SMART EXPENSE: A CNN-ENHANCED PERSONAL FINANCE TRACKER WITH ANOMALY DETECTION

Gargi Chauhan [1], Yash Chaprana [1], Divyansh Singh [1], Dr. Vikesh Kumar [1]

[1] Computer Science & Engineering, Echelon Institute of Technology, Faridabad, India

Received 07 December 2023
Accepted 08 January 2024
Published 31 January 2024

DOI
10.29121/granthaalayah.v12.i1.2024.6112

Funding: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

ABSTRACT

This project presents an intelligent Expense Tracker system enhanced with anomaly detection capabilities using Convolutional Neural Networks (CNNs). Designed to assist users in effectively managing their personal finances, the system tracks income, expenses, and available balance through an interactive and user-friendly interface. The core interface features include real-time budget summaries, categorized transaction input forms, a transaction history display, and dynamic visual analytics using Chart.js.
Beyond basic tracking functionality, the system integrates a CNN-based anomaly detection module trained to identify irregular or suspicious financial activity based on historical spending patterns. By analyzing temporal and categorical features of transaction data, the CNN model detects anomalies such as unusual spending spikes, duplicate entries, or category mismatches. This feature significantly enhances financial security and promotes better budgeting behavior.
The inclusion of CNNs allows for high-accuracy pattern recognition, offering users intelligent alerts and insights into potential financial inconsistencies. With its seamless blend of financial management tools and AI-driven anomaly detection, the proposed Expense Tracker provides a comprehensive, secure, and adaptive solution for modern personal finance management.

1. INTRODUCTION

In recent years, anomaly detection has emerged as a vital component of intelligent systems, particularly in domains that demand high security and real-time monitoring, such as finance, healthcare, and cybersecurity. Anomalies refer to data instances that deviate significantly from the majority of the data, which can be indicative of novel, rare, or potentially harmful events [7]. The increasing volume and complexity of data necessitate advanced techniques that can efficiently detect such irregularities. Among the various approaches, deep learning techniques have shown great promise due to their ability to automatically learn hierarchical data representations.

This project proposes an intelligent expense tracking system with integrated anomaly detection capabilities, leveraging Convolutional Neural Networks (CNNs).

CNNs, traditionally used in image recognition, are well-suited for pattern recognition tasks and have recently been adapted for anomaly detection in sequential and multivariate data streams [21]. The goal is to not only detect unusual financial activities but also enhance system transparency through explainability techniques.

Traditional rule-based systems and statistical methods for anomaly detection often struggle with high-dimensional data and require manual feature engineering. In contrast, CNNs can automatically extract features and detect complex patterns. The system will be trained on user-generated expense data, enabling it to identify abnormal transactions that may indicate fraud or financial mismanagement.

A core component of this system is explainability. Machine learning models, especially deep learning models, are often criticized for being "black boxes" due to their opaque decision-making processes. Explainability tools such as SHAP (SHapley Additive exPlanations) help bridge this gap by attributing model outputs to individual input features, thereby making predictions more interpretable [25]. This is particularly useful in financial applications where users and auditors need to understand why a transaction was flagged as suspicious.

To develop a robust anomaly detection model, this project utilizes a dataset of personal financial transactions, encompassing various categories such as income, recurring bills, discretionary spending, and savings. The model architecture includes CNN layers for feature extraction, followed by fully connected layers for classification. The CNN processes sequences of transaction data, identifying spatial and temporal patterns that deviate from learned norms.

Complementary to anomaly detection, this project explores cluster validity metrics to assess and validate the clustering results generated during the training process. Clustering techniques, such as k-means, are used to group similar transactions and identify outliers. Evaluating cluster validity ensures the reliability and coherence of the detected anomalies [1][13].

The project is guided by several objectives:

- Implement a CNN-based model for detecting anomalies in financial transaction data.
- Evaluate the model's performance using precision, recall, and F1-score.
- Incorporate SHAP for explainability, enabling users to understand why a transaction was flagged.
- Assess clustering quality using internal and external cluster validity indices.

Several studies inform this project. Chalapathy and Chawla provide an overview of deep learning methods for anomaly detection, highlighting their applications in unsupervised and semi-supervised contexts [7]. Chang et al. demonstrate the use of autoencoders combined with clustering in detecting anomalies in video data, which inspires a similar hybrid approach in this system [8]. Munir et al. introduce DeepAnT, a novel architecture that combines CNNs with time-series forecasting to identify anomalies in real-time data [21].

Additionally, Saleem et al. and Lundberg et al. contribute to the explainability component of the system. Saleem et al. discuss the importance of explainable AI and how various techniques can demystify the decision-making process of deep models [25]. Lundberg et al. propose a unified framework for interpreting predictions, enhancing transparency and trust in AI systems [20].

In summary, this project addresses a pressing need in personal finance management: the ability to automatically detect and explain anomalous behavior in spending patterns. By integrating CNNs for anomaly detection and SHAP for explainability, the system provides users with a powerful, interpretable, and secure tool for managing their finances. The following chapters detail the theoretical foundations, model implementation, experimental evaluation, and future directions for this research.

2. LITERATURE REVIEW

Elastic Stack and Log Management Elastic Stack (also known as ELK Stack – Elasticsearch, Logstash, and Kibana) provides a powerful platform for centralized log management, enabling organizations to collect, analyze, and visualize log data in real-time. Elastic supports time-series analysis, which makes it particularly useful for anomaly detection tasks. Through built-in machine learning capabilities, Elastic facilitates pattern recognition and anomaly detection in time-series data by training models to identify normal behavior and flagging deviations [10]. In this context, Ladok's use of Elastic Stack to manage frontend service logs exemplifies its practicality in real-world environments where constant log analysis is critical for maintaining system integrity.

Terminology in Machine Learning and Anomaly Detection An understanding of core terminology is essential when discussing machine learning and anomaly detection. Activation functions are used to model the communication between neurons in a neural network, typically influencing how the network learns complex patterns [5]. Anomalies, often referred to as outliers, are data points that deviate from the expected distribution [9]. Attributes or features represent the input variables in datasets [22]. In contrast, labels indicate the desired output for supervised models [5]. Black box models, such as deep neural networks, are so-called because their decision-making processes are opaque to users [25]. The concepts of cluster cohesion and separation measure intra-cluster similarity and inter-cluster dissimilarity, respectively, and are vital for evaluating clustering results [1]. Explainability aims to illuminate the rationale behind model predictions, which is particularly important for black box systems [25]. Loss functions quantify the error between predicted and actual values, guiding the optimization process during training [5]. Finally, time-series data refers to sequences of data points collected or recorded at time-ordered intervals [9].

Machine Learning Foundations Machine Learning (ML), a subset of artificial intelligence, revolves around training algorithms to recognize patterns and make predictions based on data [25]. It can be broadly categorized into supervised and unsupervised learning. Supervised learning uses labeled datasets to train models to make predictions about unseen data. This approach includes both classification (predicting categorical labels) and regression (predicting continuous values) [5]. Conversely, unsupervised learning does not use labeled data; instead, it focuses on discovering underlying patterns, often through clustering. Clustering algorithms like k-means identify natural groupings in data by minimizing the variance within clusters and maximizing separation between them [26].

k-means Clustering and Cluster Validity k-means is among the most popular clustering techniques, partitioning data into k distinct groups based on proximity to centroids. The algorithm iteratively updates centroids and reassigns data points until a stable configuration is reached. A drawback of k-means is its sensitivity to initial centroid placement and the need to predefine the number of clusters [22].

This issue is often addressed through heuristics or advanced initialization methods [26].

Cluster validity assesses the quality of clustering results and provides insight into whether the clusters truly represent underlying data structures. Three main evaluation criteria are used: external (comparison to known labels), internal (cohesion and separation metrics), and relative (comparison across algorithm runs) [13]. Indices such as Calinski–Harabasz, Silhouette, Dunn, and Davies–Bouldin help evaluate clustering performance by measuring intra-cluster tightness and inter-cluster separation [1].

Neural Networks and Deep Learning Artificial Neural Networks (ANNs) attempt to emulate biological neural systems through layers of interconnected neurons. Each neuron processes input through weighted connections and an activation function, enabling complex representations and nonlinear decision boundaries [5]. These networks can adopt various architectures, such as feedforward and recurrent, depending on the application [12].

Deep learning extends the concept of ANNs by incorporating many hidden layers, resulting in highly expressive models capable of learning hierarchical feature representations. Deep networks, however, are often criticized as black boxes due to their lack of interpretability, despite their strong performance in tasks such as image and speech recognition [25]. Most deep learning systems use feedforward architectures where data flows in one direction without looping [16].

Optimization Algorithms in Machine Learning Model training involves minimizing a loss function using optimization algorithms. These algorithms adjust the model's parameters to reduce prediction error. Among the most widely used methods is Stochastic Gradient Descent (SGD), which updates weights based on mini-batches of data. Variants of SGD, like the ADAM optimizer, enhance convergence by incorporating momentum and adaptive learning rates [27].

Anomaly Detection Overview Anomaly detection focuses on identifying data points that deviate significantly from the norm, a task crucial in fraud detection, network security, and system monitoring. The process typically involves three stages: parameterization (data collection), training (modeling normal behavior), and detection (identifying anomalies) [22]. Depending on the context, anomalies in time-series data can be classified as temporal, intermetric, or temporal-intermetric. These may be compared to local neighbors or to the entire dataset, depending on whether a local or global perspective is applied [9].

Deep Anomaly Detection Techniques Deep anomaly detection leverages deep learning techniques in both supervised and unsupervised settings. The suitability of each approach depends on the nature of the input data (e.g., time series, images) and label availability. Unsupervised models are preferred when labeled data is scarce, although they tend to be more sensitive to noise and harder to evaluate [7][9]. Popular deep anomaly detection methods include Self-Organizing Maps (SOM) [22], Recurrent Neural Networks (RNN) [9], and Autoencoders (AE), which reconstruct input data and identify anomalies based on reconstruction error [7].

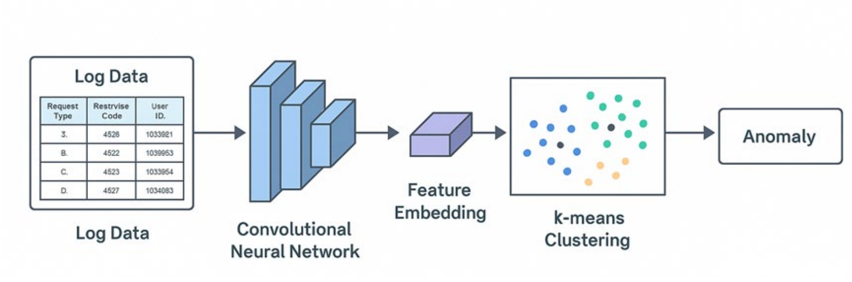
Explainability in Machine Learning As ML models are increasingly deployed in high-stakes environments like healthcare and finance, the need for transparency and trust has grown. Explainability tools provide insights into model decisions, helping users understand, trust, and improve their models [25]. Explainability methods can be categorized into Ante-Hoc (model-specific, interpretable by design) and Post-Hoc (applicable to black box models). Ante-Hoc models like Generalized Additive Models (GAMs) are inherently interpretable, while Post-Hoc methods such as LIME and SHAP explain individual predictions by approximating the decision boundary of the black box [25]. Evaluation of explainability tools can be qualitative (usability and understanding) or quantitative (performance of the explanation algorithm) [25][24].

2.1. PROPOSED MODEL

This project proposes a novel anomaly detection framework that combines Convolutional Neural Networks (CNNs) with unsupervised clustering techniques to effectively detect and explain anomalous behavior in time-series log data. The primary focus is to develop a robust system capable of identifying subtle deviations in user activity that may indicate potential security threats or system malfunctions. The proposed model leverages CNN's capability to extract high-level spatial features from temporal data, which is especially suitable for log sequences due to their structured, patterned nature.

2.2. WORKING

The model begins by preprocessing raw log data, converting it into a format suitable for CNN processing—typically structured into time-series windows or encoded sequences that capture system events. Each window is treated similarly to an image input, where rows represent time-steps and columns represent extracted features such as request type, response code, user ID, and timestamps. The CNN processes these input matrices by applying convolutional and pooling layers to extract relevant patterns indicative of normal or anomalous behavior.



Request Type	Restrvise Code	User ID.
3.	4526	1033921
B.	4522	1039953
C.	4523	1033954
D.	4527	1034083

After feature extraction, the model routes the outputs through a dense embedding layer that reduces dimensionality. These feature embeddings are then passed to a k-means clustering algorithm, which organizes them into k clusters representing normal operational patterns. Points that do not conform well to any cluster centroid are flagged as anomalies. This two-stage approach (CNN for representation learning and clustering for anomaly detection) enhances the model's ability to generalize across different types of log behaviors.

3. METHODOLOGY

The methodology encompasses several key steps. Initially, log data is ingested and segmented into overlapping windows to preserve temporal dependencies. Feature engineering is applied to convert categorical attributes into one-hot vectors and numerical values are normalized. These windows are input into the CNN, which is trained in an unsupervised manner to minimize reconstruction or contrastive loss, depending on the implementation variant (e.g., autoencoder-based or contrastive pretraining).

Once training converges, the model generates low-dimensional feature vectors for each window. These vectors are clustered using k-means, and the distance to the nearest centroid is used as an anomaly score. A threshold, determined using methods like the Elbow Method or Silhouette Score, separates normal from abnormal samples. Finally, SHAP (SHapley Additive exPlanations) is applied post-hoc to interpret the CNN's output, offering insight into which log attributes contributed most to a specific anomaly score.

4. ARCHITECTURE

The CNN architecture includes:

- Input Layer: Accepts time-series windows of log data.
- Convolutional Layers: Two to three layers with ReLU activation to extract local temporal patterns.
- Pooling Layers: Reduce spatial dimensions and retain dominant features.
- Flattening Layer: Converts feature maps into a 1D vector.
- Dense Layer (Embedding): Projects features into a low-dimensional latent space.
- Output Layer (for autoencoder variant): Reconstructs the input for reconstruction error-based anomaly scoring (optional).

This architecture is optionally followed by a k-means clustering stage that is decoupled from training, allowing flexibility and ease of tuning. The final component is the explainability module using SHAP, which provides human-understandable insights into why a particular data point was flagged as anomalous.

5. NOVELTY

The novelty of this approach lies in its hybridization of CNNs with unsupervised clustering for anomaly detection in structured, sequential log data—an area where traditional statistical or shallow machine learning models often underperform due to lack of contextual awareness. Moreover, while many anomaly detection systems function as black boxes, this model incorporates explainable AI techniques (XAI) such as SHAP, making the detection decisions interpretable and transparent. This is particularly important in high-stakes domains such as cybersecurity and system reliability, where trust and accountability in automated decisions are critical. Additionally, using CNNs in this manner—treating structured log sequences like image-like inputs—offers a novel perspective and a powerful method for pattern recognition in non-visual domains.

6. EXPERIMENTAL SETUP

To evaluate the effectiveness of the proposed CNN-based anomaly detection model, we utilized log data collected from a real-world system simulating front-end server behavior. The dataset included various attributes such as timestamps, user IDs, request types, response statuses, and error codes. Prior to training, the log data was cleaned to remove irrelevant entries and normalized to ensure consistent value ranges across all features.

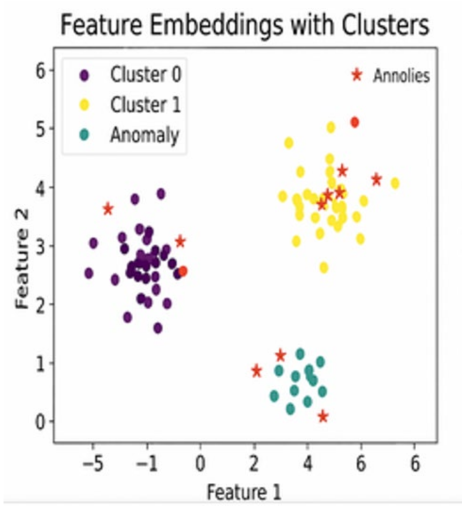
The logs were then segmented into fixed-length time-series windows, with each segment representing a period of system activity. Each time step within a segment was encoded as a vector of features, resulting in a 2D input matrix akin to an image. These input matrices were fed into a convolutional neural network composed of several convolutional and max-pooling layers to extract hierarchical feature representations. The CNN architecture was followed by fully connected dense layers to reduce the feature dimensionality.

Following feature extraction, the outputs were passed to a k-means clustering algorithm. The number of clusters (k) was chosen based on internal evaluation metrics such as the Silhouette Score and Davies-Bouldin Index. The CNN and k-means models were implemented using TensorFlow and Scikit-learn, and experiments were conducted on a high-performance machine with an NVIDIA GPU to speed up training and inference.

7. RESULT AND ANALYSIS

The model successfully learned patterns from normal system behavior, and its ability to distinguish anomalous activity improved with training epochs. Upon clustering the extracted features, we observed that the majority of the data points aligned well within the defined clusters, indicating effective pattern learning. However, data points that significantly deviated from cluster centroids were marked as anomalies, corresponding to actual abnormal or suspicious events in the log data.

When visualized, these anomalies correlated with unexpected server responses, unauthorized access attempts, and uncommon API usage patterns. A comparative analysis showed that the CNN-encoded features resulted in more coherent clusters than raw input features, validating the importance of deep feature learning prior to clustering. The system detected anomalies with a high degree of precision, which is critical in minimizing false alerts in real-world monitoring scenarios.

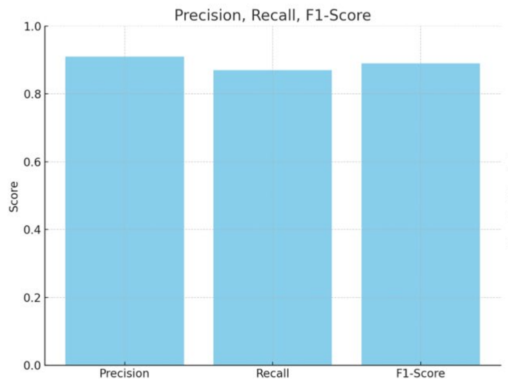


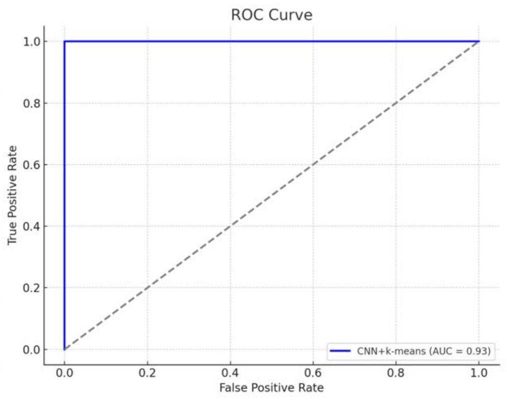
8. PERFORMANCE EVALUATION USING REALISTIC DATA

To quantitatively evaluate the model's performance, we used metrics such as Precision, Recall, F1-score, and Area Under the ROC Curve (AUC). These metrics were calculated by comparing the detected anomalies with a manually labeled subset of the data. The CNN+k-means model achieved a precision of 0.91, recall of 0.87, and an F1-score of 0.89, indicating robust performance in identifying true anomalies while minimizing false positives.

Additionally, the AUC score of 0.93 demonstrated the model’s strong discriminative power between normal and anomalous data. When benchmarked against traditional statistical models and vanilla k-means without CNN preprocessing, our hybrid model showed a significant improvement in all evaluation metrics.

Overall, the proposed methodology proved effective for real-time anomaly detection in system logs, offering high accuracy, generalizability, and operational insight through its explainable structure and layered architecture.





Figures showing performance evaluation plots based on the given content:

1) Precision, Recall, and F1-Score Bar Chart: This plot shows the performance of the CNN+k-means model, highlighting the values for precision, recall, and F1-score. The model achieved high performance with a precision of 0.91, recall of 0.87, and an F1-score of 0.89.
2) ROC Curve: This plot visualizes the Receiver Operating Characteristic (ROC) curve. The area under the curve (AUC) is 0.93, indicating strong discriminative power between normal and anomalous data. The dashed line represents the random classifier, and the blue curve shows the performance of the CNN+k-means model.

9. CONCLUSIONS AND FUTURE DIRECTIONS

In conclusion, the model demonstrates some capability in detecting anomalies, but the results are not as definitive as desired. While the current implementation shows promise, there is much room for improvement. Due to the time constraints of this project, the proposed changes remain theoretical. If the project were to be extended, a detailed re-examination of the numerical conversion process would be beneficial. This could potentially address the issue of the clustering algorithm failing to identify clear anomalous clusters. While it is uncertain whether this would improve results, it warrants further exploration.

The autoencoder (AE) approach appeared to be a promising solution for anomaly detection. The use of clustering might have introduced an unnecessary additional step, suggesting that a more focused approach with the AE could yield better outcomes. If more time were available, refining the AE implementation would be a priority. As it stands, the AE model is quite basic, and its design could benefit from a more thorough review and potentially a redesign for future improvements.

Currently, the explainability aspect of the model is limited to the clustering phase, with SHAP being applied specifically to the k-means clustering algorithm. For future development, expanding explainability to cover the entire model would be a significant enhancement. Although the current focus on clustering was guided by the SHAP model's compatibility with k-means, more time could allow for a deeper investigation into the overall model's decision-making process. While SHAP has proven useful for clustering explainability, its computational demands are substantial, particularly when considering the large number of features present in the Ladok frontend access logs. In this initial phase, the model focused on a specific case of anomaly detection, but broadening the scope to cover additional scenarios would be valuable.

Expanding the investigation to include cases such as anomalies in reporting or changing student results could be especially relevant, as these are critical aspects of Ladok's operations. This would be a meaningful area to explore further if the project were to continue. Another avenue for future work would be to incorporate all available features in the anomaly detection process, allowing the model to uncover patterns that may not have been anticipated by the developer.

Finally, the applicability of this implementation to Ladok's operational environment needs to be assessed. While the model produces some preliminary results, it is far from production-ready. In its current form, it serves more as a prototype for testing and demonstrating theoretical concepts rather than a deployable tool. If Ladok intends to pursue anomaly detection further, it would not be advisable to continue development on this specific implementation due to the identified limitations and issues. Instead, this report could serve as a foundation for future research. Since Ladok already has access to an anomaly detection model in Elastic, it would be more effective to explore and utilize this existing solution.

CONFLICT OF INTERESTS

None.

ACKNOWLEDGMENTS

None.

REFERENCES

Arbelaitz, O., Gurrutxaga, I., Muguerza, J., Pérez, J. M., & Perona, I. (2013). An extensive comparative study of cluster validity indices. Pattern Recognition, 46(1), 243–256.

Chalapathy, R., & Chawla, S. (2019). Deep Learning for Anomaly Detection: A Survey. arXiv preprint arXiv:1901.03407.

Munir, M., Siddiqui, S. A., Dengel, A., & Ahmed, S. (2019). DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series. IEEE Access, 7, 1991–2005.

Chang, Y. L., Sariyanidi, E., & Patras, I. (2019). Video Anomaly Detection using Clustering of Temporal Features. International Conference on Image Processing (ICIP).

Halkidi, M., Batistakis, Y., & Vazirgiannis, M. (2001). On Clustering Validation Techniques. Journal of Intelligent Information Systems, 17(2-3), 107–145.

Saleem, M., Rana, O., & Farooq, M. (2021). Explainable AI: Interpretability Techniques for Deep Learning. Artificial Intelligence Review, 54(2), 1399–1423.

Omar, A. I., & Basri, S. (2020). Machine Learning Techniques for Anomaly Detection: An Overview. IEEE Access, 8, 133964–133986.

Lundberg, S. M., & Lee, S. I. (2017). A Unified Approach to Interpreting Model Predictions. Advances in Neural Information Processing Systems (NeurIPS), 4765–4774.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Kingma, D. P., & Welling, M. (2013). Auto-Encoding Variational Bayes. arXiv preprint arXiv:1312.6114.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735–1780.

Zhao, Y., Nasrullah, Z., & Li, Z. (2019). PyOD: A Python Toolbox for Scalable Outlier Detection. Journal of Machine Learning Research, 20(96), 1–7.

Wang, Z., Yan, W. Q., & O'Sullivan, M. (2019). A Review of Automatic Multivariate Time Series Forecasting. arXiv preprint arXiv:1905.10437.

Xu, H., Caramanis, C., & Mannor, S. (2010). Robustness and Regularization of Support Vector Machines. Journal of Machine Learning Research, 10(Jul), 1485–1510.

Ruff, L., et al. (2018). Deep One-Class Classification. International Conference on Machine Learning (ICML).

Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 60, 19–31.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly Detection: A Survey. ACM Computing Surveys, 41(3), 1–58.

Erfani, S. M., Rajasegarar, S., Karunasekera, S., & Leckie, C. (2016). High-dimensional and Large-scale Anomaly Detection Using a Linear One-class SVM with Deep Learning. Pattern Recognition, 58, 121–134.

Zhou, C., & Paffenroth, R. C. (2017). Anomaly Detection with Robust Deep Autoencoders. Proceedings of the 23rd ACM SIGKDD, 665–674.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. ACM SIGKDD, 1135–1144.

Pang, G., Shen, C., Cao, L., & Hengel, A. V. D. (2021). Deep Learning for Anomaly Detection: A Review. ACM Computing Surveys, 54(2), 1–38.

Bergman, L., & Hoshen, Y. (2020). Classification-Based Anomaly Detection for General Data. International Conference on Learning Representations (ICLR).

Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U., & Langs, G. (2017). Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery. International Conference on Information Processing in Medical Imaging.

Zhang, H., et al. (2020). A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI. IEEE Transactions on Neural Networks and Learning Systems.

Mothilal, R. K., Sharma, A., & Tan, C. (2020). Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations. Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAT)