Ensemble-Based Deep Learning for Unsupervised Insider Threat Detection

Abstract

Insider threats from authorized users present a significant cybersecurity challenge in interconnected systems. This research develops an online, unsupervised deep learning system employing a DNN and LSTM ensemble to model normal user behavior from system logs. The system generates interpretable anomaly scores based on deviations, achieving AUCs of 0.95 (DNN) and 0.93 (LSTM) on the CERT dataset, outperforming baseline methods. The novelty lies in its adaptable ensemble framework with feature-level anomaly decomposition, not requiring labeled data. The research methodology encompasses feature extraction, structured stream neural networks, probability-based anomaly scoring, online training, anomaly detection, and comparison with baselines. Result analysis highlights the superior performance of deep learning, the detrimental impact of categorical features, and the benefit of diagonal covariance. The study concludes the effectiveness of the proposed system for real-time, adaptable, and interpretable insider threat detection. Future work will focus on enhancing categorical feature handling and testing on diverse datasets, with key lessons emphasizing ensemble models and interpretability.

1 Introduction

The proliferation of digital infrastructure and data-driven operations has amplified insider threat risks, as authorized individuals may misuse their access for malicious or negligent purposes, ranging from intellectual property theft to accidental security setting misconfigurations. The dynamic nature of modern enterprise networks—incorporating cloud services, remote work, and varied user roles—complicates distinguishing legitimate activities from potential threats, while traditional cybersecurity measures, focused on external attacks, often fail to monitor trusted users within the system. This underscores the need for advanced detection mechanisms leveraging deep learning to model complex behaviors and detect anomalies in real time, thus reducing analyst burden and increasing efficiency, especially when resources are limited.

Recent studies, such as [Homoliak et al., 2019] and [CSO and KnowBe4, 2018], highlight the prevalence and impact of insider threats, with 53% of organizations and 42% of U.S. federal agencies reporting incidents annually, resulting in significant financial, operational, and reputational damage. As mentioned by [Le and Zincir-Heywood, 2021], detecting insider threats is challenging as they know security protocols and often try to blend their actions with normal operations. Key obstacles include the authorized nature of access bypassing traditional perimeter defenses; imbalanced datasets with rare anomalies, and evolving user behaviors requiring adaptive mechanisms that update continuously.

Although supervised learning approaches have been explored, they heavily rely on labeled datasets that are often scarce or incomplete, while existing unsupervised methods like Isolation Forest, SVM, and PCA struggle with complex, non-linear streaming data and lack interpretability. Moreover, very few studies address real-time, online detection that adapts to dynamic behaviors.

The primary objective of this research is to develop an online, unsupervised deep learning system for detecting anomalous network activity using system logs and generating interpretable anomaly scores that break down contributions from individual features to assist analysts. The system was evaluated using the CERT Insider Threat Dataset v6.2 [Glasser and Lindauer, 2013], which simulates realistic user activities and insider threat scenarios.

The proposed system outperforms baseline methods, achieving higher precision and recall in identifying anomalies. By dynamically modeling user behavior, it captures deviations in streaming data despite imbalanced datasets, while feature-level anomaly decomposition helps analysts trace threats to specific features, enabling faster, more informed responses. It eliminates the need for labeled data, making it practical for real-world use. Additionally, the system adapts to evolving patterns, ensuring robustness against changing user behaviors.

The remainder of this report is organized as follows: Section 2 reviews related work, Section 3 details the methodology, including the proposed deep learning architecture and experimental setup, Section 4 presents the results and comparative analysis with baseline methods, and finally, Section 5 discusses the implications, limitations, and future directions.

2 Background and Related Work

Existing research on unsupervised anomaly detection for insider threats has shown promise but also revealed critical limitations. While unsupervised approaches excel at identifying statistically rare behaviors [Gavai et al., 2015], they often suffer from high false positive rates due to temporary workload spikes or legitimate activities. This highlights the lack of contextual prioritization, like integrating role-based access or psychological stressors, which limits actionable insights for security teams.

The reliance on handcrafted temporal feature engineering, as in [Le and Zincir-Heywood, 2021], further limits generalization, since static representations (e.g., concatenation, percentile differences, mean/median shifts) yield inconsistent results across threat scenarios, with some features underperforming. This suggests that static, manual feature engineering may not adapt to evolving threats, indicating a need for dynamic or learned temporal representations.

A common approach frames insider threat detection as an anomaly detection task. [Chandola et al., 2012] provide a comprehensive survey of anomaly detection techniques and highlight the underdevelopment of methods suitable for online and multivariate data sequences. The key findings are– effective systems must operate in real time, track multiple users concurrently, analyze structured multivariate data, adapt to shifting distributions, and provide interpretable outputs. While existing methods address some of these aspects, the proposed approach aims to address all collectively.

[Carter and Streilein, 2012] extends traditional methods like EWMA into probabilistic models for streaming data, adapting to changing distributions but relying on parametric assumptions, whereas the deep learning architecture in our work models complex distributions with fewer assumptions.

Neural network-based approaches have also gained traction in cybersecurity research. [Ryan et al., 1998] used a feedforward network to estimate user command probabilities but relied on unstructured inputs and offline training. [Debar et al., 1992] applied RNNs to detect anomalies via prediction failures, incorporating limited online learning but lacking continuous adaptation to evolving behavior.

Ensemble-based systems, such as [Veeramachaneni and Arnaldo, 2016], combine auto-encoders, PCA, and probabilistic models to detect anomalies in network logs over time windows. Their system operates online and improves via analyst feedback, though it does not explicitly track individual users' behavioral changes over time and fails to provide interpretable insights.

This paper aims to bridge these gaps by introducing a dynamic, hybrid framework that combines deep learning-based temporal modeling with contextual and role-aware anomaly scoring, offering a more robust and practical solution for insider threat detection in enterprise environments.

3 Methods

This section outlines the methodology developed for ensemble-based deep learning to enable unsupervised insider threat detection (see Fig. 1). The core technical ideas, research techniques, and rationale for key decisions are organized into feature extraction, model architecture, prediction mechanisms, anomaly detection, online training, and baseline comparisons.

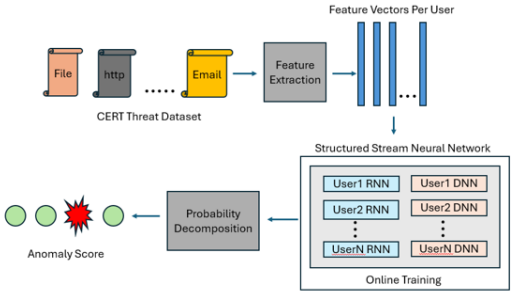

Figure 1: Methods Overview

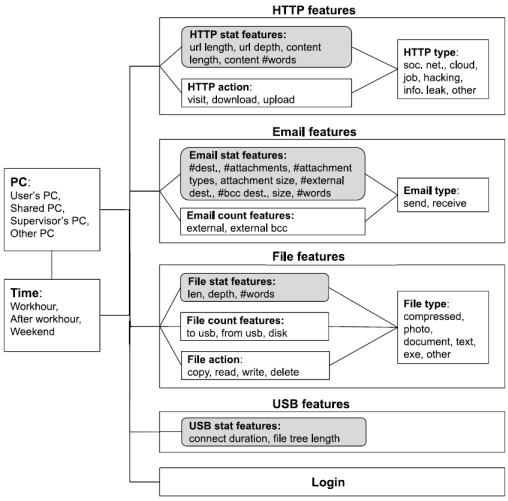

Figure 2: Feature Extraction

3.1 Feature Extraction

To prepare raw system logs for machine learning, user activity data is transformed into fixed-length numerical vectors. Actions are aggregated by user ID over fixed time windows, producing structured behavior representations. Two main feature types are extracted:

- **Count Features**: Quantitative counts of actions within each aggregation window (e.g., emails sent, files accessed, downloads after hours).
- **Categorical Features**: Metadata such as user role or department, encoded as discrete variables.

To capture richer context, upto 3 related features are combined (see Fig. 2)—such as activity on shared machines, average attachment size, or behavior after work hours—where risks might not be apparent from individual signals, but combined, they paint a high-risk scenario. This process yields a 408-dimensional count vector and six categorical variables, ensuring a comprehensive and temporally consistent representation of user behavior for insider threat detection.

3.2 Structured Stream Neural Network

The core of the proposed system is a structured stream neural network designed to predict next-day feature vectors, enabling anomaly detection through deviations from expected behavior. Two model variants are employed: Deep Neural Network (DNN) and Recurrent Neural Network (RNN)-LSTM. The DNN processes inputs step-by-step, while the LSTM uses Backpropagation Through Time (BPTT) over a fixed window to update weights, balancing personalization (per-user states) and efficiency (shared parameters). The ensemble approach combines the DNN's strength in capturing instantaneous anomalies with the LSTM's ability to model temporal trends, addressing the challenge of evolving user behaviors.

3.2.1 Deep Neural Network (DNN)

DNN with L hidden layers processes daily feature vectors for a user $(x_1^u, x_2^u, ..., x_t^u)$ independently, transforming an input sequence step by step. At each layer, the transformation follows:

$$h_{l,t}^u = g(W_l h_{l-1,t}^u + b_l) \tag{1}$$

where, g is the activation function, W is the weight matrix, and b is the bias term.

3.2.2 Recurrent Neural Network (RNN) with LSTM

In an RNN with L hidden layers (l = 1,. . . ,L), the output of hidden layer L is not a function of feature vector x_t^u alone; rather, it is a function of a long-term memory cell c_t^u as follows:

$$h_{l,t}^u = o_{l,t}^u \odot tanh(c_{l,t}^u) \tag{2}$$

$$c_{l,t}^u = f_{l,t}^u \odot c_{l,t-1}^u + i_{l,t}^u \odot g_{l,t}^u \tag{3}$$

$$g_{l,t}^u = tanh(W_l^{(g,x)} h_{l-1,t}^u + W_l^{(g,h)} h_{l-1,t}^u + b_l^g) \tag{4}$$

$$f_{l,t}^u = \sigma(W_l^{(f,x)} h_{l-1,t}^u + W_l^{(f,h)} h_{l-1,t}^u + b_l^f) \tag{5}$$

$$i_{l,t}^u = \sigma(W_l^{(i,x)} h_{l-1,t}^u + W_l^{(i,h)} h_{l-1,t}^u + b_l^i) \tag{6}$$

$$o_{l,t}^u = \sigma(W_l^{(o,x)} h_{l-1,t}^u + W_l^{(o,h)} h_{l-1,t}^u + b_l^o \tag{7}$$

Here the input to the first layer is the feature vector x_t^u. For each layer $1 \leq l \leq L$, the hidden state $h_{l,t}^u$ and cell state $c_{l,t}^u$ are initialized as zero vectors. The candidate cell update is denoted as $g_{l,t}^u$. The forget gate $f_{l,t}^u$ controls memory retention, the input gate $i_{l,t}^u$ governs updates to the cell state, and the output gate $o_{l,t}^u$ determines the hidden state derived from the updated cell state.

3.3 Probability Decomposition

Given the hidden state at time t–1, (h_{t-1}^u), our model outputs the parameters θ for a probability distribution over the next observation x_t^u. The anomaly score for the user u at time t, a_t^u, is then:

$$a_t^u = -\log P_\theta(x_t^u | h_{t-1}^u) \tag{8}$$

This probability is complicated by the fact that our feature vectors include six categorical variables in addition to the 408-dimensional count vector. Thus, $P_\theta(x_t^u | h_{t-1}^u)$ is the joint probability over The count vector $\hat{x}_t^u$ and Categorical variables: role (R), project (P), functional unit (F), department (D), team (T), and supervisor (S). This assumes features as independent once we know the past behavior.

Anomaly Score Calculation: A lower probability indicates more suspicious or anomalous user activity, suggesting a potential deviation from normal behavior. This is quantified using an anomaly score, denoted as a_t^u, which reflects the degree of abnormality in the behavior of the user u at time t.

3.3.1 Joint Probability Decomposition

The joint probability distribution is calculated as:

$$P_\theta(x_t^u | h_{t-1}^u) \approx P_{\theta^{(\hat{x})}}(\hat{x}_t^u | h_{t-1}^u) \prod_{V \in C} P_{\theta^{(V)}}(V_t^u | h_{t-1}^u) \tag{9}$$

The hidden state capturing behavioral history is denoted as h_{t-1}^u, and θ represents the parameters of a joint probability distribution, which is decomposed into Continuous and Categorical features.

- **Continuous Features**: Modeled as a multivariate normal distribution with either identity covariance (uniform variance) or diagonal covariance (feature-specific variance), expressed as: $P_{\theta^{(\hat{x})}}(\hat{x}_t^u | h_{t-1}^u)$
- **Categorical Features**: Modeled using a softmax function to produce probabilities over discrete categories (e.g., user roles), expressed as: $\prod_{V \in C} P_{\theta^{(V)}}(V_t^u | h_{t-1}^u)$

3.3.2 Conditional Independence

Computing the joint probability of all variables is complex. Assuming conditional independence given the hidden state, the joint probability is approximated as the product of individual feature probabilities. We predict seven parameter vectors: $\theta^{(\hat{x})}$ and $\theta^{(V)}$ for $V \in C$. Each parameter is generated using a small neural network.

$$\theta^{(\hat{x})} = U'_{\hat{x}} tanh(U_{\hat{x}} h_{t-1} + b_{\hat{x}}) + b'_{\hat{x}} \tag{10}$$

$$\theta^{(V)} = f(U'_V tanh(U_V h_{t-1} + b_V) + b'_V) \tag{11}$$

Here the function f denotes the softmax function, which converts a vector of numbers into a probability distribution. The terms $U_{\hat{x}}$ and U_V represent two weight matrices, and the parameters $\theta^{(\hat{x})}$ and $\theta^{(V)}$ are shared across all users, similar to LSTM weights.

3.4 Prediction Modes

Two prediction modes have been used for this project:

- **Next Time Step** forecasts the feature vector for time t using past data up to $t-1$, relying on the hidden state h^u_{t-1} to model temporal dependencies. Anomalies are flagged when actual behavior deviates from the prediction.
- **Same Time Step** reconstructs the feature vector at time t using all information up to t, resembling an **autoencoder**. It uses h^u_{t-1} to reproduce the input, and discrepancies indicate anomalies.

In experiments, the same time step mode slightly outperformed next time step prediction, better capturing real-time deviations (shown in Result). Its autoencoder-like behavior also improved robustness to noise and evolving patterns, making it the preferred choice for further evaluation.

3.5 Anomaly Detection

Anomaly detection identifies user activities that deviate from normal behavior. For each user-day, the system computes an anomaly score a^u_t using probability decomposition. User-days are ranked by these scores to help analysts prioritize reviews, especially when resources are limited. Scores are standardized using exponentially weighted moving average (EWMA) to reduce temporal bias. The score's decomposition highlights which features contribute most to the anomaly, and top-ranked user-days are flagged (TRUE/FALSE) for further investigation.

3.6 Online Training

Fixed-size retraining can cause the loss of important past events and make real-time prediction infeasible. To address this, the system uses online training, which incrementally updates model weights as new data arrives, enabling continuous adaptation to evolving behaviors.

In DNN, weights are updated through standard backpropagation after each sample, while LSTM uses Back Propagation Through Time (BPTT). DNN processes samples sequentially, updating weights immediately after each sample to minimize memory usage. LSTM, on the other hand, processes multiple users over fixed windows of past steps, maintaining per-user states and sharing weights globally. This approach ensures real-time responsiveness without losing historical patterns.

3.7 Comparison with Baseline Models

Baselines provide a robust reference to validate the proposed system's superior performance. The proposed system is benchmarked against three commonly used methods in insider threat detection: **One-Class SVM**, **Isolation Forest**, and **PCA**.

Hyperparameters tuned were: neural networks (*1–6 hidden layers, 16–1024 hidden layer dimensions, tanh activation*), PCA (*0–30 components*), Isolation Forest (*50–200 estimators, 0.01–0.1 contamination rate, max_features fixed at 1.0*), and SVM (*rbf/linear/poly/sigmoid kernels, 0–1 gamma, 1–10 degree of polynomial kernel*). For other DNN and RNN hyperparameters, refer to Table 1.

Performance was measured using **Cumulative Recall (CR-k)**, which sums recalls up to ***k*** daily reviews. Since insider threat detection prioritizes catching true positives over minimizing false positives, CR-k was preferred over precision-based metrics. It reflects the practical goal of maximizing true positives within a fixed analyst review budget.

4 Results

4.1 Anomaly Score

The anomaly detection model shows strong potential by assigning high scores to unusual behaviors. Table 2 lists the top 10 anomalies, with several confirmed as true positives. Some high-scoring cases were not labeled as anomalies, reflecting the model's sensitivity to suspicious patterns, especially valuable in real-world settings, where early identification is critical.

Table 1: DNN and RNN Hyperparameters

Hyperparameter	DNN	RNN
Batch size	256 samples	between 256 and more samples Larger batch sizes speed up model training, which is more important for the RNN than the DNN
Learning rate	0.01	0.01 Fixed
Time steps for BPTT	NA	between 3 and 40

Further reinforcing the model's effectiveness, Table 3 presents 10 additional anomalies with relatively lower scores, but all are true positives. These results reveal the model's ability to detect a broad range of threat behaviors, from highly abnormal activity to more subtle, stealthy deviations.

Table 2: Top 10 Anomalies (Max False Positive)

Rank	Index	Score	True Label
1	7951	154531.80	FALSE
2	1340	151252.50	TRUE
3	7527	69730.94	FALSE
4	1026	53474.16	FALSE
5	560	51272.34	FALSE
6	4222	42519.88	FALSE
7	9476	40128.82	FALSE
8	4413	39983.91	TRUE
9	6986	32041.28	TRUE
10	2151	29489.71	FALSE

Table 3: Next 10 Anomalies (Max True Positive)

Index	Score	True Label
3874	2443.97	TRUE
9644	1981.11	TRUE
4556	-3370.26	TRUE
918	-4000.71	TRUE
470	-4362.60	TRUE
6862	-4510.86	TRUE
5590	-4522.02	TRUE
4743	-4534.91	TRUE
3853	-4614.79	TRUE
9504	-4632.94	TRUE

4.2 Baseline Comparison

Table 4 compares model performance using AUC, Recall at Top-N (R-N), and Cumulative Recall at Top-1000 (CR-1000). Among baselines, **Isolation Forest** outperforms PCA and One-Class SVM with an **AUC of 0.89** and **CR-1000 of 230**. Deep learning models outperform all baselines, with the **DNN** achieving an **AUC of 0.95** and **CR-1000 of 320**, consistently retrieving the most true anomalies.

Table 4: Neural Model vs Baseline Model

Model	AUC	R-25	R-50	R-100	R-200	R-400	R-600	R-800	CR-1000
DNN	0.95	16	24	32	38.4	48	51.2	54.4	320
LSTM	0.93	15.5	23.25	31	37.2	46.5	49.6	52.7	310
Isolation Forest	0.89	11.5	17.25	23	27.6	34.5	36.8	39.1	230
PCA	0.75	8.5	12.75	17	20.4	25.5	27.2	28.9	170
One-Class SVM	0.7	7.5	11.25	15	18	22.5	24	25.5	150

While LSTM captures temporal patterns well, it shows no significant advantage over DNN on the CERT Insider Threat dataset, likely due to limited long-range dependencies. However, LSTM may perform better in real-world scenarios with richer temporal dynamics. The ROC curves in Fig. 3 show that the DNN curve consistently leads, showing better trade-offs between true and false positives. Overall, the results highlight the robustness of deep learning models for detecting insider threats.

4.3 Categorical Variables vs. Model Complexity

The comparison between models using count features alone versus all features (count + categorical) reveals that adding categorical data consistently reduced performance. For example, DNN's AUC dropped from 0.9229 (count only) to 0.8475, and LSTM's from 0.9251 to 0.8241 (see Table 5). This decline suggests that categorical features introduced noise without improving detection. Across recall metrics, count-only models matched or outperformed their counterparts.

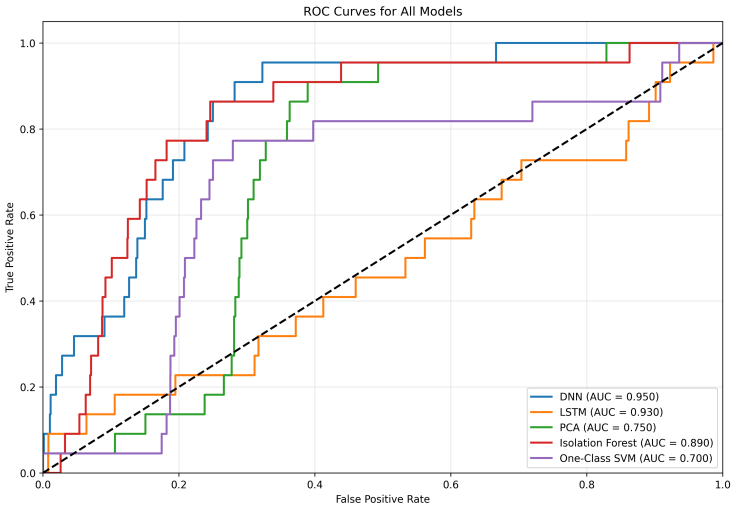

Figure 3: Experimental Result Comparision

Table 5: Categorical Variables vs. Model Complexity

Model	Features	Training Time (s)	AUC	R-25	R-50	R-100	R-200	R-400	CR-1000
DNN	count	10.74	0.92	0.00	4.55	13.64	22.73	27.27	236.36
DNN	all	12.94	0.85	9.09	9.09	9.09	22.73	27.27	177.27
LSTM	count	45.55	0.93	0.00	4.55	13.64	22.73	31.82	250
LSTM	all	55.24	0.82	9.09	9.09	13.64	18.18	27.27	177.27

As a result, later experiments used only count features to improve efficiency and accuracy, highlighting the value of careful feature selection in both sequential and non-sequential models.

4.4 Same vs. Next Time Prediction

Table 6: Same vs Next Time Prediction

Model	Prediction Model	AUC	R-100	CR-100	R-200	CR-200	R-400	CR-400	CR-1000
DNN	same	0.85	9.09	27.27	22.73	50.00	27.27	77.27	177.27
DNN	next	0.83	13.64	31.82	18.18	50.00	27.27	77.27	177.27
LSTM	same	0.82	13.64	31.82	18.18	50.00	27.27	77.27	177.27
LSTM	next	0.82	13.64	31.82	18.18	50.00	27.27	77.27	172.73

Table 6 shows performance differences between same-time-step and next-time-step predictions for DNN and LSTM models. DNNs performed better with same-time-step prediction (AUC 0.8468 vs. 0.8251), showing higher early recall and stronger results overall. LSTMs showed minimal AUC difference (0.8234 vs. 0.8247), with same-time-step slightly outperforming in long-range recall (CR-1000 = 177.27 vs. 172.73). These results suggest DNNs benefit from immediate feedback, while LSTMs perform well in both modes, with a slight edge for same-time-step in stability.

4.5 Diagonal vs. Identity Covariance

Table 7 highlights a clear performance gap between diagonal and identity covariance implementations. DNN with diagonal covariance achieved an AUC of 0.8381, a 22.6% improvement over DNN-identity (0.6837), with higher recall scores (R-100 = 13.64 vs. 4.55). LSTM-diag also showed a 19.2% AUC improvement (0.8205 vs. 0.6883) and better CR-1000 (172.73 vs. 36.36).

These findings highlight the importance of modeling feature-specific variances. Diagonal covariance outperformed identity covariance, which assumes uniform variance and zero correlation. The DNN

Table 7: Diagonal vs Identity Covariance

Model	Covariance Type	AUC	R-25	R-50	R-100	R-200	R-400	CR-1000
DNN	diag	0.84	9.09	9.09	13.64	22.73	27.27	181.82
DNN	identity	0.68	4.55	4.55	4.55	4.55	4.55	36.36
LSTM	diag	0.82	9.09	9.09	13.64	18.18	27.27	172.73
LSTM	identity	0.69	4.55	4.55	4.55	4.55	4.55	36.36

with diagonal covariance slightly outperformed the LSTM in AUC and CR-1000, indicating its feedforward structure better leverages these relationships. The performance drop with identity covariance underscores the need for proper variance modeling in insider threat detection.

4.6 Online Training Result

Table 8 compares the online training of DNN and LSTM models. Both achieved the same AUC (0.819) but differed in learning dynamics. DNN showed greater loss reduction (3.81×10^8), while LSTM started with a lower loss and improved by 1.15×10^8, suggesting better initialization or alignment with the task. LSTM slightly outperformed DNN in CR-1000 (150 vs. 145.45), indicating better early ranking. Overall, LSTM offers faster convergence for quick deployment, while DNN may benefit more from extended training.

Table 8: Online Training - Loss Improvement

Model	Initial Loss	Final Loss	Loss Improvement	AUC	CR-1000
DNN	8.25×10^8	4.44×10^8	3.81×10^8	0.819	145.45
LSTM	4.44×10^8	3.29×10^8	1.15×10^8	0.819	150.00

5 Conclusion and Future Work

This research aimed to develop an online, unsupervised deep learning system for detecting insider threats by modeling user behavior from system logs without labeled data. The core objectives were real-time anomaly detection, adaptability to evolving user patterns, interpretable anomaly scores, and improved performance over traditional methods. The hypothesis was that an integrated ensemble framework could overcome individual model limitations and provide robust, explainable insights.

These goals were achieved through an ensemble system combining a DNN and an LSTM network. Evaluation on the CERT Insider Threat Dataset v6.2 showed that the system effectively detected anomalies in real time, adapted to behavioral drift through online learning, and produced interpretable outputs for analyst decision-making, supporting the research hypothesis.

Key lessons learned include the significant impact of carefully designed ensemble models, the necessity of interpretability in security applications, the importance of thoughtful data representation, and the need for online learning. This research provides a practical, adaptive framework for insider threat detection and a foundation for future advancements in cybersecurity, with a focus on enhancing categorical feature handling, testing on more diverse datasets, and incorporating long-term behavioral patterns and richer contextual signals to improve detection accuracy and robustness against stealthy threats.

References

Kevin M. Carter and William W. Streilein. Probabilistic reasoning for streaming anomaly detection. In *Proceedings of the IEEE Statistical Signal Processing Workshop (SSP)*, pages 377–380, 2012.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection for discrete sequences: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):823–839, 2012.

U.S. Secret Service CSO, CERT Division of SEI-CMU and KnowBe4. The 2018 u.s. state of cybercrime survey. Technical report, International Data Group, Boston, MA, USA, 2018. Accessed: Apr. 1, 2021.

Hervé Debar, Marc Becker, and Didier Siboni. A neural network component for an intrusion detection system. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 240–250, 1992.

Gautam Gavai, Kumar Sricharan, David Gunning, Richard Rolleston, John Hanley, and Manish Singhal. Detecting insider threat from enterprise social and online activity data. In *Proceedings of the 7th ACM CCS International Workshop on Managing Insider Security Threats*, pages 13–20, October 2015.

Joshua Glasser and Brian Lindauer. Bridging the gap: A pragmatic approach to generating insider threat data. In *2013 IEEE Security and Privacy Workshops*, pages 98–104, 2013. doi: 10.1109/SPW.2013.37.

I. Homoliak, F. Toffalini, J. Guarnizo, Y. Elovici, and M. Ochoa. Insight into insiders and it: A survey of insider threat taxonomies, analysis, modeling, and countermeasures. *ACM Computing Surveys*, 52(2):30, April 2019. doi: 10.1145/3303779.

Duy Cuong 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.

J. Ryan, M.-J. Lin, and Risto Miikkulainen. Intrusion detection with neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 943–949, 1998.

Kalyan Veeramachaneni and Iverson Arnaldo. Ai2: Training a big data machine to defend. In *Proceedings of the IEEE Symposium on High Performance Signal Processing and Intelligent Data Security (HPSC and IDS)*, 2016.