanjaya, and M. Ismail. Smote for handling imbalanced data problem : A review. In *Proceedings of the 2021 Sixth International Conference on Informatics and Computing (ICIC)*, pages 1–8. IEEE, 2021. .

[32] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra. The design and operation of CloudLab. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, pages 1–14, July 2019. URL https://www.flux.utah.edu/paper/duplyakin-atc19.

[33] Alok Sharma, Edwin Vans, Daichi Shigemizu, Keith A Boroevich, and Tatsuhiko Tsunoda. Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture. *Scientific reports*, 9(1): 11399, 2019.

[34] Yitan Zhu, Thomas Brettin, Fangfang Xia, Alexander Partin, Maulik Shukla, Hyunseung Yoo, Yvonne A Evrard, James H Doroshow, and Rick L Stevens. Converting tabular data into images for deep learning with convolutional neural networks. *Scientific reports*, 11(1):11325, 2021.

Majduddeen Almoayed received the B.Sc. and M.Sc. degrees in computer engineering from the University of Mazandaran, Babolsar, Iran, in 2021 and 2024, respectively. His research interests include computer security, and deep learning.

Payam Mahmoudi-Nasr received the B.Sc. and M.Sc. degrees in computer engineering from the Amirkabir University of Technology, Tehran, Iran, in 1994 and 1996, respectively, and the Ph.D. degree in electrical engineering from the Tarbiat Modares University, Tehran, Iran, in 2016. Since 2008, he has been with the Computer Engineering Department, University of Mazandaran, Babolsar, Iran. His research interests include industrial network security, information security, and smart grids.

Mohammad Mosafer received the M.S. degree in computer engineering from the University of Mazandaran (UMZ), Babolsar, Mazandaran, Iran in 2025. His research interests include DL/ML, Cyber Security, HPC and Cloud computing.