A Comparative Analysis of Classical and Deep Learning Anomaly Detection Methods on Cybersecurity and Server Monitoring Datasets

Joshua Streets and Husnu S. Narman
Department of Computer Science
Marshall University, Huntington, WV, USA
{streets18, narman}@marshall.edu

***Abstract*—Anomaly detection is widely used for identifying abnormal behavior in subjects such as cybersecurity and system monitoring. While deep learning approaches have recently achieved considerable attention, their practical advantages over classical anomaly detection methods remain unclear. This study compares three classical methods, Principal Component Analysis (PCA), Isolation Forest (IForest), and K-Nearest Neighbors (KNN), with two deep learning methods, an LSTM Encoder-Decoder and an LSTM Forecaster, on the Server Machine Dataset (SMD) and the BPF-extended tracking honeypot (BETH) dataset. Performance was evaluated using both oracle and deployable threshold selection strategies to distinguish between theoretical and practical anomaly detection performance. Results show that classical methods remained highly competitive with deep learning approaches, with PCA achieving the strongest average deployable performance on SMD. A substantial gap was observed between oracle and deployable results on SMD, highlighting threshold estimation as a major challenge in practical anomaly detection systems. In contrast, deployable performance on BETH was often comparable to oracle performance, indicating more effective threshold generalization. These findings suggest that model complexity alone does not guarantee improved performance and that threshold estimation may be as important as model selection in real world anomaly detection applications.**

***Index Terms*—cybersecurity, Anomaly detection, Deep learning.**

I. Introduction

Anomaly detection plays an important role in modern monitoring and cybersecurity systems by identifying observations that deviate from expected behavior. Because anomalous events are often rare and difficult to explicitly model, unsupervised anomaly detection methods are frequently used to learn normal behavior and identify deviations without required labeled anomaly data [1], [2].

Traditional anomaly detection methods such as Principal Component Analysis (PCA), Isolation Forest (IForest), and K-Nearest Neighbors (KNN) have been widely studied as approaches to anomaly and outlier detection [3]–[5]. More recently, deep learning approaches based on Long Short-Term Memory (LSTM) networks have gained popularity because of their ability to model complex temporal dependencies in time series data [2], [6]–[8]. Although deep learning methods have demonstrated strong performance in many anomaly detection applications, their advantages over classical approaches may vary depending on the dataset, evaluation strategies, and application domain [9], [10].

Another challenge in anomaly detection research is the evaluation of anomaly detection methods under realistic deployment conditions. In this study, oracle evaluation refers to threshold selection procedures that utilize testing labels, whereas deployable evaluation refers to threshold selection procedures that rely only on information available prior to deployment. While oracle performance is useful as an upper bound performance estimate, such threshold selection procedures are not deployable in real world environments where testing labels are unavailable. Consequently, practical anomaly detection performance may differ substantially from oracle performance, especially when validation and testing score distributions differ.

Recent benchmarking studies have highlighted the importance of evaluating anomaly detection methods across diverse datasets and application domains [9]–[11].Although many studies evaluate anomaly detection methods on benchmark datasets, few studies directly compare classical and deep learning methods while also examining the gap between oracle and deployable performance. Furthermore, the impact of threshold estimation on practical deployment remains underexplored relative to model architecture. Understanding these limitations is important for selecting anomaly detection methods that maintain reliable performance outside controlled experimental settings.

The *objective* of this study is to compare three classical anomaly detection methods (PCA, IForest, and KNN) with two deep learning approaches (LSTM Encoder-Decoder and LSTM Forecaster) on the Server Machine Dataset (SMD) and the BPF-extended tracking honeypot (BETH) dataset.

The *key contributions* of this work are as follows: (i) A comparative analysis of classical and deep learning anomaly detection methods across two benchmark datasets representing sever monitoring and cybersecurity applications. (ii) An assessment of oracle and deployable threshold selection strategies to evaluate the gap between theoretical and practical anomaly detection performance. (iii) An investigation of threshold estimation challenges under distribution shift and their impact on deployable anomaly detection performance.

Performance is evaluated using both oracle and deployable threshold selection strategies and measured using F_1 and $F_{0.1}$ scores. The results provide insights into the relative performance of classical and deep learning approaches, the impact of deployable threshold estimation, and the role of dataset characteristics in anomaly detection evaluation.

The remainder of this paper is organized as follows. Section II presents the background and related work. Section III describes the datasets used in this study. Section IV details the experimental methodology, including data preprocessing, model implementations, threshold selection, and evaluation metrics. Section V presents and discusses the experimental results. Section VI discusses the limitations of the study. Finally, Section VII concludes the paper and outlines directions for future work.

II. Background and Related Work

A. Anomaly Detection Methods

This study evaluates five anomaly detection methods representing several common anomaly detection techniques. PCA and the LSTM Encoder-Decoder are reconstruction based methods that identify anomalies using reconstruction error. The LSTM Forecaster is a forecasting based method that uses prediction error to detect deviations from normal behavior. KNN is a distance based method that identifies anomalies using neighbor distances, while Isolation Forest is an isolation based method that exploits the tendency of anomalies to be isolated more quickly than normal observations [3]–[5], [7], [8].

B. Classical and Deep Learning Approaches

Recent surveys have examined both classical and deep learning approaches for anomaly detection, highlighting the wide variety of techniques available for different data types and application domains [12]–[14]. While recent research has introduced increasingly deep learning architectures, classical methods remain widely studied due to their simplicity and lower computational requirements. These surveys also identify ongoing challenges related to model selection, evaluation methodology, and practical deployment. [12]

The evaluated classical methods were PCA, Isolation Forest (IForest), and KNN. PCA is a reconstruction based anomaly detection method that projects data into a lower dimensional principal component subspace. PCA based anomaly detection can identify observations that deviate from the learned normal subspace. Observations that cannot be accurately represented within the subspace produce larger reconstruction errors and are considered more likely to be anomalous [3]. IForest is an isolation based anomaly detection method that identifies anomalies through recursive random partitioning. Because anomalous observations are assumed to be rare and distinct from normal data, they are typically isolated in fewer partitioning steps, resulting in shorter average path lengths across the forest [4]. KNN is a distance based anomaly detection method that assumes normal observations occur in dense regions of the feature space while anomalies appear more isolated. Anomaly scores are derived from distances to neighboring observations, with larger distances indicating a higher likelihood of anomalous behavior [5].

The evaluated deep learning methods were an LSTM Encoder-Decoder and an LSTM Forecaster. Both methods utilize Long Short-Term Memory (LSTM) networks, which are designed to capture long range temporal dependencies in sequential data through memory cells and gating mechanisms [6]. The LSTM Encoder-Decoder is a reconstruction based approach that compresses an input sequence into a latent representation and reconstructs the original sequence. Anomalies are identified using reconstruction error, where larger errors indicate greater deviation from learned normal behavior [8]. The LSTM Forecaster is a forecasting based approach that learns to predict future observations from historical sequences. Prediction errors are then used as anomaly scores, with larger deviations between predicted and observed values indicating a higher likelihood of anomalous activity [7].

III. Datasets

A. Server Machine Dataset

The Server Machine Dataset (SMD) is a publicly available benchmark dataset for multivariate time series anomaly detection. The dataset contains telemetry data collected from 28 server machines from a large internet company. Each observation consists of 38 monitored system metrics. Data is sampled at one minute intervals and divided into training and testing sets, where only test sets contain anomalous activity. Anomalies in the test set are labeled by domain experts based on incident reports. [15]

B. BETH Dataset

The BPF-extended tracking honeypot (BETH) dataset is a cybersecurity anomaly detection dataset consisting of over eight million host telemetry events collected from 23 stand alone cloud hosted honeypots. The dataset contains both benign system activity and real attack behavior captured using an eBPF-based monitoring system. BETH was designed for anomaly detection and out of distribution analysis and provides a realistic environment for evaluating unsupervised anomaly detection methods on cybersecurity telemetry data. For benchmarking purposes, the dataset provides predefined training, validation, and testing partitions, where only the testing partition contains malicious activity, reflecting a realistic unsupervised anomaly detection setting. [16]

IV. Methodology

A. Dataset Utilization

For the SMD dataset, each server machine was processed and evaluated independently. This enables all method performances to be assessed across a wide range of operating conditions and anomaly types.

For the BETH dataset, the predefined training, validation, and testing partitions were used to ensure consistency with

the benchmark protocol. The inclusion of BETH provides an evaluation of all methods on real cybersecurity telemetry data.

Together, both SMD and BETH enable evaluation across two distinct anomaly detection domains. SMD is a challenging multivariate time series benchmark with diverse machine behaviors, while BETH serves as a large scale cybersecurity dataset for assessing method performance on host telemetry data

B. Data Preprocessing

Data preprocessing followed a streamline workflow involving three main steps: loading files, applying a Min Max scaler, and windowing. Both the SMD and BETH datasets underwent similar preprocessing procedures, with specific differences noted below.

For the SMD dataset, the individual files for each machine were serialized and stored as a pickle file to improve loading efficiency during experimentation. The BETH dataset was loaded directly, and unnecessary columns were dropped. The data was then converted to NumPy arrays to ensure a consistent format for input across both datasets. After files are properly loaded, a Min Max scaler was fit to the training data and applied to both the training and testing sets. For SMD, the training data is split into separate training and validation sets based on a 70/30 split after scaling. The BETH dataset already contains validation data, so the Min Max scaler is directly applied to the validation set. After scaling, the data is then transformed into overlapping sliding windows of size 100 for each training, testing, and validation set. Due to the large size of the BETH dataset, a stride of 5 was applied window generation to reduce training and evaluation time while maintaining temporal coverage. Additional flattened representations of these sets were created to allow compatibility with classical anomaly detection methods that require two dimensional input data. Because the experiments were conducted using window based anomaly detection, the corresponding labels were also converted into sliding windows. A window was labeled as anomalous if at least one observation within the window was labeled anomalous.

C. Anomaly Detection Implementations

The objective of this study was to compare baseline implementations of classical and deep learning anomaly detection methods under a common evaluation framework rather than to maximize performance for any individual method. Consequently, simple architectures and limited hyperparameter tuning were employed to facilitate fair comparison across methods.

All classical anomaly detection methods were implemented using the scikit-learn library. Each method was fit using the training data and anomaly scores were generated for both the validation and test sets. Modifications to the default scikit-learn implementations are noted below:

PCA: The number of principle components was selected to preserve 95% of the variance in the training data. Anomaly scores were computed using reconstruction error, calculated as the Euclidean distance between each sample and its PCA reconstruction. Larger reconstruction errors represented higher likelihood of anomalous behavior.

IForest: The model consisted of 300 isolation trees. Anomaly scores were obtained using the Isolation Forest decision function, with the sign inverted so that larger scores represented higher likelihood of anomalous behavior.

KNN: The model was implemented using $k = 20$ neighbors. Euclidean distance was used to determine nearest neighbor relationships. Anomaly scores were calculated as the distance to the twentieth nearest neighbor, with larger distances representing higher likelihood of anomalous behavior.

Both deep learning anomaly detection methods where implemented using the Tensor Flow Keras deep learning framework. Model architectures were constructed using the Keras Layers and Models APIs and trained using the Adam optimizer. An EarlyStopping callback with a patience of 10 epochs was used during training to reduce overfitting and restore the best performing model weights based on validation loss. Details for each models are noted below:

LSTM Encoder-Decoder: The model consisted of a three layer encoder and a three layer decoder. The encoder contained LSTM layers with 128, 64, and 32 units respectively, while the the decoder mirrored this. A RepeatVector layer was used to transform latent representation into a fixed length sequence for decoding. Training was performed for a maximum of 200 epochs with a batch size of 50. Anomaly scores were computed using reconstruction error, calculated as the Euclidean distance between each input window and its reconstruction output. Larger reconstruction errors represented higher likelihood of anomalous activity.

LSTM Forecaster: The model consisted of a single LSTM layer containing 64 units followed by a connected output layer with one neuron for each input feature. The model used 100 previous observations to predict the next observation in the sequence. Training was performed for a maximum of 200 epochs with a batch size of 64. Anomaly scores were computed using prediction error, calculated as the Euclidean distance between the predicted observation and the corresponding ground truth observation. Larger prediction errors represented higher likelihood of anomalous activity.

D. Threshold Selection

All anomaly detection methods produced continuous anomaly scores rather than binary anomaly labels. Therefore, a threshold selection procedure was required to convert anomaly scores into anomaly predictions. Observations with anomaly scores exceeding the selected threshold were classified as anomalous. To establish an upper bound performance reference, an oracle threshold was obtained by evaluating all unique anomaly scores in the test set and selecting the threshold that maximized the F_1 score. This procedure used test labels, so it is not deployable in practice and is reported separately.

Several validation based thresholding methods were also evaluated. Thresholds were computed using the mean and standard deviation of the validation score distribution. Specifically, thresholds of $\mu + 3\sigma$ and $\mu + 4\sigma$ were evaluated, where μ is the mean validation anomaly score and σ is the standard deviation of the validation anomaly scores. Additional thresholds were also evaluated using the Peak Over Threshold (POT) method [17]. POT models validation set score extremes using a Generalized Pareto Distribution (GPD) and estimates an anomaly threshold according to Extreme Value Theory. Unless otherwise noted, deployable performance metrics were computed using thresholds derived exclusively from validation data. Test labels were never used during deployable threshold selection.

E. Evaluation Metrics

Model performance was evaluated using precision, recall, F_1 score, and $F_{0.1}$ score. Precision measures the proportion of predicted anomalies that were correctly classified, while recall measures the proportion of actual anomalies that were successfully detected. The F_1 score provides a balanced measurement of precision and recall, and is defined as

$$F_1 = \frac{2PR}{P+R} \tag{1}$$

where P is precision and R is recall.

In addition to F_1 score, $F_{0.1}$ score was also recorded. $F_{0.1}$ score places a greater emphasis on precision than recall is defined as

$$F_\beta = \frac{(1+\beta^2)PR}{\beta^2 P + R} \tag{2}$$

where $\beta = 0.1$. This metric was included because excessive false positive rates can significantly reduce the practical use of anomaly detection models.

V. Results and Discussion

A. Oracle vs Deployable Performance

Figure 1 provides a visual comparison of average oracle and deployable F_1 performance across all evaluated methods on SMD. Oracle threshold selection consistently produced higher performance than deployable threshold selection for every method. This gap in performance highlights the difficulty of threshold estimation under realistic deployment conditions and suggests that evaluation strategies using test labels for threshold selection can substantially overestimate achievable real world performance.

Table I presents the average oracle and deployable F_1 and $F_{0.1}$ scores across all SMD server machines. Deployable scores were obtained using the best performing validation based threshold among the evaluated deployable thresholding methods.

All methods exhibited a substantial drop in performance when transitioning from oracle thresholding to deployable thresholding. For example, PCA decreased from an oracle F_1 score of 0.604 to a deployable F_1 score of 0.296, while KNN decreased from 0.513 to 0.232. As shown in Figure 1, this behavior was consistent across all evaluated methods rather than one particular model. These results demonstrate that oracle evaluation can substantially overestimate achievable real world anomaly detection performance.

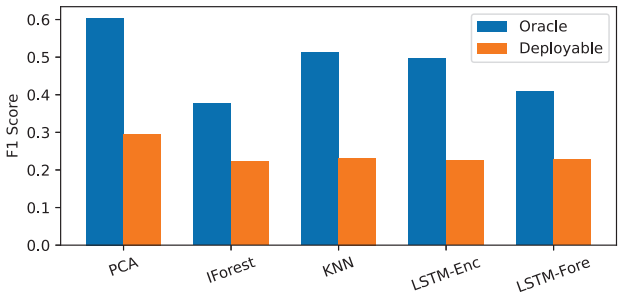


Fig. 1. Average oracle and deployable F_1 scores across SMD machines. Oracle threshold selection consistently produces substantially higher performance than deployable thresholding.

TABLE I
Average Oracle and Deployable Performance Across SMD Machines

Method	Oracle F_1	Deploy F_1	Oracle $F_{0.1}$	Deploy $F_{0.1}$
PCA	0.604	0.296	0.679	0.378
IForest	0.376	0.222	0.443	0.373
KNN	0.513	0.232	0.654	0.312
LSTM-Enc	0.496	0.226	0.619	0.295
LSTM-Fore	0.409	0.227	0.476	0.421

Oracle and deployable evaluation also resulted in different method rankings. While PCA achieved the highest performance under both evaluation settings, the performance differences between methods became much smaller under deployable evaluation. Furthermore, the LSTM Forecaster slightly exceeded the LSTM Encoder in deployable F_1 (0.227 versus 0.226), despite achieving lower oracle F_1 performance.

Although PCA achieved the highest deployable F_1 score, LSTM Forecaster achieved the highest deployable $F_{0.1}$ score. This finding demonstrates that method rankings vary depending on the evaluation metric used. While F_1 balances precision and recall equally, $F_{0.1}$ places greater weight on precision. This suggests practical method selection may depend on application rather than individual performance. In settings where false alarms are costly, methods achieving higher precision may be preferable even when their overall F_1 score is lower.

B. Classical vs Deep Learning Methods

Viewing the results in a comparative manner, the classical methods were highly competitive with the deep learning methods. Looking at F_1 scores, PCA outperformed both deep learning models, while Isolation Forest and KNN achieved performance levels similar to the LSTM based methods. These results indicate that increased model complexity did

not necessarily translate into improved anomaly detection performance.

However, the average results do not reflect performance on every machine. As shown in Table II for SMD machine 2-5, the LSTM Forecaster achieved the highest deployable F_1 score of 0.412, greatly exceeding the best classical method, PCA, which achieved a deployable F_1 score of 0.249. This demonstrates that the relative performance of anomaly detection methods may vary considerably across different server workloads and evaluation objectives.

TABLE II
ORACLE AND DEPLOYABLE PERFORMANCE FOR SMD MACHINE 2-5

Method	Oracle F_1	Deploy F_1	Oracle $F_{0.1}$	Deploy $F_{0.1}$
PCA	0.757	0.249	0.747	0.919
IForest	0.327	0.169	0.227	0.698
KNN	0.407	0.204	0.330	0.769
LSTM-Enc	0.356	0.202	0.427	0.767
LSTM-Fore	0.425	0.412	0.492	0.705

These findings suggest that method selection should be based on the characteristics and requirements of the target application rather than average benchmark performance alone. While classical methods produced stronger average results across the SMD dataset, deep learning methods were capable of achieving substantial improvements on specific machines, indicating that no single approach consistently outperformed all others across every scenario.

C. Threshold Estimation Under Distribution Shift

The large gap between oracle and deployable performance suggests that thresholds estimated from validation data did not generalize effectively to the testing data. Because validation data contained only normal observations, threshold selection relied on the assumption that validation score distributions remained representative during testing. However, this assumption was frequently wrong on SMD, resulting in thresholds that generalized poorly to the testing data.

Revisiting SMD machine 2-5, PCA had an oracle F_1 score of 0.757, significantly higher than any other evaluated method. However, under deployable thresholding, the LSTM Forecaster achieved the highest deployable F_1 score of 0.412, while PCA achieved a deployable F_1 score of only 0.249. This outcome may be attributed to differences between the validation and testing score distributions, which significantly affected threshold estimation.

Because deployable threshold estimation relied solely on the upper tail of the normal validation score distribution, even small differences between validation and testing score distributions could lead to substantial changes in deployable performance. This behavior was observed across all evaluated methods and contributed to the large gap between oracle and deployable results.

These findings suggest that threshold estimation may be as important as model selection in practical anomaly detection systems, as improvements in anomaly scoring alone may not translate into improved deployable performance.

D. Cross-Dataset Performance Comparison

Figure 2 compares deployable F_1 performance across the SMD and BETH datasets. While most methods achieved relatively modest deployable performance on SMD, nearly all methods achieved substantially higher deployable performance on BETH. This suggests that dataset characteristics strongly influence anomaly detection performance and threshold estimation.

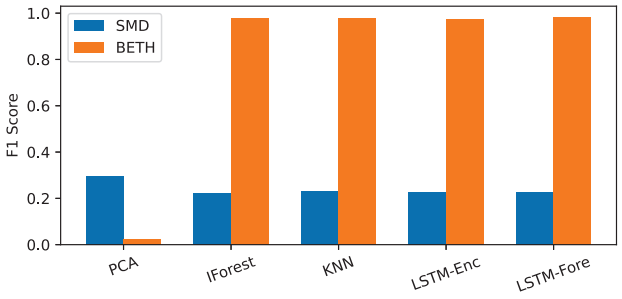


Fig. 2. Deployable F_1 performance on SMD and BETH. Most methods achieved substantially higher deployable performance on BETH, highlighting the impact of dataset characteristics on anomaly detection evaluation.

Table III presents oracle and deployable F_1 and $F_{0.1}$ scores for the BETH dataset. In contrast to SMD, the performance of nearly all evaluated methods increased substantially on BETH. Oracle and deployable performance were very similar, indicating that validation based thresholds generalized effectively to the testing data.

TABLE III
ORACLE AND DEPLOYABLE PERFORMANCE FOR BETH DATASET

Method	Oracle F_1	Deploy F_1	Oracle $F_{0.1}$	Deploy $F_{0.1}$
PCA	0.936	0.023	0.881	0.367
IForest	0.978	0.978	0.977	0.976
KNN	0.980	0.980	0.991	0.992
LSTM-Enc	0.973	0.972	0.965	0.963
LSTM-Fore	0.982	0.981	0.988	0.985

One notable exception was PCA. Although PCA achieved oracle performance comparable to the other evaluated methods, deployable performance was substantially lower. PCA achieved an oracle F_1 score of 0.936 but a deployable F_1 score of only 0.023. This large gap between oracle and deployable performance suggests that PCA was able to separate normal and anomalous observations but validation based threshold estimation failed to establish an effective deployable threshold. This finding further supports the conclusion that threshold selection can have a significant impact on practical anomaly detection performance.

The remaining classical methods, along with both deep learning methods, achieved strong oracle and deployable F_1 and $F_{0.1}$ scores. Isolation Forest and KNN performed

similarly to the LSTM based methods despite their lower computational complexity. In particular, KNN achieved a deployable F_1 score of 0.980, while LSTM Forecaster achieved 0.981. These results suggest that increased model complexity does not automatically translate into better anomaly detection performance and that classical methods can remain highly competitive when applied to suitable datasets.

VI. Limitations

A major limitation of this study is the difficulty of threshold estimation in practical anomaly detection settings. Although several deployable thresholding approaches were evaluated, substantial performance gaps remained between oracle and deployable results on SMD. As a result, model performance may be partially constrained by threshold estimation rather than anomaly scoring capability alone.

This study evaluated methods on two publicly available datasets representing industrial system monitoring and cybersecurity telemetry data. Additional datasets from other application domains may lead to different performances and should be considered in future work.

To ensure a fair comparison across methods, extensive dataset specific hyperparameter optimization was not performed. The evaluated methods should therefore be interpreted as representative baseline implementations rather than fully optimized versions of each approach. Additional tuning may improve performance of individual methods.

Only two LSTM based deep learning models were evaluated. More recent transformer based or foundation model approaches may yield different results and should be considered in future work.

VII. Conclusion

This study compared classical and deep learning anomaly detection methods using both oracle and deployable evaluation metrics across the SMD and BETH datasets. PCA, Isolation Forest, KNN, LSTM Encoder, and LSTM Forecaster were evaluated using multiple threshold selection strategies and performance metrics.

Results showed that classical methods remained highly competitive with deep learning approaches, even when relatively simple baseline implementations were employed. PCA achieved the highest average deployable F_1 performance on SMD while deep learning methods occasionally achieved substantial improvements on individual machines. These findings suggest that increased model complexity does not necessarily result in better anomaly detection performance.

A major finding of this study was the significant gap between oracle and deployable performance on SMD. Thresholds derived from validation data frequently failed to generalize to testing data, resulting in performance reductions across all methods. These results indicate that threshold estimation may be as important as model selection in practical anomaly detection systems. In contrast to SMD, deployable performance on BETH was often very close to oracle performance, suggesting that threshold estimation generalized more effectively. This highlights the importance of dataset characteristics when evaluating anomaly detection methods. Overall, the results demonstrate that practical anomaly detection performance depends not only on the quality of anomaly scores, but also on the ability to estimate effective deployable thresholds.

Future work should investigate more robust threshold estimation techniques, additional datasets and modern deep learning architectures to further improve deployable anomaly detection performance.

References

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[2] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM computing surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[3] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classifier," in *IEEE foundations and new directions of data mining workshop*, 2003, pp. 172–179.

[4] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *IEEE International Conference on Data Mining*, 2008, pp. 413–422.

[5] F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces," in *European conference on principles of data mining and knowledge discovery*. Springer, 2002, pp. 15–27.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] P. Malhotra, L. Vig, G. Shroff, P. Agarwal *et al.*, "Long short term memory networks for anomaly detection in time series," in *Esann*, vol. 89, no. 9, 2015, p. 94.

[8] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *arXiv preprint arXiv:1607.00148*, 2016.

[9] F. Rewicki, J. Denzler, and J. Niebling, "Is it worth it? comparing six deep and classical methods for unsupervised anomaly detection in time series," *Applied Sciences*, vol. 13, no. 3, p. 1778, 2023.

[10] S. Mehta, S. Mehendale, N. Fernandes, J. Sarkar, S. Sarkar, and S. Saha, "Benchmarking anomaly detection algorithms: deep learning and beyond," *arXiv preprint arXiv:2402.07281*, 2024.

[11] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao, "Adbench: Anomaly detection benchmark," *Advances in neural information processing systems*, vol. 35, pp. 32 142–32 159, 2022.

[12] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller, "A unifying review of deep and shallow anomaly detection," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 756–795, 2021.

[13] H. Huang, P. Wang, J. Pei, J. Wang, S. Alexanian, and D. Niyato, "Deep learning advancements in anomaly detection: A comprehensive survey," *IEEE Internet of Things Journal*, vol. 12, no. 21, pp. 44 318–44 342, 2025.

[14] M. Kohli and I. Chhabra, "A comprehensive survey on techniques, challenges, evaluation metrics and applications of deep learning models for anomaly detection," *Discover Applied Sciences*, vol. 7, no. 7, p. 784, 2025.

[15] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2828–2837.

[16] K. Highnam, K. Arulkumaran, Z. Hanif, and N. R. Jennings, "Beth dataset: Real cybersecurity data for unsupervised anomaly detection research," in *CEUR Workshop Proc*, vol. 3095, 2021, pp. 1–12.

[17] A. Siffer, P.-A. Fouque, A. Termier, and C. Largouet, "Anomaly detection in streams with extreme value theory," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 1067–1075.