Scientific Journal of Informatics
Vol. 12, No. No. 4, November 2025
p-ISSN 2407-7658 http://journal.unnes.ac.id/nju/index.php/sji e-ISSN 2460-0040

Smart Monitoring Implementation for Hosting Services Using Zabbix and Autoencoder Models

Rizky Fenaldo Maulana[1]*, Ardian Yusuf Wicaksono[2], Irvan Surya Nugraha[3], Yuandytha Fitria Ade Putri Sujiana[4]
[1,2,3,4] Informatics Study Program, Telkom University, Surabaya Campus, Surabaya 60231, East Java, Indonesia

Abstract.
Purpose: This study aims to develop and realistically evaluate a reliable model for identifying helpful online reviews, particularly in the context of Indonesian-language texts, which are often informal and challenging.
Methods: This study addresses several key challenges in predicting review helpfulness: the relative effectiveness of numerical features from metadata compared with traditional text representations (TF-IDF, FastText) on noisy data; the impact of severe class imbalance; and the limitations of standard validation compared with time-based validation. To address these challenges, we built an XGBoost model and evaluated various feature combinations. A hybrid approach combining SMOTE and scale_pos_weight was applied to handle class imbalance, and the best configuration was further assessed using time-based validation to better simulate real-world conditions.
Result: The results show that the model based on numerical features consistently outperformed the text-based model, achieving a peak macro F1-score of 0.7214. Compared to the IndoBERT baseline (F1-score = 0.6400) and the RCNN FastText baseline (F1-score = 0.5317), this indicates that simpler feature-driven models can provide more reliable predictions under noisy review data. Time-based validation further revealed a performance decline of up to 8.06%, confirming the presence of concept drift and highlighting that standard validation tends to yield overly optimistic estimates.
Novelty: The main contribution of this research lies in offering a robust methodology while demonstrating the superiority of metadata-based approaches in this context. By quantifying performance degradation through temporal validation, this study provides a more realistic benchmark for real-world applications and highlights the critical importance of regular model retraining.

Keywords: Autoencoder, Deep learning, Anomaly detection, Server monitoring, Zabbix
Received September 2025 / **Revised** September 2020 / **Accepted** November 2025

INTRODUCTION

The growing digital transformation across domains such as education, industry, and public administration has heightened the dependence on server infrastructures that deliver essential digital services. These infrastructures are indispensable for real-time data processing and storage, meaning that any form of disruption, ranging from gradual performance degradation to complete system failure, can have substantial consequences for service quality, data integrity, and operational continuity [1]. A key factor underlying these challenges is the lack of intelligent and proactive mechanisms that can recognize abnormal patterns at an early stage, thereby preventing their escalation into broader system-level issues [2]. Zabbix is an open-source monitoring platform widely adopted for tracking system performance metrics, such as CPU utilization, memory consumption, disk I/O, and network throughput, in real-time. It provides threshold-based alerting and visualization features that enable administration to monitor infrastructure performance effectively. However, its reliance on fixed threshold values limits its ability to detect complex anomalies, including memory leaks and botnet attacks, which often exhibit nonlinear and dynamic behaviours that cannot be adequately captured by simple rule-based mechanisms [3]. Consequently, conventional monitoring systems frequently fail to detect latent or evolving anomaly patterns, thereby increasing the risk of undetected service degradations [1].

*Corresponding author.
Email addresses: rizkyfenaldo@telkomuniversity.ac.id (Maulana), ardianyw@telkomuniversity.ac.id (Wicaksono), irvansn@student.telkomuniversity.ac.id (Nugraha), yuandythaadeputri@student.telkomuniversity.ac.id (Sujiana)
DOI: 10.15294/sji.v12i4.35851

To overcome these limitations, machine learning techniques, particularly those based on unsupervised learning, have gained increasing attention. Among them, autoencoders have demonstrated considerable effectiveness in anomaly detection tasks. The neural network architectures are designed to learn compact representations of normal input data and reconstruct them with minimal error. When anomalous inputs are encountered, the reconstruction error increases and serves as a dependable metric for detecting irregularities. This approach is particularly beneficial because it eliminates the need for labelled datasets, allowing broad applicability across diverse domains. Empirical evidence has further confirmed the studies that employ vectorized reconstruction errors to identify anomalies in cloud computing environments [1]. In the security domain, autoencoder-based methods have been widely applied for detecting anomalous patterns in network traffic and intrusion detection systems. For instance, Bårli et al. [4] used Variational Autoencoders to mitigate DoS and DDoS attacks, while Ullah and Mahmoud [5] demonstrated the effectiveness of recurrent autoencoder models in IoT networks, achieving up to 99% detection accuracy. These works confirm that autoencoders are capable of capturing complex abnormal behaviors in cybersecurity contexts, complementing their successful use in industrial systems such as SCADA fault detection [6].

Similarly, autoencoder-based techniques have been applied to detect structural anomalies in industrial container operations, showing their effectiveness in identifying irregular behaviour and improving the reliability of the operational monitoring system [7]. Similarly, autoencoder-based techniques have been applied to detect structural anomalies in industrial container operations, with their effectiveness in identifying irregular behaviour and improving the reliability of the operational monitoring system [3]. Additionally, Roelofs et al. in 2024 explored transfer learning strategies to enhance the adaptability of autoencoder-based anomaly detection models within SCADA systems, emphasizing the potential for model reuse and fine-tuning across domains [8].

Although some researchers have explored the integration of Zabbix with basic statistical alerting methods, limited efforts have been made to incorporate deep learning-based anomaly detection, particularly autoencoders, within Zabbix environments. Therefore, this study aims to develop an autoencoder-based anomaly detection model integrated with the Zabbix monitoring system, in order to enhance intelligent and adaptive monitoring capabilities for hosting services. The primary objectives of this research are to construct a hybrid system capable of detecting anomalies in real time, improving adaptability across diverse hosting environments, and minimizing reliance on static, rule-based alerting mechanisms.

LITERATURE REVIEW

Numerous studies have investigated intelligent anomaly detection methods based on machine learning and their integration with network monitoring systems. It also introduced a method of autoencoders using vectorized reconstruction errors, where each feature is evaluated separately to set its adaptive threshold. [9]. This method has been proven to improve anomaly detection accuracy and reduce false positive rates in cloud computing environments using the CIDDS-001 dataset. These studies form the foundation for deep learning-based approaches in cloud classification contexts, particularly in multi-class scenarios. Building on this, [10] integrates Random Forest, Gradient Boosting, and SVM algorithms into a microservice-based observability infrastructure to proactively detect anomalies, achieving strong predictive results ($R^2 = 0.86$ with Gradient Boosting), showing that predictive model combinations can enhance monitoring effectiveness in modern architectures [11].

Furthermore, [9] designed a real-time server monitoring system with automatic responses to failures by employing both supervised and unsupervised learning methods, such as clustering and Random Forest. Unlike prior threshold-based methods, this system demonstrates higher automation in handling dynamic anomalies [12]. To address the need for adaptability to new patterns, [11] introduced a Variational Autoencoder (VAE) as a self-learning model without manual labelling, allowing for automated network intrusion detection. Continuing this direction, [1] emphasized the importance of adaptive thresholds and hierarchical classification structures for more specific attack identification terms of implementation, by focusing on the development of Zabbix-based monitoring systems. [12] presented a large-scale deployment

in campus networks using a distributed architecture. [13] added mobility features through integration with Telegram for real-time notifications, especially valuable in environments with limited IT staff. Meanwhile, [13] demonstrated SNMP-based Zabbix capabilities in monitoring bandwidth and network interfaces in real time. Together, these show Zabbix's flexibility and efficiency as a monitoring solution.

More recent research further strengthens the effectiveness of vector reconstruction error-based autoencoders. For instance, [14] confirmed this approach's performance in cloud anomaly detection [15]. In LSTM autoencoders achieved up to 99% accuracy was achieved in industrial time-series data, while [16] enhanced autoencoder interpretability by incorporating attention mechanisms to produce visual feature-time correlations, facilitating better anomaly understanding. [17] proposed a hybrid CNN-RNN model combining LSTM, BiLSTM, and GRU for multi-class and binary attack detection in IoT networks.

From a practical integration standpoint, [18] highlighted the use of autoencoders and RNNs in systems like Zabbix and Nagios to detect complex anomaly patterns. This aligns with the semi-supervised approach in [19], which applied autoencoders to high-performance computing (HPC) environments for detecting abnormal conditions without labelled data, achieving up to 96% accuracy. Finally, [1] reported the use of Zabbix integrated with Raspberry Pi for SLA and network availability monitoring, achieving 99.9% SLA and 98.89% availability. These results highlight the robustness of Zabbix as a lightweight monitoring solution; however, the study did not involve autoencoder-based methods. This distinction emphasizes that while Zabbix is effective for infrastructure monitoring, integrating it with autoencoders—as explored in our work—provides adaptive anomaly detection beyond static SLA metrics.

Although many anomaly detection approaches have been proposed, most existing studies focus primarily on improving model accuracy without integrating real-time notifications via widely adopted communication platforms such as Telegram API. Moreover, the technical focus on reconstruction error often overlooks user experience in delivering contextual, timely alerts. Therefore, this study contributes a novel system by integrating autoencoder-based anomaly detection with Zabbix monitoring, extended with smart Telegram-based notifications, to build an adaptive, real-time, and more responsive network monitoring framework

METHODS

This study proposes an intelligent monitoring approach by integrating an autoencoder model into the Zabbix monitoring system to facilitate adaptive and real-time anomaly detection in server infrastructure metrics. Unlike traditional rule-based monitoring methods, the autoencoder model is capable of learning normal behavioural patterns from data without manual labelling, thereby enabling more accurate identification of deviations or anomalies.

The research methodology is structured into distinct stages, comprising data collection, model training, performance evaluation, and the integration of anomaly detection results into the Zabbix platform. In addition, the system is configured to deliver automated anomaly alerts through the Telegram messaging service.

Research Stages

The study commenced with the establishment of a monitoring infrastructure consisting of two virtual machines: a central monitoring machine and a target machine and a target machine for data acquisition. System metrics were collected at 15 second intervals using the Zabbix Agent over a 40-day observation period, covering ≥5 weekly cycles (weekdays/weekends) and routine monthly maintenance periods, to ensure representative workload variability. We targeted a mean time to detect (MTTD) of ≤30 s for short-lived incidents (CPU/memory spikes, bursty nginx RPS-request per second). With inference latency <0.5 second, a 15 second polling period ensures the MTTD ≤15.5 second, meeting the requirement. Operationally, Zabbix Agent overhead measured at 15 second was low and not weight in on system resources. The collected data subsequently underwent preprocessing, including normalization for comparability.

Following this stage, the autoencoder was trained exclusively on normal (non-anomalous) data to capture typical system behavior. Model performance was assessed using standard error metrics. The anomaly detection results were then integrated into the Zabbix monitoring system. Additionally, a real-time

notification mechanism was implemented using Telegram to support rapid response to detected anomalies. The overall research workflow is illustrated in Figure 1.

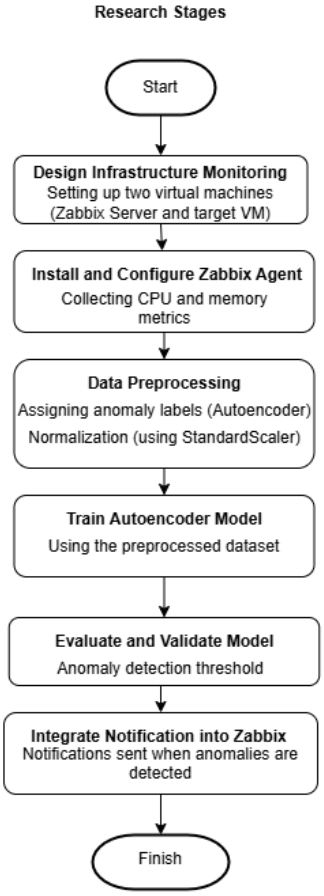

Figure 1. Research Workflow of Smart Hosting Service Monitoring Using Zabbix and Autoencoder

Data Collection and Preprocessing

The dataset was derived from a target virtual machine hosting the Nginx-powered web service. System metrics collected included CPU utilization, the five most CPU-intensive processes, memory usage, the five most memory-intensive processes, and web traffic statistics. These metrics were transmitted to the Zabbix server in CSV format, yielding 230,400 data entries over the 40-day monitoring period.

Data preprocessing was conducted in Python using the StandardScaler normalization method to maintain consistent scaling across features. The collected metrics (CPU utilization in %, memory usage in MB, nginx requests per second, etc.) differ by orders of magnitude. Directly training without rescaling would bias the autoencoder toward high-magnitude variables. We therefore applied z-score normalization (StandardScaler), which standardizes each feature to mean zero and unit variance. This ensures balanced contribution of all metrics during optimization and improves convergence stability. Anomaly labelling was performed automatically based on the reconstruction error values produced by the autoencoder. The threshold for anomaly detection was established accordingly**.** After training, each data point's reconstruction error (MAE) was computed. A data instance was labelled anomalous if its error exceeded an adaptive threshold.

Design and Evaluation of the Autoencoder Model

An autoencoder is a type of artificial neural network utilized for unsupervised learning, particularly in anomaly detection tasks, by reconstructing input data and evaluating the reconstruction error. The model comprises two principal components the encoder and the decoder (Figure 2) which are responsible for compressing and subsequently reconstructing the input data.

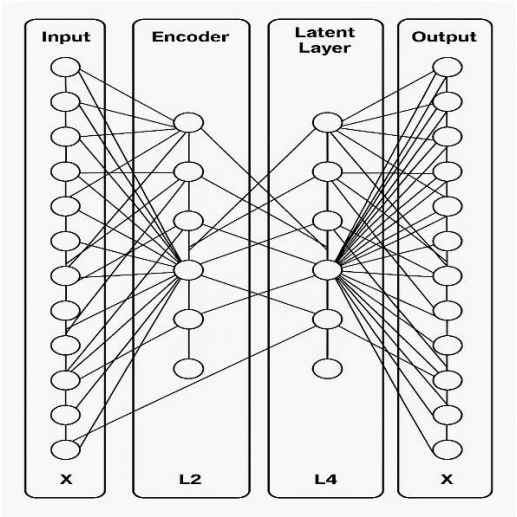

Figure 2. Autoencoder Architecture

The architecture includes three primary layers. The hidden layers utilize the Rectified Linear Unit (ReLU) activation function, while the output layer uses the Sigmoid activation function to constrain output values between 0 and 1. We selected a three-layer autoencoder with ReLU hidden activations and Sigmoid output because it provided a balance between expressive capacity and training stability. Preliminary trials with deeper networks (5–7 layers) increased complexity without appreciable gains in validation reconstruction error, while shallower networks underfit. ReLU activations prevented vanishing gradients during backpropagation. The model is trained by minimizing the Mean Absolute Error (MAE), defined as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} /y_i - \hat{y_i}/ \quad (1)$$

The training process employed the Adam optimization algorithm across 50 epochs with early stopping if the validation loss is not improved in 5 epoch and various batch size (4, 8, 16, 32, 64, 128, 256, 512), using data categorized as normal. Model evaluation was conducted by comparing reconstruction errors against a predefined threshold. Additional performance metrics Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) were also calculated:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (2)$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (3)$$

The evaluation results support both visualization and automated anomaly detection within the monitoring system.

System Architecture of the Monitoring System

The monitoring architecture consists of two connected virtual machines, as can be seen from Figure 3. Virtual Machine A serves as the main monitoring centre and contains several core components: Zabbix Server for polling data and alerting, PostgreSQL database for storing historical monitoring data, web-based visualization for system visualization, and Flask-based Detection API for running the autoencoder model.

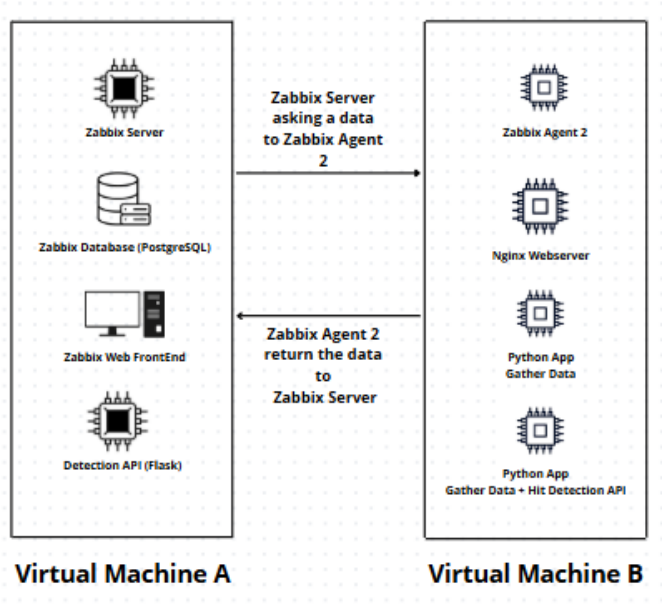

Figure 3. System Architecture Design Diagram

Virtual Machine A consists of several components that are critical to its operations, including the Zabbix Server which gathers information from agents and triggers alarms; a PostgreSQL database that maintains a history of the monitored data; the Zabbix Web Frontend that offers graphical representations of system state and alerts; and a Detection API developed using the Flask framework to run the autoencoder model for anomaly detection. In return, Virtual Machine B consists of components such as Zabbix Agent 2, which reports performance data to the Zabbix Server via TCP/10050; the Nginx Web Server, the application under monitoring; and a Python utility for gathering system data and sending it to the Detection API via HTTP/REST. In this configuration, Virtual Machine B is the monitor target where the collected metrics are processed by the Detection API, and the resulting anomaly detection data is sent back to the Zabbix Server for real-time visualization.

Integration of the Autoencoder into the Notification Framework

The autoencoder model is deployed on a Python server developed with the Flask framework. This server receives metric data from Zabbix, processes the data, and returns predictions as external items. Upon anomaly detection, Zabbix triggers a script that sends alerts via the Telegram Bot API. Notifications include contextual details such as detection time, affected metric, and reconstruction error value (Figure 4).

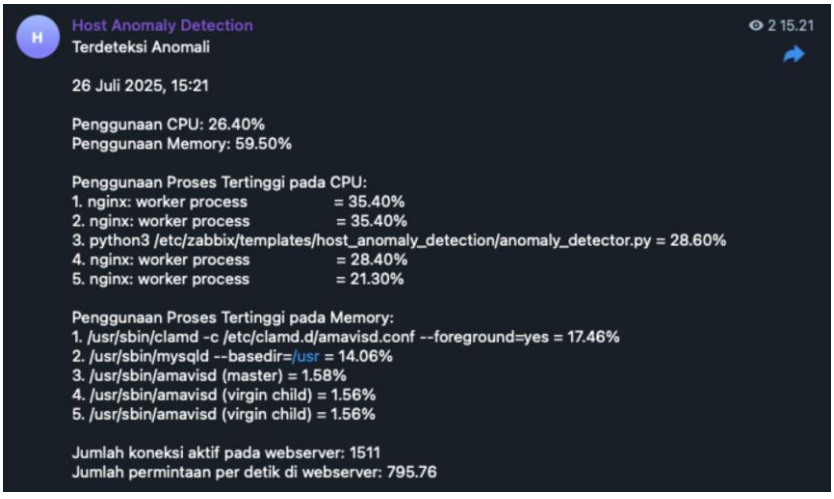

Figure 4. Example of Real-Time Telegram Notification

Once the reconstruction error exceeds the predefined adaptive threshold, Zabbix triggers a script-based notification process. This process employs the Telegram Bot API to transmit real-time alerts directly to system administrators. Each alert message contains comprehensive contextual information, including the precise timestamp of anomaly detection, the specific system metric affected, and the corresponding reconstruction error value that exceeded the threshold.
For example, a notification may state:
"[ALERT] Anomaly detected at 14:52 WIB
Metric: CPU Usage
Error Value: 0.0217 (exceeds threshold of 0.0187)
Immediate action is required."

This notification mechanism provides sufficiently detailed contextual information to support prompt and targeted responses, thereby eliminating the need for manual inspection through the monitoring dashboard. The integration of this automated alerting system significantly enhances the overall capabilities of the monitoring infrastructure, transforming it from a passive anomaly detection tool into an active and responsive solution that delivers alerts directly to the communication platform used by the operations team. This method effectively transitions the monitoring system from a rule-based architecture to an intelligent and adaptive framework capable of responding to real-time conditions within the server environment. The system generates real-time alerts that provide critical contextual information, including the timestamp on anomaly detection, CPU and memory utilization, top resource-consuming processes, the number of active server connections, and the rate incoming HTTP requests per second. All of this information is automatically compiled from system metrics that are continuously collected and processed bythe monitoring infrastructure. All of this information is automatically compiled based on metrics collected and processed by the monitoring system.

This method, particularly the integration of the autoencoder model within the Zabbix system, enables the detection of anomalies without reliance on rigid, manually predefined thresholds. Additionally, automated Telegram notifications enable system administrators to respond promptly and efficiently.

RESULT AND DISCUSSION

This chapter outlines the results alongside their explanation, such as the process of training the Autoencoder model, error reconstruction analysis, visualization of anomaly detection, and integration of the monitoring system to Zabbix. The results are presented in graph and table format to render it readable and comprehensible, while the discussion focuses on explaining their significance and contribution without duplication.

Model Training and Error Analysis

a. Batch size performance analysis

Table 1. Batch size performance analysis

Batch Size	MAE	MSE	RMSE	Epochs	Threshold
4	0,0158	0,0011	0,0332	11	0,0363
8	0,0103	0,0005	0,0217	26	0,0211
16	**0,0084**	**0,0004**	**0,0198**	13	0,0187
32	0,0117	0,0005	0,0218	10	0,0217
64	0,0149	0,0006	0,0248	9	0,0242
128	0,0107	0,0004	0,0211	30	0,0212
256	0,0124	0,0006	0,0241	20	0,0249
512	0,0153	0,0007	0,0264	17	0,0267

Table 1 illustrates the effect of varying batch size on autoencoder performance. The lowest reconstruction error was achieved at batch size 16 (MAE = 0.0084, RMSE = 0.0198), indicating that moderately small batches improve generalization. Batch sizes of 8 and 128 produced competitive results, but with either slower convergence (26 epochs for batch 8) or higher computational cost (30 epochs for batch 128). In

contrast, very small batches (4) and large batches (≥256) resulted in higher errors and inflated anomaly thresholds (up to 0.0363 for batch 4 and 0.0267 for batch 512), reducing anomaly sensitivity. These findings suggest that batch size 16 offers the optimal trade-off between training efficiency and anomaly detection sensitivity.

b. Effect of sampling interval on anomaly detection

Table 2. Batch size performance analysis

Interval (s)	Mean Lead Time	MAE	MSE	RMSE	Threshold
15	**2,26**	**0,0084**	**0,0004**	**0,0198**	0,0187
30	2,39	0,0128	0,0005	0,0228	0,0225
60	2,49	0,0151	0,0007	0,0256	0,0269

Table 2 summarizes the impact of varying the data collection interval on anomaly detection performance. The baseline dataset was collected at 15-second intervals, while the 30-second and 60-second datasets were obtained by down-sampling from this baseline. The 15-second interval delivered the best results, with the lowest reconstruction error (MAE = 0.0084, RMSE = 0.0198), the most sensitive anomaly threshold (0.0187), and the shortest mean lead time (2.26 s). Both the 15-s and 30-s intervals kept mean lead times well under the operational target of 30 s, ensuring responsiveness to short-lived anomalies, while the 60-s interval showed degraded accuracy (MAE = 0.0151, RMSE = 0.0256) and a higher threshold (0.0269). Intervals shorter than 15 s were not used, as they imposed excessive polling load on the server and negatively impacted system performance. These findings indicate that 15 s represents the optimal balance between detection timeliness, anomaly sensitivity, and system overhead.

c. Training and Validation Loss Graph

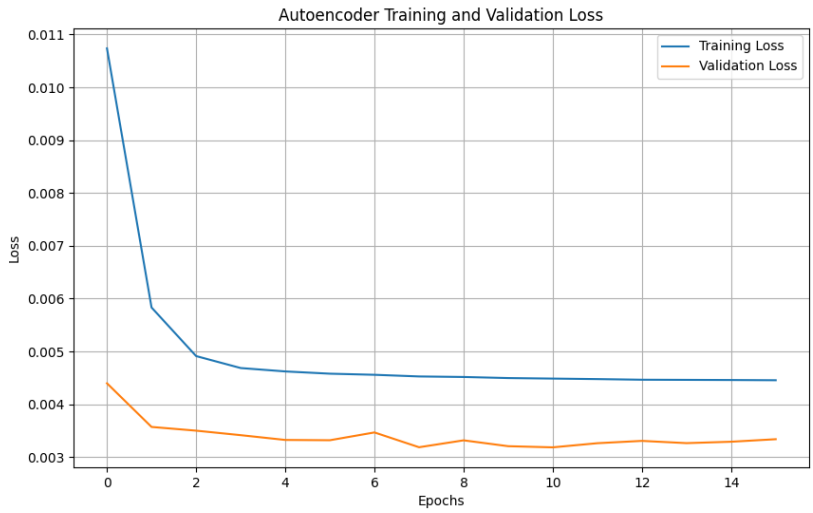

Figure 5. Training and Validation Loss Graph of the Autoencoder Model

Figure 5 shows the training and validation loss trajectories of the Autoencoder model for 18 epoch.
The training loss exhibits a rapid decline during the initial epochs, particularly within the first three, indicating effective learning of data patterns. From the fifth epoch onward, both loss values stabilize at low levels, suggesting model convergence. The absence of overfitting is indicated by the validation loss consistently aligning with or staying below the training loss, confirming the model's capability to generalize to unseen data.

d. Reconstruction Error (MAE) Distribution

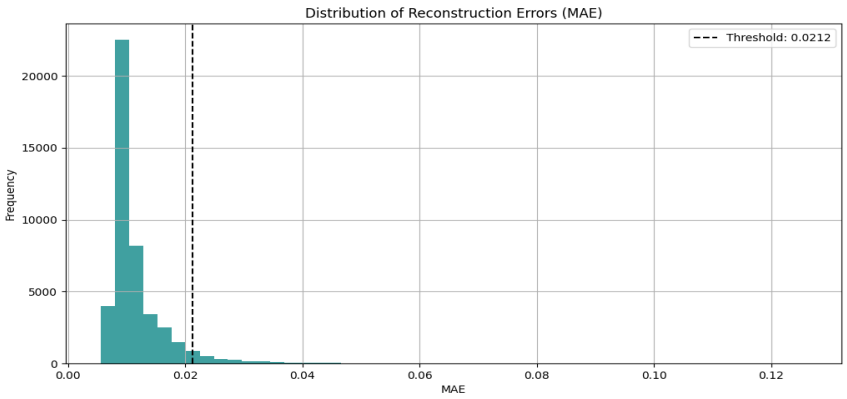

Figure 6. Reconstruction Error (MAE) Distribution and Threshold for Anomaly Labelling

Figure 6 shows the distribution of reconstruction error values (MAE) generated by the Autoencoder model during the training phase. The histogram reveals that most error values fall below 0.02, with a sharp left-skewed distribution pattern. Based on this distribution, an anomaly detection threshold was set at 0.0187, as indicated by the vertical dashed line.

This threshold corresponds to the 95th percentile of the error values and is used to distinguish between normal and anomalous data instances. Any data point with a reconstruction error exceeding this value is classified as an anomaly. The clear concentration of normal data below the threshold validates the model's capacity to learn the typical patterns of system behaviour effectively.

e. Changes in Reconstruction Error Over Time

The variation in reconstruction error values over the observation period generally exhibits a stable and low trend, consistent with the previously described error distribution. However, intermittent spikes in error values that exceed the established threshold were observed, indicating the occurrence of significant temporal anomalies. These spikes suggest instances of abnormal server activity that are transient in nature and do not follow a repetitive pattern.

Feature Characteristics and Correlation Analysis

a. Metric Feature Distribution

Feature distribution analysis reveals stable server behaviour. CPU usage and nginx_rps show left-skewed distributions, indicating minimal load under normal conditions. Memory usage from specific processes exhibits frequent peaks, representing dominant routine activities. These stable characteristics improve the Autoencoder's sensitivity to subtle anomalies.

b. Feature Correlation Matrix

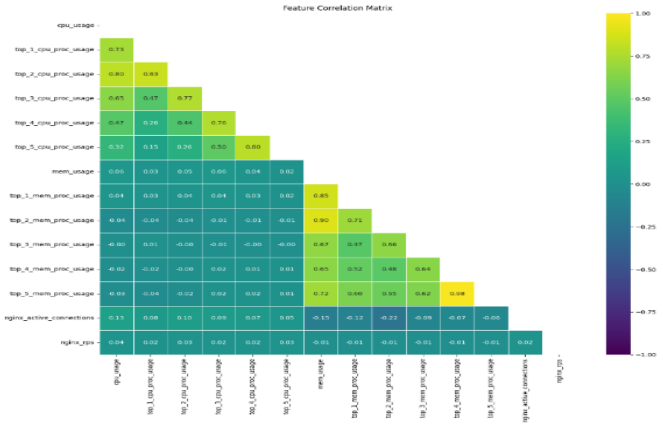

Figure 7. Feature Correlation Matrix

As shown in Figure 7, strong correlations exist among several metrics: CPU usage and the second most CPU-intensive process ($\rho = 0.83$), memory usage and the fifth most memory-intensive process ($\rho = 0.98$). These relationships enhance the model's reconstruction accuracy. Conversely, web-related metrics like nginx activity and RPS demonstrate weak correlations ($\rho \leq 0.13$), indicating their potential to signal localized anomalies.

Anomaly Detection Results Visualization

a. Anomaly Visualization in Feature Space

Visual analysis of the feature space reveals that anomalous data points are clearly distinguishable from normal data in two-dimensional scatterplots, such as the pairing of CPU usage and memory usage. Anomalies tend to appear outside the primary concentration of normal data, particularly within the medium to high value ranges. A similar distribution pattern is observed in the relationship between memory usage and nginx_active_connections, where anomalous instances deviate noticeably from the main data cluster. These observations reinforce the reliability of the Autoencoder model in detecting deviations in system behavior, both visually and quantitatively. The consistency of these visual patterns, particularly in the memory usage versus nginx_active_connections plot, further validates the model's effectiveness in identifying anomalies.

Implementation of Automatic Monitoring System

a. Monitoring Dashboard and Notifications

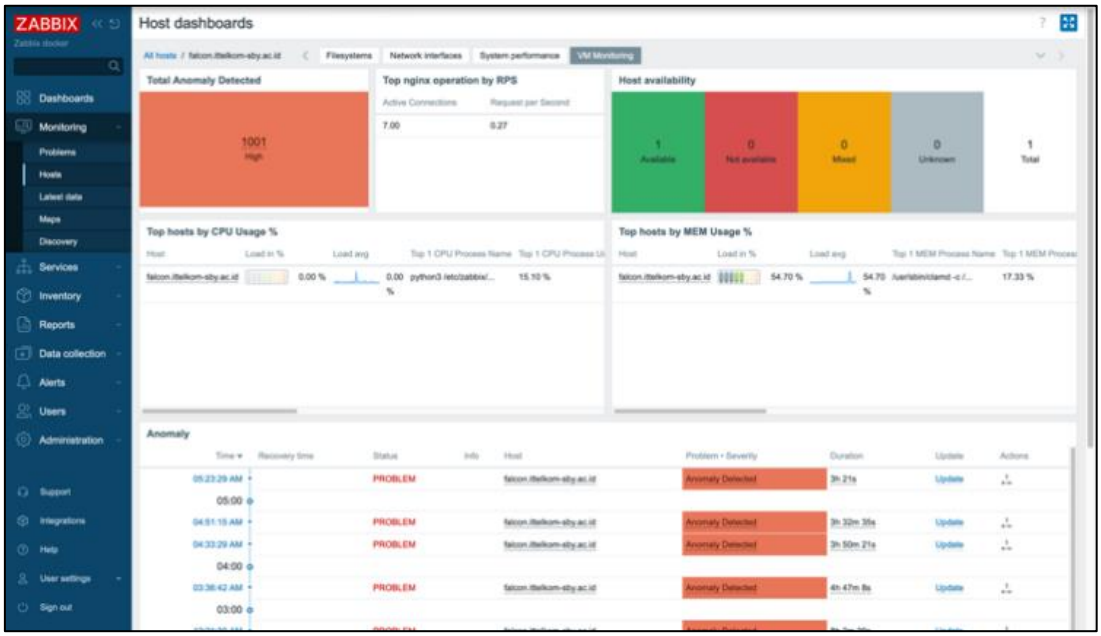

Figure 8. Zabbix Monitoring Dashboard for Anomaly Visualization

Figure 8 shows the Zabbix monitoring dashboard, which integrates real-time metrics, anomaly results, and Telegram notifications. The 40-day experimental evaluation, involving over 230,000 data entries, confirmed that the proposed system is capable of consistently detecting anomalies and generating alerts enriched with timestamps, error values, and contextual details. By embedding an Autoencoder model within the Zabbix platform, conventional monitoring is transformed into a more intelligent and adaptive mechanism. The model demonstrated strong capability in learning and generalizing normal patterns of CPU, memory, and network utilization, reflected by low training and validation losses without indications of overfitting, thereby validating its robustness. A central contribution of this study is the introduction of an adaptive threshold derived from the distribution of reconstruction errors, which enhances detection sensitivity, reduces false positives, and enables statistical differentiation between normal and anomalous behaviors. Reliability of the model was also maintained through visual inspections, whereby clustering demonstrated clean segregation between normal and abnormal data and correlation analysis revealing high dependence for memory-intensive tasks and lesser connections for network traffic, therefore pointing to

potential outliers. Additionally, the use of Telegram-based real-time notifications increased operational responsiveness by giving administrators detailed alerts including timestamps, types of metrics, and magnitudes of errors. Together, these findings provide the practical importance of combining deep learning with traditional monitoring frameworks to produce a fully autonomous pipeline from data gathering to anomaly detection and alerting dissemination. Nevertheless, the current system is restricted to a limited set of metrics; future research should explore the incorporation of additional parameters and hybrid models, such as LSTM-Autoencoders or attention-based mechanisms, to improve adaptability and temporal awareness. Ultimately, this research contributes to the advancement of infrastructure monitoring by uniting machine learning with Zabbix, offering a scalable and intelligent solution for real-time anomaly detection.

CONCLUSION

The results demonstrate that the Autoencoder model can be effectively integrated with the Zabbix monitoring framework to improve anomaly detection in hosting environments. The model successfully recognizes deviations in system behavior without relying on manually labeled data and shows robust generalization capability without signs of overfitting with MAE = 0.0084 and RMSE = 0.0198.

Anomaly detection based on reconstruction error with an adaptive threshold offers greater flexibility compared to conventional rule-based methods. Moreover, the integration with Telegram facilitates real-time notifications, thereby enhancing the timeliness of incident response.

Despite the promising results, the scope of this study remains limited to fundamental system metrics. Future research is recommended to investigate hybrid modelling approaches, broaden the spectrum of monitored parameters, and evaluate the system's effectiveness in more complex and dynamic real-world environments.

REFERENCES

[1] H. Torabi, S. L. Mirtaheri, and S. Greco, "Practical autoencoder based anomaly detection by using vector reconstruction error," *Cybersecurity*, vol. 6, no. 1, Dec. 2023, doi: 10.1186/s42400-022-00134-9.

[2] Encalada-Dávila, C. Tutivén, B. Puruncajas, and Y. Vidal, "Wind turbine multi-fault detection based on scada data via an autoencoder," *Renewable Energy and Power Quality Journal*, vol. 19, pp. 487–492, Sep. 2021, doi: 10.24084/repqj19.325.

[3] S. Jakovlev and M. Voznak, "Auto-Encoder-Enabled Anomaly Detection in Acceleration Data: Use Case Study in Container Handling Operations," *Machines*, vol. 10, no. 9, Sep. 2022, doi: 10.3390/machines10090734.

[4] E. M. Bårli, A. Yazidi, E. H. Viedma, and H. Haugerud, "DoS and DDoS mitigation using Variational Autoencoders," *Computer Networks*, vol. 199, p. 108399, Nov. 2021, doi: 10.1016/j.comnet.2021.108399.

[5] I. Ullah and Q. H. Mahmoud, "Design and Development of RNN Anomaly Detection Model for IoT Networks," *IEEE Access*, vol. 10, pp. 62722–62750, 2022, doi: 10.1109/ACCESS.2022.3176317.

[6] A. Encalada-D ´ avila, C. Tutiven´, B. Puruncajas, and Y. Vidal, "Wind Turbine Multi-Fault Detection based on SCADA Data via an AutoEncoder," *RE&PQJ*, vol. 19, no. 4, Jan. 2024, doi: 10.24084/repqj19.325.

[7] E. M. Bårli, A. Yazidi, E. H. Viedma, and H. Haugerud, "DoS and DDoS mitigation using Variational Autoencoders," *Computer Networks*, vol. 199, Nov. 2021, doi: 10.1016/j.comnet.2021.108399.

[8] C. M. A. Roelofs, C. Gück, and S. Faulstich, "Transfer learning applications for autoencoder-based anomaly detection in wind turbines," *Energy and AI*, vol. 17, Sep. 2024, doi: 10.1016/j.egyai.2024.100373.

[9] S. Oladele, D. Luis, R. Talha, and S. Racheal, "Development of an Intelligent Server Monitoring System using Machine Learning Algorithms for Real-Time Anomaly Detection and Automated Response." [Online]. Available: https://www.researchgate.net/publication/387970970

[10] D. Noetzold, A. G. D. M. Rossetto, V. R. Q. Leithardt, and H. J. de M. Costa, "Enhancing Infrastructure Observability: Machine Learning for Proactive Monitoring and Anomaly Detection," *Journal of Internet Services and Applications*, vol. 15, no. 1, pp. 508–522, Mar. 2024, doi: 10.5753/jisa.2024.4509.

[11] B. Gate, "Variational Autoencoders for Real-Time Anomaly Detection in Network Traffic: Toward Self-Learning Intrusion Detection Systems," 2025. [Online]. Available: https://www.researchgate.net/publication/391666577

[12] C. Chen and K. Li, “Fangming Guo Research on Zabbix Monitoring System for Large-scale Smart Campus Network from a Distributed Perspective,” 2024.
[13] A. Mardiyono, W. Sholihah, and F. Hakim, “Mobile-based Network Monitoring System Using Zabbix and Telegram,” in *2020 3rd International Conference on Computer and Informatics Engineering, IC2IE 2020*, Institute of Electrical and Electronics Engineers Inc., Sep. 2020, pp. 473–477. doi: 10.1109/IC2IE50715.2020.9274582.
[14] S. Erniyazov, Y.-M. Kim, M. A. Jaleel, and C. G. Lim, “Comprehensive Analysis and Improved Techniques for Anomaly Detection in Time Series Data with Autoencoder Models,” vol. 14, no. 6, 2024.
[15] D. P. Pydi and S. Advaith, “Attention boosted autoencoder for building energy anomaly detection,” *Energy and AI*, vol. 14, Oct. 2023, doi: 10.1016/j.egyai.2023.100292.
[16] I. Ullah and Q. H. Mahmoud, “Design and Development of RNN Anomaly Detection Model for IoT Networks,” *IEEE Access*, vol. 10, pp. 62722–62750, 2022, doi: 10.1109/ACCESS.2022.3176317.
[17] A. Aluwala, “AI-Driven Anomaly Detection in Network Monitoring Techniques and Tools,” *Journal of Artificial Intelligence & Cloud Computing*, pp. 1–6, Jun. 2024, doi: 10.47363/JAICC/2024(3)310.
[18] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, and L. Benini, “Anomaly Detection Using Autoencoders in High Performance Computing Systems.” [Online]. Available: www.aaai.org
[19] H. A. Damanik and M. Anggraeni, “Implementation Scheme SLA and Network Availability Mechanism for Customer Service Provider,” *Jurnal Penelitian Pos dan Informatika*, vol. 10, no. 2, pp. 125–144, Dec. 2020, doi: 10.17933/jppi.v10i2.318.