AIMS Mathematics, 10(10): 23496–23517.
DOI: 10.3934/math.20251043
Received: 29 March 2025
Revised: 26 September 2025
Accepted: 30 September 2025
Published: 16 October 2025

http://www.aimspress.com/journal/Math

Research article

Cybersecurity threat detection based on a UEBA framework using Deep Autoencoders

Jose Fuentes[1], Ines Ortega-Fernandez[1,*], Nora M. Villanueva[2,3] and Marta Sestelo[2,3]

[1] Galician Research and Development Center in Advanced Telecommunications (Gradiant), 36214 Vigo, Spain
[2] Galician Center for Mathematical Research and Technology (CITMAga), 15782 Santiago de Compostela, Spain
[3] Universidade de Vigo, Department of Statistics and O.R. & SiDOR Group, 36310 Vigo, Spain

* **Correspondence:** Email: iortega@gradiant.org, Tel: +34986120430.

Abstract: The increasing sophistication of cyberattacks, especially insider and process-related anomalies, poses a major challenge to enterprises, as traditional rule-based or shallow anomaly detection systems often fail to capture complex behavioral patterns. User and Entity Behavior Analytics (UEBA) is a broad branch of data analytics that attempts to build a normal behavioral profile in order to detect anomalous events. Among the techniques used to detect anomalies, deep autoencoders constituted one of the most promising deep learning models on UEBA tasks, allowing explainable detection of security incidents that could lead to the leak of personal data, hijacking of systems, or access to sensitive business information. In this study, we introduced the first implementation of an explainable UEBA-based anomaly detection framework that leveraged deep autoencoders in combination with Doc2Vec, a neural network-based approach that learns the distributed representation of documents, to process both numerical and textual features. Additionally, based on the theoretical foundations of neural networks, we offered a novel proof demonstrating the equivalence of two widely used definitions for fully-connected neural networks. The experimental results demonstrated the proposed framework's capability to detect real and synthetic anomalies effectively generated from real attack data, showing that the models provided not only correct identification of anomalies but also explainable results that enabled the reconstruction of the possible origin of the anomaly. Compared to existing UEBA and anomaly detection approaches, the novelty of our framework lied in combining explainable multimodal feature processing with formal mathematical guarantees. Our findings suggested that the proposed UEBA framework can be seamlessly integrated into enterprise environments.

Keywords: anomaly detection; user and entity behavior analytics; autoencoders; deep learning; cybersecurity; cyber threat detection

Mathematics Subject Classification: 68M25, 68T07

1. Introduction

In the current digital era, cybersecurity and the reliability of both physical and logical systems have become of increasing importance for industry and academia alike. The exponential growth of interconnected devices, the increasing volume of sensitive data, and the complexity of technological infrastructures highlight the need for robust algorithms that improve security and resilience. Mathematical models and methods play a central role in this task by offering formal frameworks to identify cyberattacks, develop cryptographic protocols, simulate potential incidents under a wide range of scenarios, and design defense strategies against adversarial threats. As these challenges intensify, mathematical approaches are becoming more crucial to guarantee robust system performance and to mitigate future cyber risks. Learning-based methods are increasingly employed to secure diverse environments. For example, in internet of things (IoT) systems, they are applied to detect attacks using heterogeneous sensor and network data [1]. In cyberphysical infrastructures, they have been explored to monitor and defend interconnected systems such as the energy-water nexus [2]. A related challenge arises in enterprise settings, where detecting anomalies in user activity —such as signs of cyberattacks, rogue insiders, or negligent behavior— has become an essential task for modern security operations centers.

User and Entity Behavior Analytics (UEBA) [3] is a powerful methodology to identify cyber threats by creating models of normal behavioral patterns and detecting deviations that may indicate malicious or negligent activities. To profile the behavior of an entity, these models allow the incorporation of multiple sources of information such as sensor readings, network traffic, system logs, security alerts, email information, and even geo-positioned or biometric data. UEBA uses advanced statistical learning techniques to model the behavior of users, employees, and customers, as well as machines, such as servers, switches, and personal systems. By analyzing anomalies in the behavior of users and devices, UEBA can detect intrusions, impersonation attacks, or negligent users [4].

In parallel, explainable artificial intelligence (xAI) has emerged as a key enabler in cybersecurity scenarios by addressing the growing need for interpretability and trust in complex AI-driven security systems. Since modern cybersecurity solutions rely on deep learning models, its inherent lack of transparency can prevent the analysts' ability to understand and respond to detected threats effectively. The integration of xAI techniques such as SHapley Additive exPlanations (SHAP), Local Interpretable Model-agnostic Explanations (LIME), or latent-space analysis into cybersecurity operations enable cybersecurity experts to interpret and understand why a system has identified a specific event as anomalous behavior, facilitating root-cause analysis and informed decision-making, prioritizing genuine threats, and helping to identify false positives. Therefore, by integrating xAI techniques, UEBA-based cybersecurity tools not only improve technical performance, but also improve compliance with regulatory frameworks (such as the AI Act) that demand transparency in AI-based decision making, especially in sensitive areas such as finance.

Several methods have been proposed in the literature related to the use of these techniques, which we summarize in Table 1. In [3], UEBA models based on Mahalanobis distance and singular value

decomposition (SVD) are implemented to identify anomalous behavior in users accessing a server. Voris et al. [5] uses Gaussian mixture models (GMM) for each computer, collecting data for the file system, process launch, and network behavior, in addition to establishing a series of trap files to attract and identify attackers. They also apply UEBA to continuously identify the user in the system by monitoring their activity. Another example is found in [6], where decision trees are applied to mouse movement data to identify the user. Similarly, in [7], results of applying UEBA with mouse movement data, keyboard typing dynamics, and event sequences in the context of online banking operations are compared. In [8], a similar approach based on the radial basis function network (RBFN) classifier with particle swarm optimization (PSO) is applied to verify the user identity through touchscreen usage and other biometric data collected during web browsing. Moreover, the combination of static (logins, cookies, system type, etc.) and dynamic (mouse, keyboard, microphone, network usage, etc.) data to build user models was explored in [9], where UEBA models are also used to guarantee user coherence when performing authentication with identity federations.

UEBA can be considered a use case for anomaly detection (or outlier detection) with personalized models. Once the users are identified, each model has to detect data points that do not conform to the expected behavior. As anomaly detection has grown in popularity, a wide array of methods and techniques has emerged. Among these, autoencoders [10] are a type of artificial neural network used for this purpose. Their goal is to learn (in an unsupervised way) a representation of the dataset by filtering out insignificant data or noise. Recent work in xAI for cybersecurity highlights the necessity of designing inherently interpretable (ante-hoc) models, prioritizing explainability principles from model conception through training [11]. A key advantage of the use of autoencoders for cybersecurity is that they tend to be more explainable than other deep learning models [12, 13]. Moreover, in cases where the well-defined anomaly distribution (WDAD) assumption does not hold [14, 15], autoencoders can be trained on data assumed to be normal (even if slightly contaminated) [16].

Autoencoders (AEs) have been used for anomaly detection since the work of Hawkins et al. [17], and a general overview can be found in [18]. Examples include using variational AEs (VAEs) to construct anomaly scores [19] and convolutional AEs for video signal anomaly detection [20]. In [21], both incomplete and overcomplete AEs are applied in satellite data, while [22] proposes a VAE coupled with a transformer architecture to account for dependencies in satellite data. In industrial anomaly detection, [23] uses a norm-regularized AE, and [12] combines an AE with an long short-term memory (LSTM) network. Also, [13] utilizes VAEs to classify anomalies in engineering systems. Finally, [24] showed that AEs outperform other algorithms in the detection of denial of service cyberattacks in industrial scenarios.

The motivation of this work lies in two key challenges of anomaly detection in enterprise environments. First, labeled attack data is rarely available, and the heterogeneity between infrastructures and behaviors limits the reuse of public datasets, which makes unsupervised approaches especially valuable. Second, most previous contributions focus only on numerical features, while enterprise logs also contain textual elements that are often ignored. In this work, we focus on executable paths as textual features, while noting that other sources, such as web addresses or email content, could also be explored in future research. In response, we propose a UEBA framework that integrates deep AEs with text embeddings, grounded on solid mathematical foundations.

Our work makes a twofold contribution. First, we provide novel theoretical results by proving the

equivalence of two common definitions of fully-connected neural networks. Second, to the best of our knowledge, we present the first implementation of an explainable UEBA-based anomaly detection framework using autoencoders. Our methodology includes the use of text encoding models (Doc2Vec) alongside AEs to leverage both numerical and textual data, training unlabeled (possibly contaminated) data, and using model residuals for explainability.

Table 1. Comparison of related UEBA and anomaly-detection methods highlighting gaps addressed by our approach.

Method	Domain	Novelty / Differences w.r.t. this work
Shashanka et al. (2016) [3]	UEBA	Mahalanobis distance + SVD for user/device modelling; limited to numerical features without explainability; our work adds multimodal logs and interpretable residuals.
Voris et al. (2019) [5]	UEBA	GMM with decoys for active authentication; trap-based, not scalable to enterprise monitoring; our work scales to large heterogeneous logs.
Pusara & Brodley (2004) [6]	Biometrics	Mouse-movement re-authentication; narrow biometric focus; our work targets full enterprise behavior.
Slipenchuk & Epishkina (2019) [7]	UEBA	Survey of UEBA statistical vs. ML methods; descriptive only, no unified DL framework; our work proposes a practical autoencoder pipeline.
Meng et al. (2018) [8]	Biometrics	Touch behavior authentication with RBFN+PSO; mobile-focused, limited features; our work generalises to multimodal enterprise UEBA.
Martín et al. (2021) [9]	UEBA	UEBA for federated identity (OpenID Connect); restricted to identity scope; our work expands to enterprise-wide logs with textual features.
Morales-Forero & Bassetto (2019) [12]	Industrial	Semi-supervised AE for Industrial Control Systems (ICS) anomaly detection; domain-specific, no text integration; our work extends AE to enterprise logs with multimodal features.
González-Muñiz et al. (2022) [13]	Industrial	VAE with two-step residual classification for engineering systems; limited to industrial datasets; our work adapts VAE to enterprise UEBA with textual features and explainability.
Sakurada & Yairi (2014) [21]	Satellite	AE for anomaly detection in satellite telemetry; domain-limited; our work applies AE to enterprise UEBA with multimodal features.
Wang et al. (2022) [22]	Satellite	Likelihood-based AE for telemetry anomaly detection; strong in satellite domain, not generalisable; our work extends to enterprise logs with interpretability.
Zhou & Paffenroth (2017) [23]	Industrial	Robust deep AE for anomaly detection; designed for ICS data; our work adapts robust AE concepts to UEBA with text+numeric logs.
Ortega-Fernández et al. (2023) [24]	Industrial	Deep AE network intrusion detection system designed to detect distributed denial-of-service (DDoS) attacks in ICS; effective but network-traffic only; our work generalises AE to enterprise UEBA with multimodal logs and interpretability.

This paper is structured as follows. Section 2 details the proposed methodology, including theoretical results and a description of the methods employed for feature extraction, residual space analysis, and the proposed architecture for UEBA-based anomaly detection. Section 3 presents the

results of the application of the proposed methodology to a real use-case of cybersecurity in a financial institution. Finally, Section 4 outlines the main conclusions and future research directions.

2. Materials and methods

In this work, we propose a novel UEBA-based anomaly detection framework based on deep AEs and the Doc2Vec algorithm for the preprocessing of text features. In the following Subsections 2.1–2.4, we describe the mathematical foundations of the used algorithms and present our theoretical contributions with a new proof of the equivalence of two common definitions of neural networks. Moreover, Subsection 2.5 describes the architectures of the UEBA-based anomaly detection framework.

2.1. Neural networks

Neural networks constitute a large set of learning models that originate from the early work on the Rosenblatt perceptron [25]. They approximate functions by interleaving affine transformations with nonlinear activation functions. A feed-forward deep neural network uses longer chains of concatenated affine and activation functions to improve the representation of the target function. This definition of a feed-forward deep neural network can be formally expressed in the following manner.

Definition 1. *Let $\mathcal{F} \subset \{\varphi : \mathbb{R} \to \mathbb{R}\}$ be a set of activation functions. Given $d \geq 2$ and an input dimension $n^{(0)} = n$, for each $l = 1, \ldots, d$, let $n^{(l)} \in \mathbb{Z}^+$ with $n^{(d)} = m$, and let $A^{(l)} : \mathbb{R}^{n^{(l-1)}} \to \mathbb{R}^{n^{(l)}}$ be affine transformations. Define $\Phi^{(l)} : \mathbb{R}^{n^{(l)}} \to \mathbb{R}^{n^{(l)}}$ by*

$$\Phi^{(l)}(\mathbf{x}) = (\varphi_1(x_1), \ldots, \varphi_{n^{(l)}}(x_{n^{(l)}})),$$

with each $\varphi_j \in \mathcal{F}$ for $j = 1, \ldots, n^{(l)}$. Then, a feed-forward deep neural network with $d-1$ hidden layers is the function

$$\hat{f} = \Phi^{(d)} \circ A^{(d)} \circ \Phi^{(d-1)} \circ A^{(d-1)} \circ \cdots \circ \Phi^{(1)} \circ A^{(1)} : \mathbb{R}^n \to \mathbb{R}^m.$$

However, as noted in [26], this definition does not uniquely determine the network structure and makes it difficult to formalize concepts such as sparsity and convolutions. An alternative, more constructive description is based on a layered graph where each node (or neuron) implements a simple function, as formalized by [27, 28]. For brevity, we omit the definitions of layered graph and neuron, and they can be found in [28].

Definition 2. *Given a layered graph $\mathfrak{G}$, a feed-forward deep neural network with structure $\mathfrak{G}$ is any function defined on $\mathfrak{G}$ ($\mathfrak{G}$-function) such that each constituent function is a neuron.*

Neural networks of fixed depth or fixed width can approximate a wide range of functions modeling real-life processes [29, 30]. The universality property is crucial for any application; hence, it is significant to prove that both definitions are equivalent. Below, we provide a mathematical proof (Proposition 1) demonstrating this equivalence:

Proposition 1. *Definitions 1 and 2 are equivalent.*

Proof. We prove the equivalence of Definitions 1 and 2.

(Definition 1 ⇒ Definition 2): Let $\hat{f} = \Phi^{(d)} \circ A^{(d)} \circ \cdots \circ \Phi^{(1)} \circ A^{(1)}$ with

$$\hat{f}^{(i)} = \Phi^{(i)} \circ A^{(i)} : \mathbb{R}^{n^{(i-1)}} \to \mathbb{R}^{n^{(i)}}, \quad n^{(0)} = n,\ n^{(d)} = m.$$

For $\mathbf{x} \in \mathbb{R}^n$, set

$$\mathbf{h}^{(0)} = \mathbf{x}, \quad \mathbf{h}^{(i)} = (\hat{f}^{(i)} \circ \cdots \circ \hat{f}^{(1)})(\mathbf{x}).$$

Since each coordinate

$$\pi_j(\hat{f}^{(i)}(\mathbf{h}^{(i-1)})) = \varphi_j\Big(W_j^{(i)} \cdot \mathbf{h}^{(i-1)} + b^{(i)}\Big)$$

defines a neuron (i.e., the function $\pi_j \circ \hat{f}^{(i)}$), we construct a layered graph $\mathfrak{G} = (V, E)$ by:

(1) *Input layer:* $V^{(0)}$ consists of n nodes (the input coordinates $\mathbf{x}$).

(2) *Layers* 1 *to d:* For each i, let $V^{(i)}$ consist of $n^{(i)}$ nodes, where each node $v_j \in V^{(i)}$ is assigned the function $\pi_j \circ \hat{f}^{(i)}$ and has incoming edges

$$I_v^- = \{(u, v) :\ u \in V^{(i-1)}\}.$$

Thus, $\hat{f}$ is a $\mathfrak{G}$-function.

(Definition 2 ⇒ Definition 1): Conversely, let $\mathfrak{G} = (V, E)$ be a layered graph with layers

$$V^{(i)} = \{v_1, \ldots, v_{n^{(i)}}\} \quad (n^{(0)} = n,\ n^{(d)} = m),$$

and let each node $v \in V^{(i)}$ have an associated function f_v. For $\mathbf{h}^{(i-1)} \in \mathbb{R}^{n^{(i-1)}}$, let $\mathbf{z}_v$ be the subvector of inputs corresponding to the predecessors of v. Define

$$\hat{f}^{(i)}(\mathbf{h}^{(i-1)}) = (f_{v_1}(\mathbf{z}_{v_1}), \ldots, f_{v_{n^{(i)}}}(\mathbf{z}_{v_{n^{(i)}}})).$$

Then, the overall network can be written as $\hat{f} = \hat{f}^{(d)} \circ \cdots \circ \hat{f}^{(1)}$, which is of the form given in Definition 1.

□

2.2. AEs

An AE is a model that approximates the identity function under a constraint that forces the model to capture the most salient features of the input. In our case, for a random sample $X = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$ with $\mathbf{x}_i \in \mathbb{R}^n$, we define an AE $\mathbf{AE}_n^p$ as follows.

Definition 3. *Given positive integers n and p (with $p < n$), an AE $\mathbf{AE}_n^p$ is a tuple*

$$(n, p, f, g, \mathcal{E}, \mathcal{D}, X, \Delta)$$

where:

- *$\mathcal{E}$ and $\mathcal{D}$ are sets of functions from $\mathbb{R}^n$ to $\mathbb{R}^p$ and from $\mathbb{R}^p$ to $\mathbb{R}^n$, respectively;*
- *$f \in \mathcal{E}$ is the encoder and $g \in \mathcal{D}$ is the decoder;*
- *Δ is a dissimilarity measure (typically a metric) on $\mathbb{R}^n$.*

The latent space is the codomain of f, where the compressed representation of $\mathbf{x}$ is stored. The reconstruction error is defined as:

Definition 4. *The reconstruction error of the AE is given by*

$$E_{f,g}(X) = \sum_{i=1}^{m} \Delta(g(f(\mathbf{x}_i)), \mathbf{x}_i).$$

Training an AE involves finding functions f and g that minimize $E_{f,g}(X)$. In our work, both encoder and decoder are implemented as fully connected (regularized) neural networks. We focus on under-complete AEs ($p < n$) to force a compressed representation, which in turn leads the model to learn the dominant patterns of normal behavior. Since anomalous data points are rare, the model prioritizes the reconstruction of normal samples, making the reconstruction error an effective anomaly score.

It should be highlighted that Proposition 1 establishes that the two common formalizations of feed-forward neural networks (functional composition vs. layered graphs) are equivalent. Since our encoder and decoder are implemented as fully connected networks, this equivalence guarantees that the AE used in our UEBA pipeline is a well-defined object in either formalism, with no loss of generality when moving between them. This allows us to apply the classical universal approximation results for feed-forward networks [29, 30], providing a rigorous foundation for our modeling choice: the AE has sufficient expressivity to capture the dominant patterns of normal behavior, while its residuals define a mathematically coherent and explainable anomaly score used throughout the framework.

2.3. Doc2Vec

To process text-based variables (e.g., lists of executed processes), we use the Doc2Vec model [31]. Doc2Vec is a neural network-based embedding method that learns vector representations of documents in an unsupervised manner. Similar to Word2Vec [32], it clusters similar texts in the vector space. In the same way that Word2Vec embeds related words like synonyms or topics close in the vector space, Doc2Vec also clusters similar texts together, for example, by identifying the topic of the text or by finding similar words between texts. Two main algorithms exist for training Doc2Vec: distributed bag of words (DBOW) and distributed memory (DM). In DBOW, the document vector is used to predict random word vectors from the document; in DM, both the document vector and word vectors are used to predict the next word in a sequence. The resulting document embeddings capture semantic similarities that are later used in our UEBA framework by applying the Doc2Vec trained with DBOW to extract information from the lists of processes recovered from the activity logs. We specifically adopt Doc2Vec with the DBOW algorithm because executable paths behave more like structured categorical tokens than natural language, so co-occurrence at the window level captures useful behavioral similarity (e.g., programs launched together) without requiring heavy language models.

2.4. t-distributed stochastic neighbor embedding

The t-distributed stochastic neighbor embedding (t-SNE) [33] is a dimensionality reduction technique used primarily for visualizing high-dimensional data in two or three dimensions. It preserves local structures by mapping similar points from high-dimensional space to nearby points in the lower-dimensional embedding, while distant points remain separated. t-SNE works by first

computing pairwise conditional probabilities based on distances between points in the high-dimensional space, and modeling the local similarities as Gaussian distributions. It then maps these points into a lower-dimensional space using a Student's t-distribution to model the similarity of the embeddings. The algorithm iteratively adjusts embedding positions by minimizing the Kullback-Leibler divergence between these two distributions:

$$\mathrm{KL}(P\|Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}},$$

where P and Q represent the joint probability distributions of pairwise similarities between data points in the high-dimensional and low-dimensional spaces, respectively. This lower dimensionality embedding allows us to identify patterns that are characterized by their local structure, such as clusters and anomalies in high dimensional datasets.

We use t-SNE to analyze the dataset and validate the AE model's behavior. We apply t-SNE to the test data to analyze the presence of clusters indicative of different user groups and behaviors, and validate the feasibility of a UEBA-based approach. Moreover, we also use t-SNE to study the residuals of the reconstruction error of the model, verifying that anomalous data points remain distinguishable within the residual space.

2.5. UEBA-based anomaly detection framework

The dataset used in this study is derived from multiple real data sources, including Windows events of user activity, emails, and antivirus logs from a financial institution. For this reason, preprocessing plays a crucial role in preparing the raw data for effective anomaly detection. The preprocessing steps involve cleaning, transforming, and encoding the data using the Doc2Vec model described in Section 2.3 to derive features into a format suitable for the anomaly detection model.

The data is ingested as time-series from logs and is aggregated into fixed windows, summarizing them into key statistics, like total counts and average time intervals. In addition, we perform feature engineering to derive new variables that better capture behavioral patterns, including metrics such as the average time between logins, the ratio of failed to successful logins, and the frequency of antivirus alerts. The set of derived features is detailed in Table 2, with 2 indexing variables and 19 features.

For handling missing values, a lack of activity is assumed to correspond to a zero count, while timing variables (e.g., `avg_sec_bet_logins`) are imputed using the duration of the aggregation window in seconds as a maximum time. Meanwhile, text data (specifically, the executable names from processes executed within each window) are combined into a single field (`process_list`) and encoded using a Doc2Vec model (trained with DBOW) to generate a 64-dimensional embedding vector for each window. These embeddings are concatenated to the derived numerical features, obtaining the final input vector with 83 features.

Finally, all numerical features are normalized using a robust scaler followed by a min-max scaler to ensure that each variable contributes equally to the model. At the end of this preprocessing pipeline, the dataset is clean, structured, and ready to be used in the UEBA models and anomaly detection framework.

Table 2. Variables collected in the dataset.

Variable	Description	Type
`time`	Date and time when the data was generated (used for window aggregation and indexing only)	date
`CallerUser`	Username (used for role aggregation and indexing only).	factor
`WorkstationName`	Machine name where the data was generated.	factor
`num_new_process`	Number of new processes created (e.g., Windows event 4688).	numeric
`num_logins`	Number of successful logins (e.g., Windows event 4624).	numeric
`avg_sec_bet_logins`	Average time in seconds between successful logins.	numeric
`num_f_logins`	Number of failed login attempts (e.g., Windows event 4625).	numeric
`avg_sec_bet_f_logins`	Average time between failed logins.	numeric
`num_antivirus_alerts`	Number of incidents detected by the antivirus.	numeric
`num_firewall_alerts`	Number of incidents detected by the firewall.	numeric
`sent_emails`	Number of emails sent by the user.	numeric
`received_emails`	Number of emails received by the user.	numeric
`incident_emails`	Number of emails flagged as incidents.	numeric
`sent_emails_size`	Total size of sent emails (body and attachments).	numeric
`received_emails_size`	Total size of received emails (body and attachments).	numeric
`sent_email_files`	Number of file attachments in sent emails.	numeric
`received_email_files`	Number of file attachments in received emails.	numeric
`sent_email_links`	Number of web links in sent emails.	numeric
`received_email_links`	Number of web links in received emails.	numeric
`4100_events`	Number of PowerShell errors (e.g., Windows event 4100).	numeric
`4104_events`	Number of remote PowerShell commands executed (e.g., Windows event 4104).	numeric

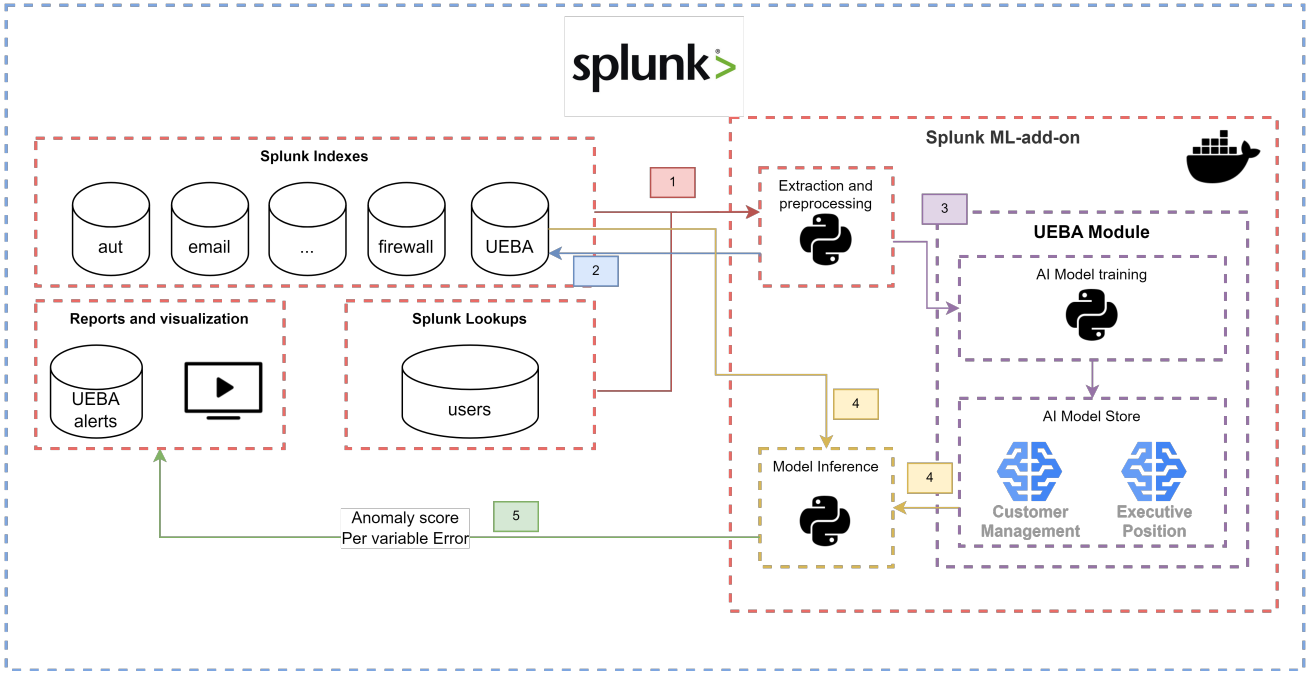


Figure 1. Architecture of the UEBA-based anomaly detector.

Figure 1 illustrates the overall architecture of the proposed UEBA-based anomaly detection framework, where the numbered arrows correspond to the different stages of the process (1–5). The process starts with data collection (1) from multiple sources (e.g., Windows events, emails, antivirus,

firewall), which are stored in Splunk Enterprise, a common choice in the industry to collect, index, search, analyze, and visualize large volumes of machine-generated data in real time, as time-series events. The feature extraction pipeline aggregates and processes these events into summary variables, while the text data is encoded using Doc2Vec (see Table 2). Once the raw data has been transformed into security events and the features have been computed, this aggregated data is grouped into entities based on business roles, e.g. customer (CM) and executive positions (EPs) and stored at the UEBA index in Splunk (2). This aggregated UEBA dataset can now be used to train the AE and Doc2Vec models (see Algorithm 1), which are later stored in a model store (3). The hyperparameters used to train both models are available in Table 3, and the network diagram is presented in Figure 2. Once the models are trained and stored, they are made available through the model store (4) to generate predictions for new data (see Algorithm 2) coming from the index (4) in near real-time, including the residual-based anomaly scores which are stored in the alert database, and therefore are available for plotting and advanced analysis (5).

Table 3. Hyperparameters of the UEBA anomaly detection framework for the CM and EPs roles.

Component	**Setting**
Doc2Vec	64-dimensional embeddings; DBOW architecture; window size = 5; epochs = 20
AE	Input size = 83 features; hidden layers = [64, 32, 16, 8, 16 32, 64]; latent dimension = 8; activations = ELU (internal), `tanh` (first/last)
Training	Optimizer = Adam (lr = 0.001 (CM) / 0.01 (EPs)*); batch size = 64 (CM) / 256 (EPs); early stopping (patience = 10, monitor = validation MSE); L_1 regularization (λ = 0.001); validation split = 20%

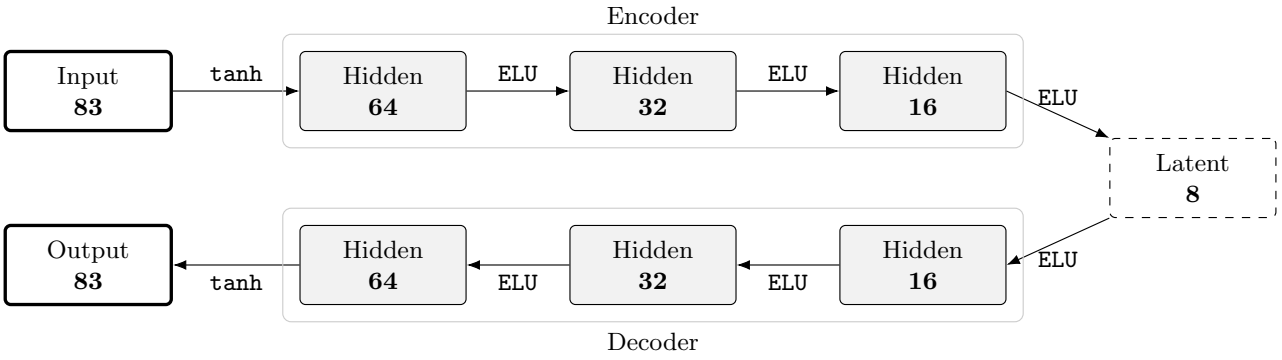


Figure 2. Network diagram of the proposed AE.

*The higher learning rate in the EPs role was selected via grid search in combination with the larger batch size, resulting in stable convergence and improved generalization.

Algorithm 1: Training of the UEBA framework

Input: Windowed log events $\{\mathcal{E}_i\}$

Output: Trained AE f_θ, Doc2Vec encoder ϕ, scaler, decision threshold τ

(1) Extract numerical features $n_i = g_{\text{num}}(\mathcal{E}_i)$.

(2) Encode process list tokens $\mathcal{T}_i$ with Doc2Vec (DBOW) to obtain $t_i = \phi(\mathcal{T}_i)$.

(3) Concatenate: $x_i = [n_i; t_i]$, then apply scaling to get $\tilde{x}_i$.

(4) Train deep AE f_θ to minimize reconstruction loss with L_1 regularization and early stopping.

(5) Compute scores $s_i = \|\tilde{x}_i - f_\theta(\tilde{x}_i)\|_1$ on validation data.

(6) Set threshold τ as the 95th percentile of $\{s_i\}$.

Algorithm 2: Inference and anomaly scoring

Input: New window $\mathcal{E}$; trained f_θ, ϕ, scaler; threshold τ

Output: Score s, residual vector r, decision $y \in \{\text{normal}, \text{anomaly}\}$

(1) Extract features: $n = g_{\text{num}}(\mathcal{E})$, $t = \phi(\mathcal{T})$, $x = [n; t]$.

(2) Scale input: $\tilde{x} = \text{scale}(x)$.

(3) Reconstruct: $\hat{x} = f_\theta(\tilde{x})$.

(4) Compute residuals $r = \tilde{x} - \hat{x}$ and anomaly score $s = \|r\|_1$.

(5) Decision: $y = \mathbb{I}[s \geq \tau]$. Return (s, r, y) for alerting and visualization.

We generated two different datasets of one year of historical data, one for the CM group (25313 records) and the other for the EPs group (9804 records). Both datasets contain records of user behavior that have been cleared as normal behavior by the existing security filters from the institution; however, they may contain a small number of anomalies that evaded these measures. For this reason, we assume the data to be contaminated data [34], and thus unlabeled. Both datasets are split into training, validation, and testing sets using a standard split: 20% of the records are held out as the testing set, and the remaining 80% is used for training, of which 20% is further reserved as a validation set. We train a separate deep AE model for each of the user groups. The model architecture starts with 83 input features (including the 19 aggregated features and the 64 components of the Doc2Vec transformation) and compresses them to an 8-dimensional latent space via three hidden layers (with 64, 32, and 16 neurons, respectively). The encoder and decoder networks use Exponential Linear Unit (ELU) activations (except in the first and last layers, which use `tanh`). Training employs the Adam optimizer, early stopping, and L_1 regularization. A decision threshold τ is determined as the 95th percentile of the reconstruction error on the validation set.

Trained models are stored using MLflow (an open-source platform designed to manage the end-to-end ML life-cycle), and later deployed to analyze incoming data. The anomaly score (based on the reconstruction error) and auxiliary statistics are sent back to Splunk for reporting and further analysis.

3. Results and discussion

We evaluate the proposed UEBA framework in two complementary settings. First, we conduct experiments on real-world data collected from a financial institution in Section 3.1, providing a strong validation of the proposed framework under real operational conditions. Next, we extend these findings in Section 3.2 using simulated data sampled from real attack scenarios, allowing us to systematically probe the system's response to specific threat vectors and anomalous behavior of different intensities. By combining these two perspectives—actual enterprise data and controlled simulations—our analysis offers a robust demonstration of how UEBA can enhance security monitoring, highlight anomalous user or entity behavior, and detect sophisticated cyber threats in an explainable manner.

In the following sections, we present the results of these evaluations, focusing on detection rate, residual-space analysis using t-SNE projections, performance under synthetic anomalies, and overall explainability.

3.1. Performance on real data

Evaluating unsupervised models is challenging in the absence of labeled data in real scenarios. To evaluate the anomaly detection capabilities of the proposed framework, we assess its ability to learn normal behavioral patterns by analyzing the positive rate on the test set. The decision threshold τ is fixed at the 95th percentile of the reconstruction error on the validation set. Table 4 shows the positive rates for both user groups (CM and EPs). We can observe how positive rates for both groups are close to the 5% value, indicating that the models are well-calibrated. While the overall calibration is consistent across groups, we note that the positive rate of the EPs model (4.61%) is slightly below the 5% target, whereas the Customer Management model (5.09%) is slightly above it. This difference is minor in absolute terms, but it reflects the fact that user groups with fewer samples and more heterogeneous activity patterns may lead to tighter thresholds and a slightly more conservative model. Such variations are expected in UEBA applications, where behavior differs by role, and they showcase the importance of tailoring models to specific roles or groups.

Table 4. Positive rates for the two AE models.

UEBA model	Positive rate
Customer management	5.09%
Executive positions	4.61%

These positive-rate results constitute the primary evaluation of our framework in an unsupervised setting. Since the threshold was fixed during training, the fact that calibration holds on the test set shows that the models generalize well without relying on labeled anomalies. Standard supervised metrics such as precision, recall, or F1 cannot be meaningfully computed here, since the dataset lacks anomaly labels, and the data is taken as possibly contaminated data. In addition, as reported in Section 3.2, we were provided with a small set of attack events. Although these real anomalies were valuable, their very limited number made the evaluation less informative: the models detected them almost trivially, with a true positive rate close to 100%. This limitation motivated the use of synthetic anomaly experiments, which should be regarded as a complementary stress test probing robustness under varying anomaly

intensity rather than as the main evaluation.

(a) Normal data. **(b)** Residuals.

Figure 3. t-SNE representation of CM. (a) Normal test data and (b) corresponding residuals.

Moreover, to better understand the model's learning effectiveness, we perform a t-SNE projection of both the original test data and the corresponding residuals. Figure 3 shows a t-SNE projection of normal test data from the CM group (left) and the corresponding residuals (right). We can observe how the residual space shows less dependency clustering compared to the original data, confirming that the model has successfully captured the dominant patterns of normal behavior. The residual distribution reveals anomalies as scattered points on the edges.

3.2. Performance on synthetic anomalies

To assess the model's capability in detecting real-world anomalies, we conducted an experiment with synthetic anomalies generated from 10 real attack scenarios provided by the financial institution (login anomalies, antivirus incidents, email anomalies, and process-related anomalies). These synthetic anomalies are generated by taking convex combinations of the real anomalies with normal behavior data. This procedure allows us to increase the sample size of the test set, using the variability of normal behavior to provide more varied anomalies and study the model's detection capability as a function of an anomaly intensity factor, λ_k. Particularly, for each $j = 1, \ldots, 10$, and for $k = 1, \ldots, 100$, we obtain a synthetic test set as follows

$$\mathbf{a}_k^{*j} = \mathbf{z}_j(1 - \lambda_k) + \lambda_k \mathbf{a}_j,$$

where $\mathbf{z}_j$ is a randomly sampled element with normal behavior data from the test set, $\lambda_k \in [0, 1]$ is the anomaly intensity factor, which takes values in steps of 0.01, and $\mathbf{a}_j$ is a real-type anomaly. Note that, with this procedure, we obtain a synthetic test set of sample size 1000.

This convex interpolation scheme provides a simple yet controlled way to validate the anomaly detection sensitivity. By varying the anomaly intensity factor λ, we can gradually shift normal samples toward real anomalies and observe how the model responds, while ensuring that the resulting points remain inside the convex hull of the observed feature space, reducing the risk of generating unrealistic samples. At the same time, this construction mainly alters feature magnitudes and does not capture temporal dependencies, cross-feature correlations, or coordinated attack strategies. Therefore,

synthetic anomalies should be regarded as a complementary stress test of robustness, rather than a substitute for larger-scale evaluation on real-world attack data.

Figures 4 and 5 present the detection rate as a function of the anomaly intensity factor λ for both user groups (CM and EPs). The results demonstrate that the models reliably detect anomalies when $\lambda > 0.7$ for all anomaly types. For specific types, such as login and antivirus anomalies, detection occurs at much lower intensity levels ($\lambda > 0.2$). In contrast, process anomalies require higher intensity levels for reliable detection, primarily due to the complexity of encoding text-based data. While Doc2Vec provides a compact and efficient encoding, its averaging nature can obscure rare but highly informative tokens. More expressive embeddings could, in principle, capture richer temporal and syntactic structure, though at higher computational cost (see Section 4 for possible alternatives for future work).

These detection-vs-intensity curves serve as surrogate evaluation metrics in the absence of extensive labeled examples. Rather than relying on precision or recall, which cannot be meaningfully computed here, we evaluate how detection rates evolve when normal samples are gradually shifted towards known attack behaviors. This provides a principled and interpretable way to assess sensitivity under scarce-label conditions, showing that the framework can reliably separate normal from anomalous behavior once the anomalies reach sufficient intensity. In this sense, the detection–vs–intensity curves can be interpreted as an analogue of statistical power curves: the anomaly intensity parameter λ plays a role similar to the effect of the sample size, and the resulting curves quantify how quickly the detector achieves high detection probability as anomalies become more pronounced.

Figure 6 shows a t-SNE embedding of the test set. Each color represents a different anomaly type (login, email, antivirus and process), with color and saturation indicating the type and intensity of anomalies. The results confirm that anomalies become increasingly distinguishable as their intensity (λ) increases. Notably, anomalies with high λ values form clear clusters in the residual space, confirming the model's capability to separate abnormal behavior from normal patterns. Figure 6a illustrates how synthetic anomalies are embedded alongside normal data in the original feature space, while Figure 6b shows the corresponding residuals, where anomalies show clearer clusters, thus being easier to identify. Process-related anomalies appear less distinct due to their complexity and dependence on text-based feature encoding.

At last, we will assess the proposed methodology's ability to provide explainable model results through the per-feature reconstruction error. Figure 7 shows the logarithm of the per-feature reconstruction error for each model for fully anomalous data ($\lambda = 1$). We can observe that they are easier to detect, and this effect becomes more evident, with higher errors appearing on features related to the anomaly. For instance, the email-related anomalies show the highest reconstruction error on the `sent_email_*` variables. In addition, in the case of login anomalies, the highest errors are observed in the antivirus and login variables.

However, for anomalies that are more challenging, such as process anomalies, the per-variable reconstruction error alone may not suffice in identifying the origin of the anomaly. We speculate that the reason these anomalies are harder to detect and explain is that they heavily depend on the process encoding by Doc2Vec. However, we have to remark that, when removing the text variables, the detection performance worsened, so the text encoding provides valuable information for the detection of these anomalies, even if it's not enough for clear interpretation.

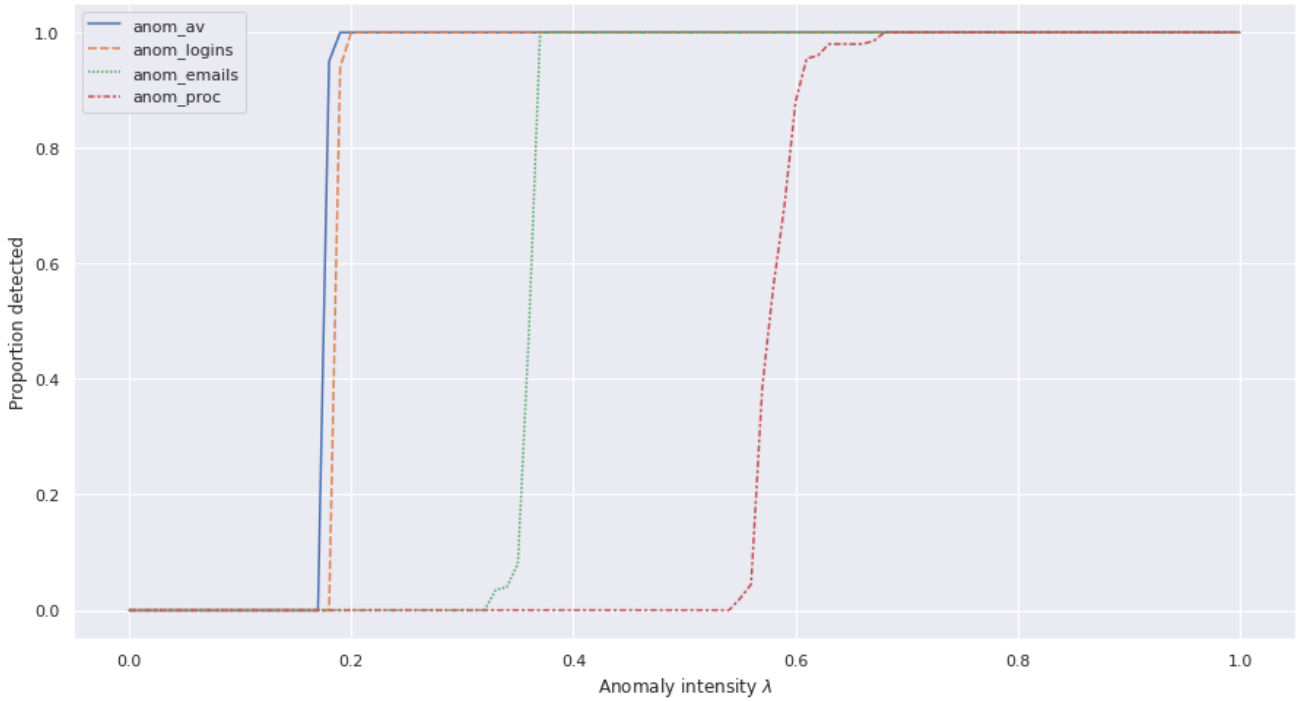


Figure 4. Anomaly detection rates as a function of anomaly intensity for each model for the CM model.

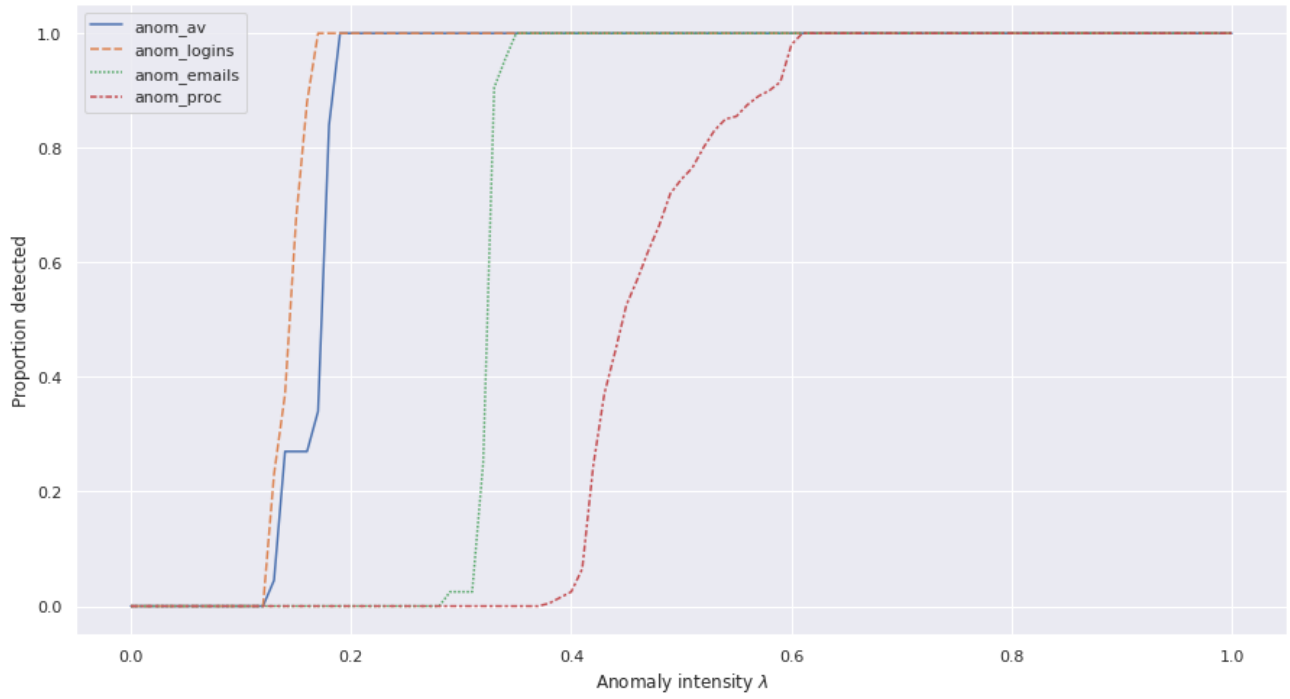
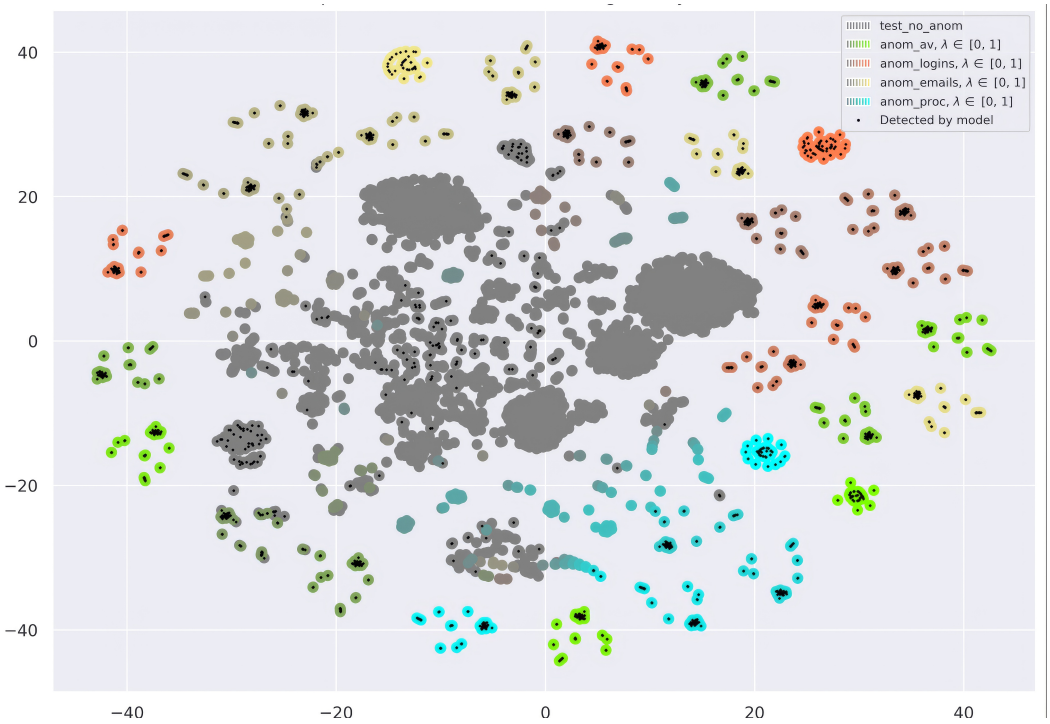
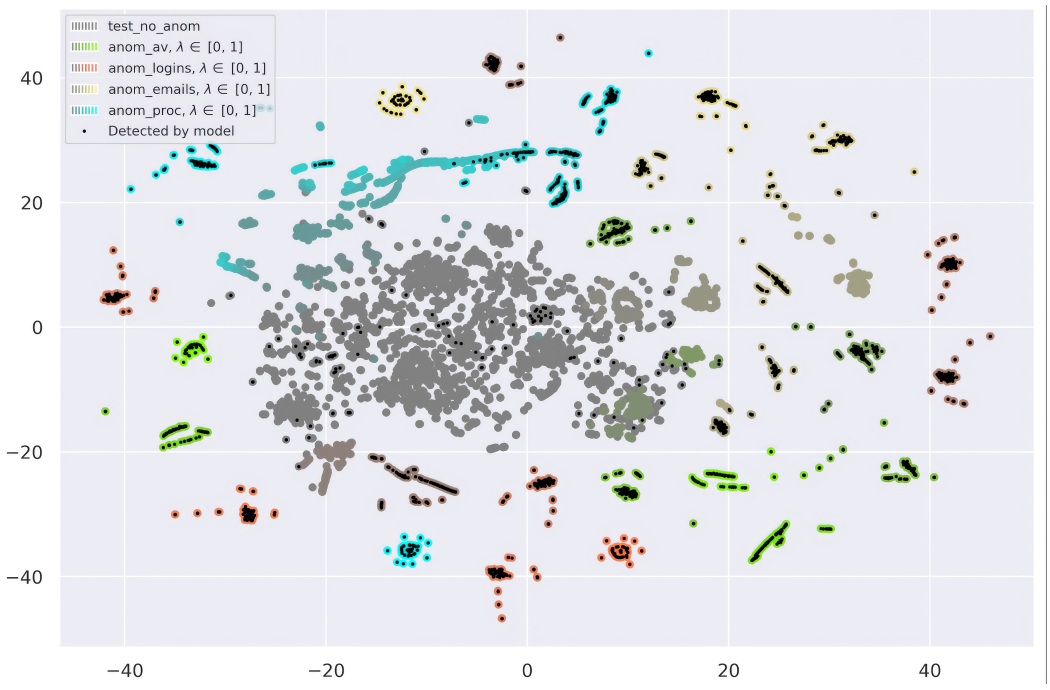


Figure 5. Anomaly detection rates as a function of anomaly intensity for each model for the EPs model.

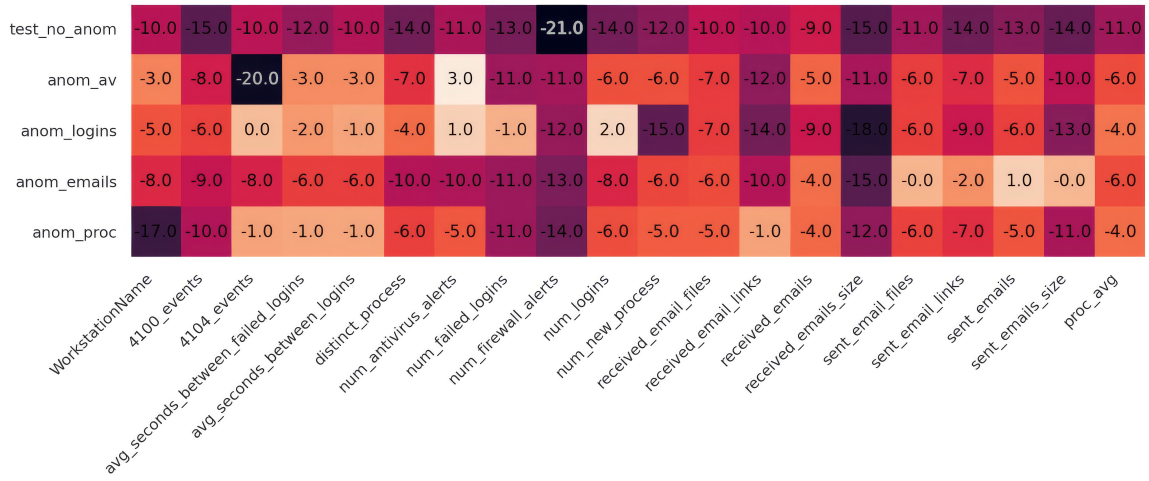


(a) Sample of test data with anomalies.

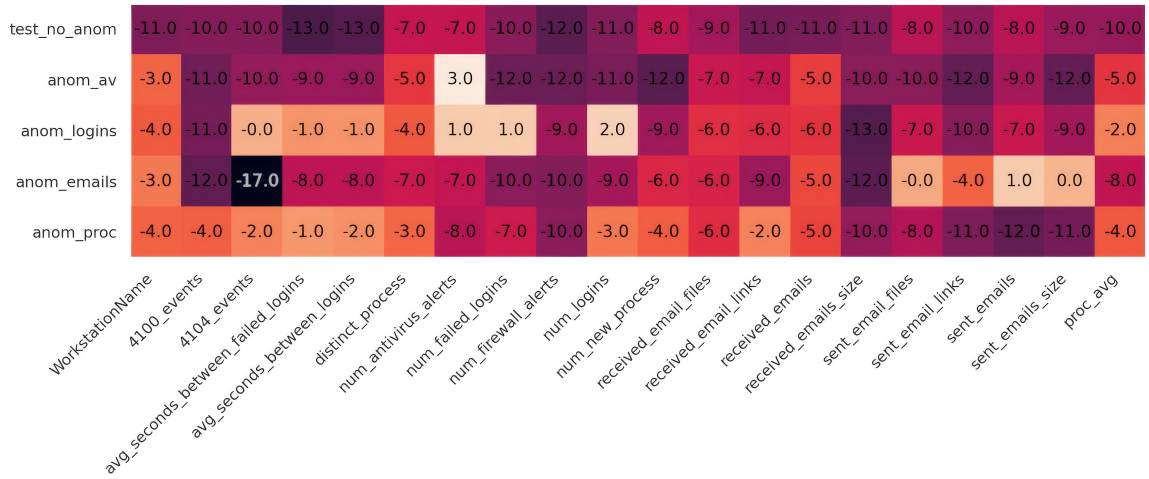


(b) Corresponding residuals.

Figure 6. t-SNE representations of data and residuals for the CM group. Dotted points indicate instances flagged as anomalies. Saturation indicates the intensity of anomalies.



(a) Customer Management.



(b) Executive Positions.

Figure 7. Logarithm of the reconstruction error per feature for each model.

We can observe how features related to login patterns, email attachment size, and failed login attempts exhibit the highest contributions to the anomaly score, offering clear indications of potential security incidents. Anomalies in email activity, such as unusually large attachments or an abnormal volume of sent emails, are immediately recognizable. Process-related anomalies, on the other hand, show a more distributed error pattern due to the diverse and complex nature of process command sequences.

These results highlight the strength of the proposed framework in detecting a wide range of anomalies with high accuracy while being explainable. The combination of deep AEs and Doc2Vec allows for an effective integration of numerical and text-based features, providing a comprehensive view of user behavior. The ability to visualize residuals and analyze feature-level contributions significantly enhances explainability, making the framework more practical and effective for its use in cybersecurity applications.

4. Conclusions

This study presents a UEBA-based anomaly detection framework that leverages deep AEs to identify suspicious activities within a real-world cybersecurity use case in a financial institution. By integrating both numerical and text-based features through Doc2Vec embeddings, the proposed approach can capture complex behavioral patterns to detect anomalies that may otherwise remain undetected by more traditional methods. A key contribution is a novel theoretical result proving the equivalence of two common definitions of fully connected neural networks, thereby grounding our AE design within universal approximation theory. In addition, we provide a set of experimental evaluations to showcase how advanced deep learning techniques can be employed for explainable, behavior-based anomaly detection.

Experimental evaluations showed that the proposed anomaly detection framework achieves a high detection rate, even in challenging conditions with contaminated training data. Additional experiments with synthetic anomalies—reflecting anomalous login patterns, email activity anomalies, and antivirus alerts—confirmed the framework's robustness and adaptability to diverse cybersecurity threats. A key advantage of our method lies in the ability to perform a residual-based analysis, which enhances explainability by pinpointing specific features that deviate from the normal baseline profile. This explainability is essential in practical cybersecurity contexts where timely responses and a deep understanding of anomalies can significantly improve investigation efficiency and reduce false alarms.

Compared to more traditional anomaly detection techniques such as principal component analysis (PCA) based monitoring, statistical thresholds, or clustering approaches, our framework offers several distinctive advantages. As also summarized in Table 1, classical methods typically assume clean or labeled training data, can only be applied to numerical features, and provide limited interpretability. In contrast, our approach is explicitly designed to operate under data contamination, integrates both numerical and textual features through Doc2Vec, and leverages residual-based scores to deliver ante-hoc explanations. These aspects underline the novelty and uniqueness of our framework in enterprise UEBA scenarios.

While the proposed framework demonstrated strong performance, several limitations should be acknowledged. First, the amount of labeled anomalies was extremely limited, which prevented the use of standard supervised metrics and restricted baseline comparisons. Second, the reliance on Doc2Vec embeddings for process features, while efficient, may obscure rare but highly informative tokens. Finally, the evaluation was performed on sensitive institution-specific data that cannot be released, which constrains reproducibility.

These limitations point to several directions for future research. A deeper and systematic comparison with standard unsupervised baselines (e.g., Isolation Forest, Local Outlier Factor, one-class SVM) remains a priority, ideally using a public benchmark dataset such as the community emergency response team (CERT) insider threat dataset [35] or datasets with more textual features. Future work should also explore more expressive text encoders, such as transformer-based or recurrent architectures, especially for longer and more natural textual features (e.g., email content, web addresses). In parallel, advancing explainability remains essential. Our residual-based approach already highlights which features are most difficult to reconstruct, offering ante-hoc insights into the features that most influence anomaly detection. Post-hoc techniques such as Shapley values [36] or LIME [37] could complement this by accounting more explicitly for feature interactions and

correlations, providing analysts with a second layer of explanation to validate and enrich our residual-based insights. Moreover, counterfactual analyzes [38] further extend this perspective by showing what minimal behavioral changes would render an anomalous user or entity normal, which could help analysts perform root-cause analysis and guide remediation strategies. Finally, ensemble strategies that combine complementary anomaly detectors offer a promising route to enhance robustness and reliability in practical deployments.

Data availability

The real-world enterprise logs analyzed in this study were provided by a financial institution under confidentiality and security agreements and are not publicly available. Aggregate statistics and the synthetic anomaly generation procedure are described in Section 2 and Subsection 3.2.

Author contributions

Jose Fuentes: Conceptualization, Software, Formal analysis, Investigation, Data curation, Writing—original draft, Visualization; Ines Ortega-Fernandez: Conceptualization, Validation, Formal analysis, Resources, Writing—original draft, Writing—review & editing, Supervision, Project administration, Funding acquisition; Nora M. Villanueva: Conceptualization, Validation, Formal analysis, Writing—original draft, Writing—review & editing, Supervision; Marta Sestelo: Conceptualization, Validation, Formal analysis, Resources, Writing—review & editing, Supervision, Funding acquisition. All authors have read and agreed to the published version of the manuscript.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was partially supported by the European Union's Horizon Europe Research and Innovation programme under the project PRESERVE (Grant Agreement N°101168309) and by the grant PID2020-118101GB-I00 from the Ministerio de Ciencia e Innovación (MCIN/AEI/10.13039/501100011033).

Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. The funding sources listed in the Acknowledgments had no role in the study design; in the collection, analysis, or interpretation of data; in the writing of the report; or in the decision to submit the article for publication.

References

1. U. Inayat, M. F. Zia, S. Mahmood, H. M. Khalid, M. Benbouzid, Learning-based methods for cyber attacks detection in IoT systems: A survey on methods, analysis, and future prospects, *Electronics*, **11** (2022), 1502. https://doi.org/10.3390/electronics11091502
2. H. M. Khalid, S. M. Muyeen, J. C.-H. Peng, Cyber-attacks in a looped energy-water nexus: An inoculated sub-observer-based approach, *IEEE Syst. J.*, **14** (2020), 2054–2065. https://doi.org/10.1109/JSYST.2019.2941759
3. M. Shashanka, M.-Y. Shen, J. Wang, User and entity behavior analytics for enterprise security, *2016 IEEE International Conference on Big Data*, 2016, 1867–1874. https://doi.org/10.1109/BigData.2016.7840805
4. D. Maher, Can artificial intelligence help in the war on cybercrime? *Comput. Fraud Secur.*, **2017** (2017), 7–9. https://doi.org/10.1016/S1361-3723(17)30069-6
5. J. Voris, Y. Song, M. B. Salem, S. Hershkop, S. Stolfo, Active authentication using file system decoys and user behavior modeling: Results of a large scale study, *Comput. Secur.*, **87** (2019), 101412. https://doi.org/10.1016/j.cose.2018.07.021
6. M. Pusara, C. E. Brodley, User re-authentication via mouse movements, Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, 2004, 1–8. https://doi.org/10.1145/1029208.1029210
7. P. Slipenchuk, A. Epishkina, Practical user and entity behavior analytics methods for fraud detection systems in online banking: A survey, *Biologically Inspired Cognitive Architectures 2019*, 2020, 83–93. https://doi.org/10.1007/978-3-030-25719-4_11
8. W. Meng, Y. Wang, D. S. Wong, S. Wen, Y. Xiang, Touchwb: Touch behavioral user authentication based on web browsing on smartphones, *J. Netw. Comput. Appl.*, **117** (2018), 1–9. https://doi.org/10.1016/j.jnca.2018.05.010
9. A. G. Martín, M. Beltrán, A. Fernández-Isabel, I. Martín de Diego, An approach to detect user behavior anomalies within identity federations, *Comput. Secur.*, **108** (2021), 102356. https://doi.org/10.1016/j.cose.2021.102356
10. D. Rumelhart, G. Hinton, R. Williams, Learning internal representations by error propagation, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Cambridge: MIT Press, 1986, 318–362.
11. I. Ortega-Fernandez, M. Sestelo, N. M. Villanueva, Explainable generalized additive neural networks with independent neural network training, *Stat. Comput.*, **34** (2024), 6. https://doi.org/10.1007/s11222-023-10320-5
12. A. Morales-Forero, S. Bassetto, Case study: A semi-supervised methodology for anomaly detection and diagnosis, *2019 IEEE International Conference on Industrial Engineering and Engineering Management*, 2019, 1031–1037. https://doi.org/10.1109/IEEM44572.2019.8978509
13. A. González-Muñiz, I. Díaz, A. A. Cuadrado, D. García-Pérez, D. Pérez, Two-step residual-error based approach for anomaly detection in engineering systems using variational autoencoders, *Comput. Electr. Eng.*, **101** (2022), 108065. https://doi.org/10.1016/j.compeleceng.2022.108065

14. N. Görnitz, One-class classification in the presence of point, collective, and contextual anomalies, PhD thesis, Technische Universität Berlin, 2019.

15. J. S. Flynn, C. Giannetti, H. Van Dijk, Anomaly detection of DC nut runner processes in engine assembly, *AI*, **4** (2023), 234–254. https://doi.org/10.3390/ai4010010

16. R. R. Mauritz, F. P. J. Nijweide, J. Goseling, M. van Keulen, A probabilistic database approach to autoencoder-based data cleaning. Available from: https://doi.org/10.48550/arXiv.2106.09764 .

17. S. Hawkins, H. He, G. Williams, R. Baxter, Outlier detection using replicator neural networks, In: *Data Warehousing and Knowledge Discovery*, Berlin, Heidelberg: Springer, 2002, 170–180. https://doi.org/10.1007/3-540-46145-0_17

18. H. Wang, M. J. Bah, M. Hammad, Progress in outlier detection techniques: A survey, *IEEE Access*, **7** (2019), 107964–108000. https://doi.org/10.1109/ACCESS.2019.2932769

19. Z. Xiao, Q. Yan, Y. Amit, Likelihood regret: An out-of-distribution detection score for variational auto-encoder, *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020, 1–12.

20. M. Ribeiro, A. E. Lazzaretti, H. S. Lopes, A study of deep convolutional auto-encoders for anomaly detection in videos, *Pattern Recognit. Lett.*, **105** (2018), 13–22. https://doi.org/10.1016/j.patrec.2017.07.016

21. M. Sakurada, T. Yairi, Anomaly detection using autoencoders with nonlinear dimensionality reduction, *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, 2014, 4–11. https://doi.org/10.1145/2689746.2689747

22. X. Wang, D. Pi, X. Zhang, H. Liu, C. Guo, Variational transformer-based anomaly detection approach for multivariate time series, *Measurement*, **191** (2022), 110791. https://doi.org/10.1016/j.measurement.2022.110791

23. C. Zhou, R. C. Paffenroth, Anomaly detection with robust deep autoencoders, *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, 665–674. https://doi.org/10.1145/3097983.3098052

24. I. Ortega-Fernandez, M. Sestelo, J. C. Burguillo, C. Piñón-Blanco, Network intrusion detection system for DDoS attacks in ICS using deep autoencoders, *Wirel. Netw.*, **30** (2024), 5059–5075. https://doi.org/10.1007/s11276-022-03214-3

25. F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain, *Psychol. Rev.*, **65** (1958), 386–408. https://doi.org/10.1037/h0042519

26. H. N. Mhaskar, T. Poggio, Function approximation by deep networks, *Commun. Pure Appl. Anal.*, **19** (2020), 4085–4095. https://doi.org/10.3934/cpaa.2020181

27. H. N. Mhaskar, T. Poggio, Deep vs. shallow networks: An approximation theory perspective, *Anal. Appl.*, **14** (2016), 829–848. https://doi.org/10.1142/S0219530516400042

28. F. Cano-Córdoba, S. Sarma, B. Subirana, Theory of intelligence with forgetting: Mathematical theorems explaining human universal forgetting using "forgetting neural networks", Technical Report CBMM Memo No. 071, Center for Brains, Minds and Machines (CBMM), MIT, 2017.

29. M. Leshno, V. Y. Lin, A. Pinkus, S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, *Neural Netw.*, **6** (1993), 861–867. https://doi.org/10.1016/S0893-6080(05)80131-5

30. P. Kidger, T. Lyons, Universal approximation with deep narrow networks, In: *Proceedings of Thirty Third Conference on Learning Theory*, PMLR, 2020, 2306–2327. Available from: `https://proceedings.mlr.press/v125/kidger20a.html`.

31. Q. Le, T. Mikolov, Distributed representations of sentences and documents, *Proceedings of the 31st International Conference on Machine Learning*, PMLR, 2014, 1188–1196. http://proceedings.mlr.press/v32/le14.html

32. T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, 2013. Available from: https://doi.org/10.48550/arXiv.1301.3781 .

33. L. van der Maaten, G. Hinton, Visualizing data using t-SNE, *J. Mach. Learn. Res.*, **9** (2008), 2579–2605.

34. B. Tian, Q. Su, J. Yu, Leveraging contaminated datasets to learn clean-data distribution with purified generative adversarial networks, *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023, 9989–9996. https://doi.org/10.1609/aaai.v37i8.26191

35. J. Glasser, B. Lindauer, Bridging the gap: A pragmatic approach to generating insider threat data, *2013 IEEE Security and Privacy Workshops*, 2013, 98–104. https://doi.org/10.1109/SPW.2013.37

36. S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, In: *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, New York: Curran Associates Inc., 2017, 4765–4774.

37. M. T. Ribeiro, S. Singh, C. Guestrin, "Why should i trust you?": Explaining the predictions of any classifier, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, 1135–1144. https://doi.org/10.1145/2939672.2939778

38. S. Wachter, B. D. Mittelstadt, C. Russell, Counterfactual explanations without opening the black box: Automated decisions and the gdpr, *Harvard J. Law Technol.*, **31** (2018), 841–887.

AIMS Press