EasyChair Preprint

№ 15318

Applying Mahalanobis-Augmented Vector Reconstruction in Autoencoders and Choosing a Scaler to Improve Anomaly Detection Performance

Seung Bum Ha, Joon-Goo Shin, Yong-Min Kim and Chang Gyoon Lim

October 28, 2024

Applying Mahalanobis-Augmented Vector Reconstruction in Autoencoders and Choosing a Scaler to improve Anomaly Detection Performance

Seung Bum Ha
Dept. of Computer Engineering
Chonnam National University
Yeosu, S. Korea
aqua4lob1@naver.com

Joon-Goo Shin
Dept. of Information Security Convergence
Chonnam National University
GwangJu, S. Korea
22shinjg@jnu.ac.kr

Yong-Min Kim
Dept. of Electronic Commerce
Chonnam National University
Yeosu, S. Korea
ymkim@jnu.ac.kr

Chang Gyoon Lim *
Dept. of Computer Engineering
Chonnam National University
Yeosu, S. Korea
cglim@jnu.ac.kr

***Abstract*— This study aims to enhance the efficacy of anomaly detection techniques through the application of autoencoders. Autoencoders, neural network models that compress and reconstruct input data by learning patterns from normal instances, typically struggle with reconstructing anomalous data. To address this limitation, we propose integrating Mahalanobis Distance, a method for measuring the distance between a data point and the distribution center, into the autoencoder's latent space. Our approach diverges from conventional methods by treating reconstruction error as a vector rather than a scalar value, allowing for more granular outlier information. We evaluate the model's performance across multiple metrics, including accuracy, precision, recall, F1 score, and ROC-AUC, utilizing five different scaling techniques. Experimental results indicate that RobustScaler offers superior performance due to its resilience to outliers, ensuring consistent results across varied data distributions. This research contributes to the advancement of anomaly detection methodologies, potentially enhancing their applicability in real-world scenarios.**

Keywords— anomaly detection, autoencoders, Mahalanobis distance, reconstruction error

I. INTRODUCTION

Anomalies are unusual activity or behavior that deviates from normal behavior. Anomaly detection is the process of removing anomalies, such as unusual activity or deviant behavior, from data or a particular service. It has applications in various domains, including fraud detection, cybersecurity, and system health monitoring. Anomalies can be classified as point, contextual, or collective, with point anomalies being the most common [1].

Recently, various techniques such as clustering algorithms, Markov chains for time series data, and principal component analysis (PCA) have been used for anomaly detection [1, 2]. Methods that use outliers by establishing normal behavior and then identifying deviations using probabilistic models and adaptive thresholds have also been proposed [3]. The effectiveness of anomaly detection depends on the nature of the data, where it is important that normal patterns outnumber anomalies and that anomalies can be distinguished from normal patterns [1]. In the era of the Fourth Industrial Revolution, the importance of anomaly detection and various challenges surrounding it are expected to be proposed as data complexity increases with the emergence of the Internet of Things [3].

Autoencoders are used in many areas, but anomaly detection is one of the most popular models [4]. Borghesi et al. proposed a convolutional autoencoder (CAE) for network anomaly detection, showing improved accuracy and reduced training time compared to existing methods. After training using the autoencoder with normal system behavior on a high-performance computing system, it achieved very high accuracy compared to other models in detecting previously unseen anomalies [5]. A convolutional autoencoder designed for industrial defect detection achieved F1 scores averaging around 89% using only defect-free training data [6]. Zhou et al. introduced a novel feature encoding strategy for weakly supervised anomaly detection by exploiting the hidden representation of the autoencoder, reconstruction residual vectors, and reconstruction errors. This approach outperformed other competing methods and demonstrated that autoencoders can be efficiently used to detect anomalies in a wide range of applications, from network security to industrial inspection and high-performance computing systems [4].

In this paper, we attempt to detect outliers using the temperature of a steam trap. The aim of this research is to improve the performance of an anomaly detection method using Autoencoder. Autoencoder is a neural network model that compresses and reconstructs input data, training patterns from normal data. This model can reconstruct normal data well, but it has the characteristic of not reconstructing anomalous data well. Therefore, we use Mahalanobis Distance to detect outliers in the data by measuring the distance between the data points and the center of the distribution. We apply this distance measure to the Autoencoder's latent space to improve the performance of anomaly detection. Instead of treating reconstruction error as a single value, we treat it as a vector. Vector reconstruction error calculates the reconstruction error for each feature, providing more detailed outlier information. For anomaly detection, after compressing the data with an encoder, the Mahalanobis Distance is calculated for each feature. This

distance vector is then compared to a threshold value to determine whether each data point is an outlier.

We present a model with new features for detecting outliers in temperature data from steam traps. To improve the performance of this model, the results of anomaly detection using different scaling methods on the data are compared and analyzed. The generated the experimental results make it easy to compare the performance of each scaling method, which can be used to select the most effective anomaly detection method.

II. Related Works

A. Autoencoder

An autoencoder is a neural network model that predicts its own input. It encodes the input data in a compressed form and then reconstructs it to match the original input as closely as possible. This process helps it learn an efficient representation of the input data. It consists of an encoder, which compresses the input data into a latent space representation, and a decoder, which reconstructs the original input from this representation as represented in Fig. 1.

Like principal component analysis (PCA), autoencoders provide a mapping between the original data and the encoded representation, but they provide a non-linear transformation, which increases flexibility. Autoencoders have been applied in a variety of fields, including dimensionality reduction, classification, denoising, and anomaly detection.

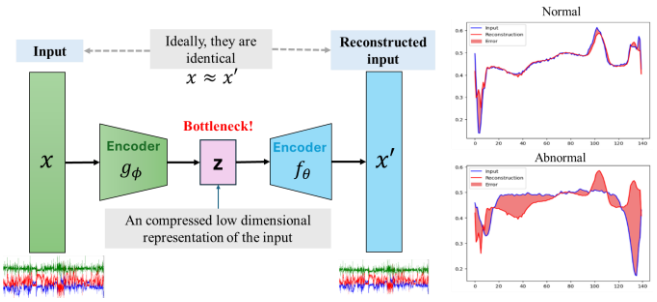


Fig. 1 Autoencoder architecture with normal and abnormal data detection [7]

An autoencoder reconstructs the input data and identifies anomalies based on the reconstructed errors. Consequently, during inference, anormal inputs typically result in higher reconstruction errors compared to normal inputs. The loss in autoencoder architecture is calculated by Eq. (1):

$$L(\theta, \phi) = \frac{1}{n}\sum_{i=1}^{n}(x^i - f_\theta(g_\phi(x^i)))^2 \quad (1)$$

where $\boldsymbol{f_\theta}$ and $\boldsymbol{g_\phi}$ represent the encoder and decoder parameters, respectively. The overall equation sums up differences between the original input **x** and the reconstructed image $\boldsymbol{f_\theta}(\boldsymbol{g_\phi}(\boldsymbol{x^i}))$. The idea is that the autoencoder learns to minimize these losses by making the reconstructed output as close as possible to the original input. This minimization process tunes the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ of the encoder and decoder, respectively.

B. Mahalanobis Distance

Mahalanobis distance (MD) measures the distance between a data point and the mean of a distribution, taking into account the covariance matrix. It is a widely used metric in statistics, machine learning, pattern recognition or classification, anomaly detection, and ecological niche modelling. Mahalanobis distance is a measure used in multivariate chemical measurements to calculate the distance between objects in both the original space and the principal component space. Unlike the Euclidean distance (ED), this method takes into account the correlation between variables and therefore can more robustly capture the relationships within the data, and is expressed as Eq (2).

$$D(x, \mu) = \sqrt{(x-\mu)\sum^{-1}(x-\mu)^T} \quad (2)$$

where x is data point (vector), μ is mean vector of a data distribution, and $\sum$ is covariance matrix of data.

C. Vector Reconstruction

Vector Reconstruction refers to the process of reconstructing the input vector in the context of the Autoencoder. This can be done by evaluating the reconstruction error for each feature to determine which feature is out of whack. If some features are difficult to reconstruct, it is possible to reflect their importance, i.e., whether they may be more important for anomaly detection. This not only allows you to detect different types of anomalies more accurately, but also makes it easier to interpret the results by clearly seeing which attributes have the largest error and by how much. The vector reconstruction process can be defined mathematically as follows:

- $\mathbf{x} = [x_1, x_2, x_3, \ldots, x_n]$: An n-dimensional vector representing each characteristic of the source data.
- $f(\mathbf{x}) = \mathbf{h}$: The encoding process compresses the input vector x into a low-dimensional latent space. In the process, it extracts important characteristics of the data.
- $g(\mathbf{h}) = \hat{\mathbf{x}}$: The decoding process restores the compressed representation **h** to the original input space. It attempts to reconstruct the original data as closely as possible.
- $\hat{\mathbf{x}} = [\hat{x}_1, \hat{x}_2, \hat{x}_3, , \ldots . \hat{x}_n]$: The reconstruction vector, which is the result of the decoding process and has the same dimensions as the original input vector. Each element $\hat{x}_i$ is a reconstruction of the corresponding characteristic x_i of the original input.
- $\mathrm{e} = \mathbf{x} - \hat{\mathbf{x}} = [e_1, e_2, e_3, \ldots, e_n]$: As reconstruction error vector, a vector representing the difference between the original input and the reconstructed vector. It calculates the error for each feature and expresses it as a vector. Unlike the traditional method of representing the error as a single scalar value, this allows you to evaluate the reconstruction performance for each attribute separately.

D. Scaling methods: Data preprocesing for performance analytics

We intend to use various scaling methods to evaluate the performance of our proposed model. It is a data preprocessing technique used to adjust the scale or distribution of features in a dataset. These methods are crucial in preparing data for machine learning algorithms, ensuring that each feature contributes equally to the performance of the model. We present the mathematical definitions of the typical scalers used in this study.

* Corresponding author

- MinMaxScaler: It scales the features to a fixed range, typically between 0 and 1. It scales each feature individually by subtracting the minimum value and dividing by the range (maximum minus minimum) of the feature. For a given vector $\mathbf{x} = [x_1, x_2, x_3, \ldots\ldots, x_n]$, the scaled value x_i' is calculated as:

$$x_i' = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \times (b - a) + a \quad (3)$$

where x_i is the original feature value, $x_{\min}$ is the minimum value of the feature, and $x_{\max}$ is the maximum value of the feature. $[a, b]$ is the desired range.

- StandardScaler: It standardizes features by removing the mean and scaling to unit variance. It centers the distribution around zero and scales the data so that the standard deviation is one. For a given vector $\mathbf{x} = [x_1, x_2, x_3, \ldots\ldots, x_n]$:

$$x_i' = \frac{x_i - \mu}{\sigma} \quad (4)$$

where x_i is the original feature value, μ is the mean of the feature, and σ is the standard deviation of the feature.

- RobustScaler: It scales features using statistics that are robust to outliers. It removes the median and scales the data according to the Interquartile Range (IQR). For a given vector $\mathbf{x} = [x_1, x_2, x_3, \ldots\ldots, x_n]$:

$$x_i' = \frac{x_i - \text{Median}(\mathbf{x})}{Q_3 - Q_1} \quad (5)$$

where x_i is the original feature value and Median($\mathbf{x}$) is the median of the feature. Q_1 is the 1st quartile (25th percentile) of the feature and Q_3 is the 3rd quartile (75th percentile) of the feature.

- Normalizer: It scales each sample (row) independently to have unit norm (i.e., the vector has a length of one). Unlike other scalers that operate on features (columns), the Normalizer operates on samples (rows). For a given vector $\mathbf{x} = [x_1, x_2, x_3, \ldots\ldots, x_n]$:

$$\mathbf{x}' = \frac{\mathbf{x}}{\|\mathbf{x}\|_2} \quad (6)$$

where $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$.

III. Mahalanobis-Augmented Vector Reconstruction of Autoencoder for Improved Anomaly Detection

In this section, we present an improvement to autoencoder-based anomaly detection using Mahalanobis Distance and applying the concept of vector reconstruction error. Autoencoder, with different scaling methods, is a neural network model that compresses and reconstructs input data, learning patterns from normal data. The model can reconstruct normal data well, but it can be characterized by poor reconstruction of anomalous data. Mahalanobis Distance measures the distance between a data point and the center of a distribution, and is effective at detecting outliers in multivariate data.

This distance measure is applied to the autoencoder's latent space to improve the performance of anomaly detection. Traditional methods treat the reconstruction error as a single value, but in this paper, we treat it as a vector. Vector reconstruction error calculates the reconstruction error for each attribute, providing more detailed outlier information. The anomaly detection uses the result of Mahalanobis Distance for each attribute after compressing the data through an encoder. The calculated distance vector is then compared to a threshold value to finally determine whether each data point is an outlier.

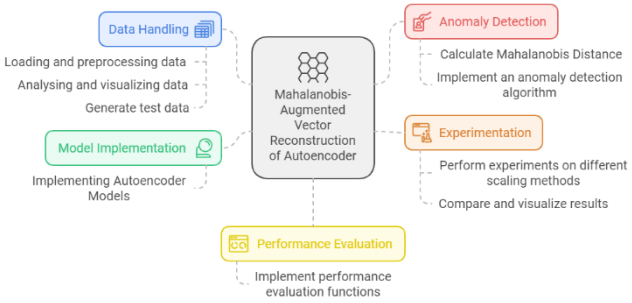


Fig. 2 Mahalanobis-Augmented vector reconstruction of autoencoder

A. Data Handling

In this study, we leveraged a HAI (HIL-based Augmented ICS) security dataset collected from a real industrial control system (ICS) testbed augmented with a hardware-in-the-loop (HIL) simulator [8]. From this dataset, we selected and applied only temperature-related data. All training datasets were assumed to be normal. For the test dataset, we applied our rules to generate normal and outlier data and performed performance evaluation based on it.

The first step involves loading the temperature data from data file. This file likely contains time-stamped temperature readings from a steam trap. We extract the 'outlet_temperature' column, which represents the temperature measurements over time. Time series data, like temperature readings, often exhibit patterns such as trends, seasonality, and cycles. Understanding these patterns is crucial for detecting anomalies.

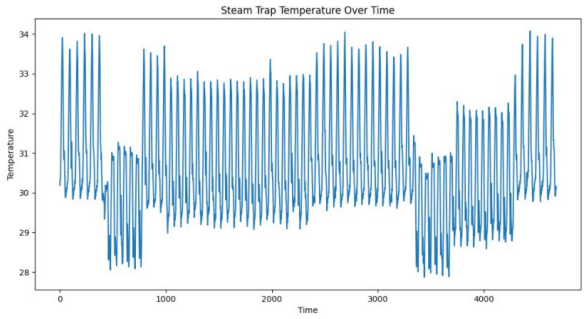


Fig. 3 Steam trap temperature over time in the training data set

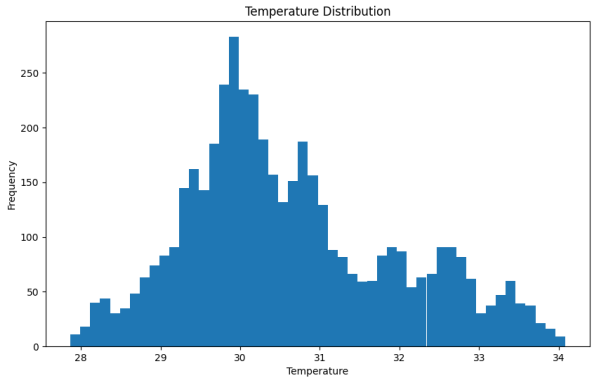


Fig. 4 Temperature distribution in the training data set

A time series plot would show temperature fluctuations over time. This graph might reveal daily cycles in steam trap operation, with higher temperatures during peak usage hours and lower temperatures during off-hours (Fig. 3). The data analysis might reveal that the steam trap operates in a temperature range of 28°C to 33°C, with a mean of 30.5°C and a standard deviation of 1.2°C (Fig. 4).

The test data for anomaly detection system is carefully constructed to evaluate the model's performance under various conditions. We generate a total of 1,500 synthetic data points, which serves as a comprehensive test set for our steam trap temperature anomaly detection model.

Most of the test data, specifically 90% or 1,350 data points, are designed to represent normal operating conditions. These normal data points are generated using a Gaussian distribution with parameters derived from our original dataset. The mean and standard deviation of the normal operating temperatures from the training data guide this generation process. This ensures that our normal test data closely mimics the typical temperature readings we would expect from a properly functioning steam trap.

The remaining 10% of the test data, amounting to 150 data points, are deliberately created as anomalies. These anomalous data points are generated to fall outside the normal operating range, which we define as two standard deviations above and below the mean temperature from our training data. To create these anomalies, we use a uniform distribution that spans from three standard deviations below the lower bound of the normal range to three standard deviations above the upper bound. This approach ensures that our anomalies are distinctly different from normal operations, yet not so extreme as to be unrealistic. Fig. 5 shows the test data, including normal and abnormal data.

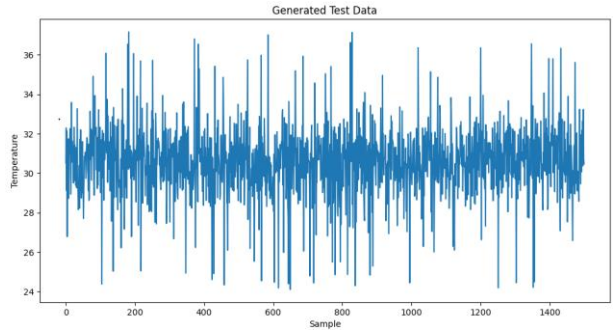


Fig. 5 Generated test data

B. *Model implementation*

We implement an autoencoder using Keras. The autoencoder consists of an encoder that compresses the input into a lower-dimensional latent space, and a decoder that reconstructs the input from this latent representation.

The implementation allows us to repeat for all scaling methods for performance evaluation with preprocessed data. The loss values during the training process for each scaling method can be plotted as shown in Fig. 6. This graph allows us to compare the starting points of each scaling method based on the initial loss values. It shows the extent to which each scaling method transformed the initial data distribution.

The slope of the graph serves as an indicator for comparing the learning rates of different methods, with a steeper slope signifying a more rapid learning process. Additionally, the final loss values after a specified number of training epochs provide insight into the effectiveness of each scaling method in enhancing model performance. The stability of the learning process for each method can be assessed by examining the fluctuations in the graph; smooth curves suggest stable training, whereas jagged curves indicate instability.

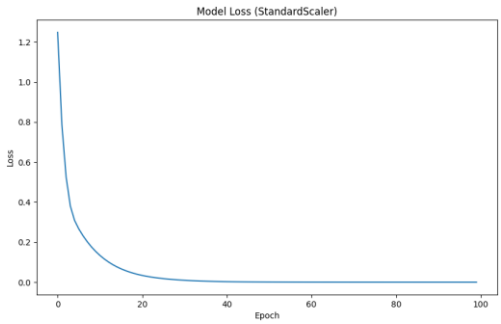


Fig. 6 Model Loss by using StandardScaler

C. *Anomaly detection*

The trained encoder is employed to transform the data into latent space. Subsequently, we compute the Mahalanobis distance for each point within this space to detect anomalies. The Mahalanobis distance quantifies the number of standard deviations a point is from the mean of a distribution, considering the covariance structure. This methodology is particularly efficacious for identifying outliers in multivariate datasets.

Using no scaling preprocessing, our anomaly detection algorithm might achieve with AUC (0.50). The ROC curve for this method would show a steep initial climb, quickly reaching a high true positive rate with a low false positive rate as shown in Fig. 7.

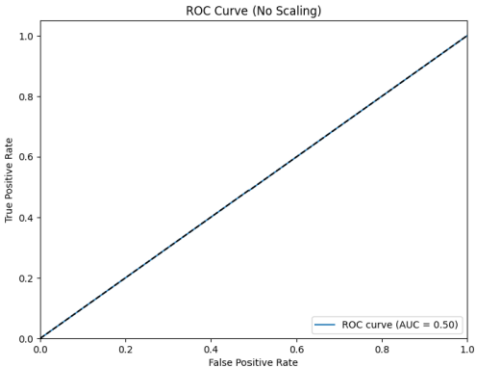


Fig. 7 ROC curve with No Scaling

D. Performance Comparison

A key part of our strategy is comparing different data scaling methods. We implement and evaluate five scaling approaches: No Scaling, MinMaxScaler, StandardScaler, RobustScaler, and Normalizer.

For each scaling method, we train a separate autoencoder model and perform the entire anomaly detection process. This allows us to assess how different scaling techniques affect the model's ability to detect anomalies. The comparison of scaling methods is crucial because it helps us understand how different ways of preprocessing the data impact the performance of our anomaly detection system.

We provide a comprehensive view of the model's performance, balancing its ability to correctly identify anomalies with its tendency to raise false alarms using the following metrics:

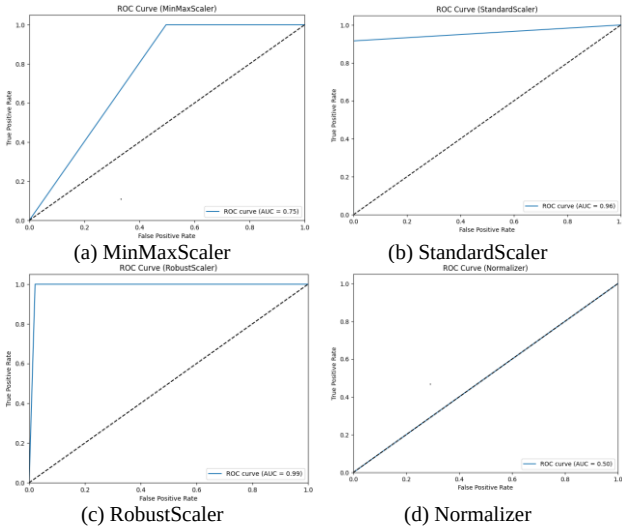
- Accuracy: Overall correctness of the model
- Precision: Proportion of true anomalies among detected anomalies
- Recall: Proportion of detected anomalies among all true anomalies
- F1-score: Harmonic mean of precision and recall
- ROC-AUC: Area under the Receiver Operating Characteristic curve
- Confusion Matrix: Detailed breakdown of true positives, false positives, true negatives, and false negatives

IV. Experimental Results and Analysis

In this section, we present the results of experiments conducted with the model proposed in this work. We repeat for the five scaling methods including no scaling proposed for our experiments. The autoencoder is trained with training data consisting of normal data.

A. Authors and Affiliations

After training with normal data, the model is evaluated using the test data presented earlier to select the best scaler. This test data includes both normal and anomalous data. The trained autoencoder is used to detect outliers on the test data for comparison. The ROC curves (Fig. 8) are draw based on these detections and various evaluation metrics are calculated.



(a) MinMaxScaler (b) StandardScaler

(c) RobustScaler (d) Normalizer

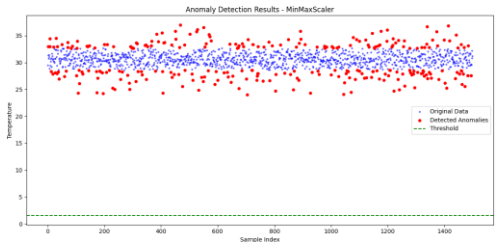
Fig. 8 ROC curves using training data based on scaler selection

Thus, both the ROC curve and the evaluation metrics such as Accuracy, Precision, Recall, F1-score, and AUC are calculated based on the test data. They are intended to evaluate the generalization performance of the model and represents the model's performance on new data that was not used for training.

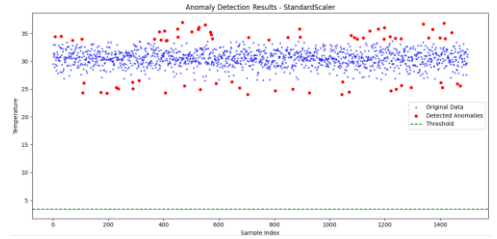
This approach can avoid overfitting, meaning that by using separate test data rather than training data. We can also check if the model is overfitting the training data. By using new data to evaluate the performance of your model, we can better predict how the model will perform in real-world situations. By using the same test data for all scaling methods, we can fairly compare the performance of each method. The scaling methods selected here should perform well in real-world applications.

B. Comparing anomal detection by scalers

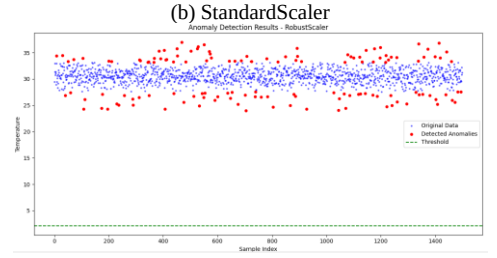
We conducted a comparative visual analysis of outlier identification across various scaling methodologies. Our experimental protocol involved applying distinct scaling techniques to both the training and test datasets. The procedure commenced with the training of an Autoencoder, followed by outlier detection on the test data. Subsequently, we generated visualizations to illustrate the results.



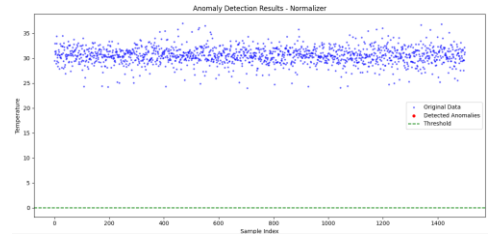
(a) MinMaxScaler



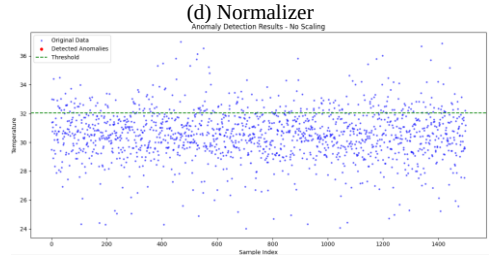
(b) StandardScaler



(c) RobustScaler



(d) Normalizer



(e) No Scaling

Fig. 9 Anomaly detection results based on scaler selection

The visual representations comprise three key elements: the original test data (represented by blue data points), the detected anomalies (indicated by red data points), and the anomaly detection threshold (demarcated by a green dashed line). This multifaceted visualization approach enables a rigorous comparative analysis of the impact of various scaling methodologies on the outlier detection process within our Autoencoder-based anomaly detection framework. By juxtaposing these elements across different scaling techniques, we can elucidate the nuanced effects of data preprocessing on the model's ability to discriminate between normal and anomalous instances in the latent space.

These graphical representations facilitate a comparative visual analysis of outlier determination across various scaling methodologies. The resultant visualizations are intended to yield the following analytical insights:

- The distribution of outliers that each scaling method detects
- The appropriateness of the thresholds
- Patterns of false positives and false negatives
- Differences in anomaly detection across scaling methods

C. Performance Metrics Comparison Across Different Scalers

The predictive outcomes exhibit varying degrees of congruence with the actual results. In the domain of machine learning and model performance evaluation, this phenomenon can be elucidated through the application of a confusion matrix. A confusion matrix represents a tabular configuration designed to facilitate the visualization of algorithmic performance, particularly in classification tasks. By juxtaposing the model's predictive output—categorized as either positive or negative—against the ground truth, we can systematically categorize the outcomes into four distinct quadrants. This classification scheme provides a comprehensive framework for assessing the model's predictive accuracy and error patterns:

- True Positive (TP): Positive prediction + Positive outcome
- False Positive (FP): Positive prediction + Negative outcome
- False Negative (FN): Negative prediction + Positive outcome
- True Negative (TN): Negative prediction + Negative outcome

TABLE I. COMPARING CONFUSION MATRIX BY SCALERS

		TP	FP
No Scaling	**FN**	1,346	0
	TN	154	0
MinMaxScaler	**FN**	1,231	115
	TN	0	154
StandardScaler	**FN**	1,346	0
	TN	81	73
RobustScaler	**FN**	**1,339**	**7**
	TN	**27**	**127**
Normalizer	**FN**	1,346	0
	TN	154	0

TABLE I presents a comparative analysis of various scaling methodologies' performance, utilizing a confusion matrix framework. The unscaled model demonstrates perfect discrimination between positive and negative classes, achieving optimal precision and recall.

The application of MinMaxScaler maintains the model's ability to correctly identify negative instances but introduces a degree of misclassification in positive cases. This suggests a diminished capacity for class differentiation when employing MinMaxScaler, as evidenced by an increase in false positive rates.

StandardScaler implementation preserves the model's accuracy in positive case identification. However, it exhibits a tendency to misclassify certain negative instances as positive, indicating a shift in the decision boundary that favors positive predictions.

Interestingly, the Normalizer scaling method yields results identical to the unscaled model, suggesting that this normalization technique does not significantly alter the model's discriminative capabilities in this context.

The RobustScaler demonstrates a marginal decline in overall performance, manifesting as slight decreases in both true positive and true negative rates. This implies that the RobustScaler somewhat compromises the model's ability to accurately classify both positive and negative instances, albeit to a lesser extent than some other scaling methods.

These findings underscore the critical role of scaling method selection in model performance, particularly in the context of binary classification tasks. The varied impacts of different scaling techniques on classification outcomes highlight the importance of careful consideration and empirical validation when choosing preprocessing methods for machine learning models.

TABLE II. PERFORMANCE METRICS COMPARISON ACROSS DIFFERENT SCALERS

	Acc.	Pre.	Recall	F1-score	ROC Curve
No Scaling	0.89	0.0	0.0	0.0	0.5
MinMaxScaler	0.92	0.57	1.00	0.3	0.75
StandardScaler	0.95	1.00	0.47	0.64	0.96
RobustScaler	**0.98**	**0.95**	**0.82**	**0.8**	**0.99**
Normalizer	0.89	0.89	0.0	0.0	0.50

As shown in TABLE II, RobustScaler performed the best across all performance metrics, with the highest ROC-AUC of 0.99 and overall superior performance. StandardScaler shows best on Precision, but has a lower Recall and therefore a lower F1 Score. MinMaxScaler has the highest value for Recall at 1.0, but its F1 Score is somewhat lower due to its lower Precision. No Scaling and Normalizer have relatively low performance, making them less efficient than the other scalers.

No Scaling scored very poorly on all performance metrics, showing that not scaling can lead to poor performance because the model does not learn the distribution of the data well. In particular, both Precision and Recall are zero, suggesting that the model made no predictions at all for that particular class.

When using MinMaxScaler Accuracy (0.923) and Recall (1.0) are high, but Precision (0.572) is low. This indicates that the model overpredicts the positive class, resulting in a high accuracy for true positives, but also a high number of false positives. This can be advantageous in situations where Recall is important like a diagnosing a disease, but the low Precision can lead to many false alerts.

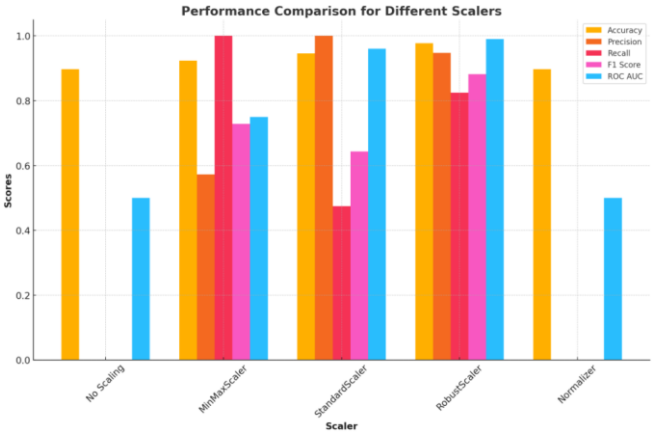


Fig. 10 Performance metrics comparison across different scalers

Fig. 10 provides an at-a-glance comparison of the performance of Accuracy, Precision, Recall, F1 Score, and ROC-AUC values for each scaler. The x-axis represents the scaler used and the y-axis represents the value of each performance metric, with values ranging from 0 to 1.

StandardScaler scores the highest in Precision (1.0) but has a low Recall (0.474). This means that the model is very strict when predicting the positive class, predicting only true positives, but may miss many positives. This can be advantageous in situations where precision is important for spam filtering.

Normalizer performs similarly to No Scaling, with lower overall performance. The Normalizer converts the length of each sample to 1, which makes the data have a relatively uniform size, but it can lose information about the distribution or relationships in the data. It may not be suitable if your data needs to retain its original distribution.

RobustScaler performs well across all performance metrics, with particularly high scores in Accuracy (0.977), F1 Score (0.882), and ROC-AUC (0.99). It is robust to outliers in the data, providing stable performance across a wide range of data distributions. It performs well on general classification problems and is efficient even on data with outliers.

V. CONCLUSON

In this paper, we aimed to enhance the performance of anomaly detection methods using autoencoders. By applying the Mahalanobis Distance to the latent space of the autoencoder, we improved the effectiveness of anomaly detection in real-world applications. Unlike existing methods that treat reconstruction error as a single value, we considered it as a vector by calculating the reconstruction error per feature, providing more detailed outlier information. Our experimental evaluation demonstrated that data scaling significantly affects the results of anomaly detection. Among the five scalers tested, RobustScaler emerged as the most advantageous choice due to its resistance to the influence of outliers, leading to stable performance not distorted by extreme values. The proposed model showed improved performance in accuracy, precision, recall, F1 score, and ROC-AUC metrics. These findings suggest that our approach can effectively enhance anomaly detection methods, making them more practical for real-world applications.

ACKNOWLEDGMENT

This research was supported by “Regional Innovation Strategy (RIS)” through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (MOE) (2021RIS-002) and Institute for Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government (MSIT)(No.IITP-RS-2022-II221203, Regional strategic Industry convergence security core talent training business).

REFERENCES

[1] Sanketh Harnoorkar, "A Study of Anomaly Detection Techniques," International Journal for Research in Applied Science and Engineering Technology, Vol. 8, pp. 960-962, 2020.

[2] A. S. Deepthi and K.Venkata Rao, “Anomaly Detection Using Principal Component Analysis,” IJCST Vol. 5, Issue 4, pp. 124-126, 2014.

[3] T. Dunning and E.Friedman, “Practical Machine Learning: A New Look at Anomaly Detection,” O’Reilly, 2014.

[4] Z. Chen, C. Yeo and B. Lee and C. Lau, “Autoencoder-based network anomaly detection,” 2018 Wireless Telecommunications Symposium (WTS), pp. 1-5, 2018.

[5] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano and Luca Benini, “Anomaly Detection using Autoencoders in High Performance Computing Systems,” arXiv, vol. 1811.05269, 2018.

[6] M. S. Minhas and J. S. Zelek, “Semi-supervised Anomaly Detection using AutoEncoders,” ArXiv, Vol. 2001.03674, 2020.

[7] S. Erniyazov, Y. Kim, and C. G. Lim, “Comparative Study on Anomaly Detection Performance Using Autoencoder and LSTM Autoencoder in Temperature and Sound Time Series Data,” International Conference of Next Generation ConvergenceTechnology 2024 (ICNGCT 2024), 106-111, 2024.

[8] Shin, Hyeok-Ki, Woomyo Lee, Jeong-Han Yun, and HyoungChun Kim. "{HAI} 1.0:{HIL-based} Augmented {ICS} Security Dataset." In 13Th USENIX workshop on cyber security experimentation and test (CSET 20), 2020.