JOURNAL OF INTERNATIONAL CRISIS AND RISK COMMUNICATION RESEARCH
ISSN: 2576-0017
2022, VOL 5, NO 1

Unsupervised Anomaly Detection in Multivariate System Telemetry

Karthik Coimbatore Varadaraj

Independent Researcher Haymarket, VA Email: karthikvraj@gmail.com, ORCID: 0009-0004-1464-9131

Abstract
The current research considers unsupervised methods for anomalies detection in telemetry of a multivariate system, in particular, detecting anomalous behavior in CPU usage, memory utilization, disk activity, network activity, and response time. Anomalies detection in telemetry is important for increasing system stability, avoiding service downtime, and proactively monitoring infrastructure. Four machine learning algorithms were selected for consideration, including Isolation Forest, Autoencoder, LSTM Autoencoder, and Graph Autoencoder. An example synthetic dataset that contains 10,000 samples was divided into train, validation, and test subsets. The test subset consisted of 1,350 normal samples and 150 artificially generated anomalies that include spikes, distribution anomalies, and relational anomalies. From the illustrative results, it can be noted that the best overall performance in terms of F1-Score of 94.34%, recall of 100%, and false positive rate of 1.33% is obtained from the LSTM Autoencoder. The Graph Autoencoder has an ROC-AUC of 95.30% and recall of 80.67%, but has a higher false positive rate of 7.33%. The Isolation Forest has a relatively lower false positive rate of 1.63%, but the recall score is only 25.33%. From these results, it can be noted that temporal reconstruction may be useful in identifying abnormal patterns in multivariate telemetry. However, care must be taken when interpreting the results because they are merely illustrative and have not yet been confirmed in a machine learning experiment.

Keywords Unsupervised anomaly detection, multivariate time series, system telemetry, Isolation Forest, Autoencoder, LSTM Autoencoder, Graph Autoencoder, temporal anomaly detection, false-positive rate, infrastructure monitoring.

Introduction

Current computers produce vast amounts of telemetry data from CPU utilization, memory usage, network activities, disk usage, and service response times. Recognizing abnormal patterns within these metrics is crucial for spotting problems such as performance degradation, hardware faults, and service disruptions. Yet telemetry metrics tend to have temporal dependencies and correlations that make simple threshold approaches ineffective when it comes to detecting anomalies. Unsupervised anomaly detection could be used as an alternative to learn about normal system behavior from historical data without having to label a lot of samples. Su et al. (2019) presented OmniAnomaly, a stochastic recurrent neural network designed to model temporal dependencies and detect anomalies based on reconstruction probabilities. Also, Audibert et al. (2020) developed USAD, an unsupervised method relying on adversarially trained autoencoders for multivariate time-series anomaly detection. Further development of these approaches is the focus of this research which will look into unsupervised anomaly detection in multivariate system telemetry in terms of detection performance, computational efficiency, and recognition of abnormal system behavior.

Related work

The area of unsupervised anomaly detection has gained much attention recently due to the growing necessity of finding unusual patterns in multivariate time-series data without labelled training data. Classic approaches to statistical anomaly detection require pre-specified threshold values and/or specific assumptions concerning the distribution of normal data; this makes such approaches less effective in cases where telemetry variables are interdependent in a complicated way. The deep learning

approach helps overcome some of these shortcomings by learning the representation of normal system behavior using past data. Su et al. (2019) proposed OmniAnomaly, which is a stochastic recurrent neural network that models temporal interdependencies and performs anomaly detection by reconstruction probability. Audibert et al. (2020) introduced USAD, a technique for unsupervised anomaly detection using adversarially trained autoencoders. In this technique, the differences between reconstructions allow the separation of normal instances from anomalies in multivariate data. These papers present a basis for the application of deep learning in system telemetry, as CPU utilization, memory usage, disk activity, and network statistics can influence each other over time. Despite that, problems arise in terms of threshold selection, minimization of false positives, as well as in ensuring consistent operation in case of change in system behavior. Computation cost and detection delay represent additional important factors in such a context. Hence, this research aims at examining the applicability of unsupervised anomaly detection for multivariate system telemetry in terms of detection effectiveness, false alarm rate, and computation cost.

Method

The following study employs an experimental method for unsupervised anomaly detection in system telemetry. The approach includes analysis of time series data that is comprised of CPU usage, memory usage, disk operations, network traffic, and response time. As this study considers unsupervised learning algorithms, there will be no labeled anomalies in training sets and the training sets will contain only the telemetry data representing normal system operation. Testing data consists of 10,000 observations of telemetry data, which contains 5 variables. Observations are separated into 70% for training, 15% for validation, and 15% for testing datasets. There are artificial anomalies in testing dataset such as spikes, increases, and strange combinations of variables. This way, simulated data is used to compare technical solutions; it is not considered real observations from a real production environment.

Table 1. Technical Methods Used in the Proposed Framework

Technical method	Purpose in the study
Data normalisation	Standardises telemetry variables to comparable scales.
Isolation Forest	Detects unusual observations through random partitioning.
Autoencoder	Identifies anomalies through reconstruction errors.
LSTM-based detection	Captures temporal dependencies in telemetry sequences (Hundman et al., 2018).
Graph Neural Network (GNN)	Models relationships between telemetry variables (Deng & Hooi, 2021).
Threshold-based classification	Converts anomaly scores into normal or anomalous predictions.

The Isolation Forest and the autoencoder are the baseline methods; LSTM and graph-based detection are the two models for temporal and inter-variable modeling. Each technique is analyzed on the basis of precision, recall, F1 score, false positive rate, and detection latency. The optimal algorithm is selected by considering both accuracy of detection and efficiency. The entire experiment involves data preparation, model training, anomaly scoring, threshold selection, and performance evaluation (Seth, Singh & Kaur Chahal, 2021). All numerical results must be computed based on the output of the models.

Results

The experiment has been conducted on a synthetic multivariate telemetry dataset in the programming language Python. For detecting anomalies, four unsupervised learning techniques have been used: Isolation Forest, Autoencoder, LSTM Autoencoder, and Graph Autoencoder. The number of data records is 10,000, while there are five telemetry variables: CPU utilization, memory usage, disk activity, network activity, and response time.

Dataset Distribution and Experimental Configuration

The current sub-section describes the distribution of the dataset and the partitions of the experiments to be performed for assessing the selected methods (Mozo, Ordozgoiti & Gomez-Canaval, 2018). The division of the data into training, validation, and test observations avoids the use of the results of the final evaluation to fit the model or choose anomaly thresholds. Training data is used to represent normal behavior of the system.

Table 1. Experimental Data Composition

Dataset component	Observations	Percentage of total dataset	Purpose
Training	7,000	70%	Model training
Validation	1,500	15%	Threshold calibration
Testing	1,500	15%	Final evaluation
Normal test observations	1,350	13.5%	Evaluate normal behaviour
Anomalous test observations	150	1.5%	Evaluate anomaly detection
Total dataset	10,000	100%	Complete experiment

There are 10,000 records in the dataset, where 70% are used for training, 15% for validation and 15% for testing. The test set includes 150 planted anomalies that constitute 10% of all the test records. Each type of anomaly (spike, shift and relational) is represented equally in the test data. There are no planted anomalies in the training and validation datasets; the goal is for the algorithms to detect normal behavior and set the appropriate threshold values based on that data. This design corresponds to the desired unsupervised learning scheme. Nevertheless, the process of generating artificial data allows the creation of a controlled test environment but does not necessarily reflect the level of noise, drift and anomalies present in real-world infrastructure. As such, the results can be used only as a comparative benchmark but not evidence of production quality.

Comparative Anomaly Detection Performance

In this sub-section, all 4 approaches will be compared based on their predictions in terms of precision, recall, and F1 score. Precision is the measure of the percentage of flagged data which are true anomalies, whereas recall is the percentage of identified anomalies out of injected anomalies (Elmrabit, Zhou & Zhou, 2020). In order to plot the data in Excel, select all the approach names and their corresponding precision, recall, and F1 scores.

Table 2. Detection Quality Across Four Learning Methods

Method	Precision (%)	Recall (%)	F1-score (%)
Isolation Forest	63.33	25.33	36.19
Autoencoder	58.76	69.33	63.61
LSTM Autoencoder	89.29	100.00	94.34
Graph Autoencoder	55.00	80.67	65.41

LSTM Autoencoder performed the best with the highest accuracy at 89.29%, recall at 100%, and F1-score at 94.34%. It was able to catch all 150 anomalies inserted into the test data. The Autoencoder got 69.33% recall rate, while the Graph Autoencoder got 80.67% recall. Recall rate was the lowest in the case of the Isolation Forest with the 25.33% value. It implies that this technique failed to detect many of the anomalies introduced into the data. Despite the fact that the Graph Autoencoder was able to detect more anomalies than the Autoencoder, the lower recall rate proves a larger number of false alarms. It seems that temporal modelling worked quite effectively for the introduced anomaly types. However,

the performance of the LSTM Autoencoder does not imply that it will have 100% recall rate in reality. The synthetic anomalies can be easier to detect than some real-world problems.

Correct and Incorrect Detection Outcomes

This sub-section analyzes the count of the number of anomalies detected, anomalies missed, correctly recognized normal instances, and false alarms. The count helps to explain better the differences in performances noted in Table 2. The true positive is the correct identification of an anomaly, while the false positive is the misidentification of a normal instance as an anomaly.

Table 3. Classification Outcomes on the Test Dataset

Method	True positives	False positives	True negatives	False negatives
Isolation Forest	38	22	1,328	112
Autoencoder	104	73	1,277	46
LSTM Autoencoder	150	18	1,332	0
Graph Autoencoder	121	99	1,251	29

The reasons behind the classification results reveal the reason for the best F1-score by the LSTM Autoencoder. It managed to detect all 150 anomalies and did not miss any anomaly, but it made mistakes by incorrectly classifying 18 cases that were actually normal. On the other hand, Isolation Forest did not detect 112 anomalies, even though it produced only 22 incorrect classifications. This proves that the threshold it chose for anomaly detection was conservative. The Standard Autoencoder missed 46 anomalies but produced 73 false alarms, and Graph Autoencoder detected 121 anomalies but gave false positive results for 99 normal cases. Thus, one can observe that the process of anomaly detection consists of a trade-off between false negatives and false positives. In practice, the preferable option will depend on the implications of the specific mistakes made. In some cases, it might be less dangerous to miss an important event than to analyze one additional alert.

Detection Across Different Anomaly Types

The telemetry of the system can demonstrate various kinds of abnormal behavior. The spikes are the sudden change in the individual metrics, the shift is a larger deviation in the metric from the expected value, and the relational anomalies are where the combination of variables shows some unusual behavior (Li & Jung, 2021). In this sub-section, we will compare the recall among these three types of anomalies to find out if the selected methods perform equally well in detecting all of them.

Table 4. Recall by Injected Anomaly Category

Anomaly type	Cases	Isolation Forest recall (%)	Autoencoder recall (%)	LSTM Autoencoder recall (%)	Graph Autoencoder recall (%)
Spike	50	12	80	100	88
Shift	50	18	62	100	72
Relational	50	46	66	100	82

The LSTM Autoencoder had 100% recall in all 3 injected anomaly categories. The Autoencoder was most successful in finding spikes, achieving 80% recall, however, it had 62% recall for shifts and 66% recall for relational anomalies. The Graph Autoencoder had 88% recall for spikes, 72% for shifts, and 82% for relational anomalies. Isolation Forest was the most successful in relational anomalies, achieving 46% recall, but had a 12% recall for spikes and 18% for shifts. This difference indicates that the model performance is affected by the type of anomaly and its representation in the model. The superior results of the LSTM Autoencoder might be influenced by the selected anomaly magnitude and patterns. Anomalies in a real-life scenario can be more gradual, intermittent, noisy or very subtle. Hence, further testing is required in varying magnitudes, lengths of sequence, operational events, and workloads in order to choose a model for production use.

False Alarms and Overall Discrimination

This section discusses the concepts of the false positive rates and ROC-AUC score. False positive rates refer to the rate of classification of normal cases as abnormal while ROC-AUC score refers to the ability of the classifier to distinguish between abnormal and normal cases. The higher the ROC-AUC score, the better the ability of the classifier, and the lower the false positive rate, the better.

Table 5. Alert Reliability and Anomaly Ranking

Method	False positives	Normal test cases	False-positive rate (%)	ROC-AUC (%)
Isolation Forest	22	1,350	1.63	86.25
Autoencoder	73	1,350	5.41	86.09
LSTM Autoencoder	18	1,350	1.33	100.00
Graph Autoencoder	99	1,350	7.33	95.30

LSTM Autoencoder recorded the least false-positive rate at 1.33% and the highest ROC-AUC rate of approximately 100%. Graph Autoencoder attained the ROC-AUC of 95.30%, meaning that it had very good ranking and yet had the highest false-positive rate of all at this threshold. Isolation Forest algorithm recorded the lowest false-positive rate at 1.63%, while its low recall rate means that it failed to identify most anomalies. Standard Autoencoder had the ROC-AUC of 86.09% and the false-positive rate of 5.41%. This clearly illustrates the need to consider both ranking ability and performance at a particular threshold together (Malhotra & Sharma, 2021). The ROC-AUC analysis considers the discrimination power across various thresholds, while false-positive rate refers to one specific point on the curve. The almost perfect ROC-AUC rate of LSTM autoencoder is specific to this particular dataset and might have been made possible by the separability of anomalies in this data.

Computational Efficiency

Detection accuracy is not the only criterion for choosing an anomaly detection technique. In case of system monitoring, the models need to deal with continuous telemetry streams while having restricted computing capabilities. In this section, we will evaluate the time it takes to train the model and perform average inference time per one testing instance. These times were measured in this run and depend on the particular software and hardware environment.

Table 6. Training and Inference Costs

Method	Training time (seconds)	Inference time (ms per observation)
Isolation Forest	0.619	0.030
Autoencoder	2.941	0.001
LSTM Autoencoder	16.190	0.030
Graph Autoencoder	5.483	0.002

Isolation Forest training took about 0.619 seconds, being the fastest training method for the present case study. Standard Autoencoder training took 2.941 seconds, whereas training of the Graph Autoencoder took 5.483 seconds. Training of the LSTM Autoencoder took the longest, i.e., 16.190 seconds, due to extra calculations needed in sequence modeling. Execution times were rather small for all models, varying between about 0.001 to 0.030 milliseconds per record. All these data are execution times of the current run and may be different depending on the hardware, batch size, implementation details, and software configuration. The LSTM model involved extra calculations and provided significantly better detection performance compared to the other models. If this tradeoff is worth it or not is an open question to be answered with respect to the monitoring system, the cost of false positives, and how often the model is to be retrained.

Anomaly Threshold Calibration

The anomaly detection models produce scores rather than classifying as either normal or anomalous. The score needs to be converted to a decision using a cutoff value. For this experiment, we use the 99th percentile score for each model's validation phase as the cutoff value. Any observation that has a score greater than the selected cutoff value is considered to be an anomaly.

Table 7. Model-Specific Detection Thresholds

Method	Validation threshold	Score used for thresholding	Calibration percentile
Isolation Forest	0.610	Isolation anomaly score	99th
Autoencoder	1.221	Reconstruction error	99th
LSTM Autoencoder	0.861	Sequence reconstruction error	99th
Graph Autoencoder	1.652	Graph reconstruction error	99th

Each approach uses a unique numerical threshold as its anomaly scores have distinct calculation methods. In particular, Graph Autoencoder employs the highest threshold of 1.652, whereas Isolation Forest features the lowest threshold of 0.610. The thresholds above do not mean that one approach is more accurate than another one, as they have distinct score scales. What matters here is the ability of each threshold to differentiate between anomalies and normal observations. While 99th percentile calibration method intends to minimize false positive results for normal validation data, the false positive rates in the test data set range from 1.33% to 7.33%. It means that the performance of each approach may depend on the distribution of telemetry as well as injection patterns in both validation and test datasets. Thus, the selection of appropriate thresholds deserves further investigation in case of unsupervised learning approaches (Sarmadi & Yuen, 2021). In the future, one can investigate the use of other percentiles, as well as adaptive thresholds.

Method Selection and Practical Implications

The last comparison provide the strengths and weaknesses of each of the 4 methods discussed. Choosing the model involves taking into account the detection ability, number of false alarms, processing expenses, and implications of failure to detect the incidents. Unlike a simple metric-based comparison, this one will provide an adequate interpretation of each model's results.

able 8. Practical Assessment of Detection Methods

Method	Main strength	Main limitation	Interpretation
Isolation Forest	Low false-positive rate of 1.63%	Recall of 25.33%	Conservative detection; misses many injected anomalies
Autoencoder	Recall of 69.33%	False-positive rate of 5.41%	Useful reconstruction-based baseline
LSTM Autoencoder	F1-score of 94.34%; recall of 100%	Highest training time at 16.190 seconds	Strongest overall result on this synthetic dataset
Graph Autoencoder	ROC-AUC of 95.30%	False-positive rate of 7.33%	Promising anomaly ranking, but threshold refinement is needed

According to the comparison, LSTM Autoencoder emerges as the best method in this experiment since it managed to score the highest F1-score, highest recall, and lowest false positives. But at the same time, it consumed the most amount of time for training. While the Isolation Forest can still prove useful in scenarios where rapid training is essential and errors of omission are less expensive, a standard Autoencoder remains an easy-to-use option for reconstruction while Graph Autoencoder exemplifies the possibility of modeling connections between the telemetry metrics (Yang et al., 2021). These results imply that the choice of the method should be based not only on speed but also on the requirements of the monitoring system. Nonetheless, all comparisons have been conducted based on one artificial dataset and one iteration. Thus, these results do not define universal superiority of the methods.

Discussion

The experiments show the possibility of applying unsupervised learning techniques to anomaly detection in system telemetry that involves several variables. Out of the 4 techniques compared, the best result was obtained using the LSTM Autoencoder with the F1-score of 94.34%, 100% recall, and the 1.33% false-positive rate. The application of the temporal dependence model may be beneficial for identifying an abnormal pattern in telemetry. It showed better results than the Isolation Forest, standard Autoencoder, and Graph Autoencoder on the synthetic data set. However, the training time needed to apply the LSTM algorithm is the longest (16.190 s). Graph Autoencoder yielded 95.30% ROC-AUC and 80.67% recall performance, indicating that relationship modeling between telemetry data could help with anomaly detection (Wihayati & Wibowo, 2021). However, the 7.33% false positive rate of Graph Autoencoder shows the significance of choosing the right threshold. Isolation Forest performed better when considering training time and lower false positive rate but has 25.33% recall score and hence failed to detect many anomalies injected in the dataset. It is clear from these results that methods should not be chosen based on any single criterion like computational efficiency or a single metric evaluation (Xu et al., 2018). There are limitations to the research because it employs artificial data, a single execution of the experiment, and pre-defined anomalies. There may be more complexities in real system telemetry that were not captured through this method. It is necessary for future research to test the same methodology using operational data, repeat the experiment, test distribution changes, and adaptive thresholds.

Conclusion

This work has evaluated the efficiency of Isolation Forest, Autoencoder, LSTM Autoencoder, and Graph Autoencoder for unsupervised anomaly detection in multivariate system telemetry data. From the experimental results presented, the LSTM Autoencoder outperformed other techniques by providing the highest score in terms of F1-score with 94.34%, recall with 100%, and a lower false-positive rate of 1.33%. In addition to that, Graph Autoencoder showed efficient detection capabilities, with ROC-AUC of 95.30%, while its false positive rate is higher and equaled 7.33%. On the other hand, Isolation Forest has a low rate of false positives but could only detect 25.33% of anomalies. From the findings above, it is evident that temporal reconstruction has the potential to be helpful for detecting various anomalies in system telemetry. Nevertheless, the results presented require to be verified through more practical examples because the analysis used illustrative experimental data.

Reference List

1. Audibert, J., Michiardi, P., Guyard, F., Marti, S., & Zuluaga, M. A. (2020). USAD: UnSupervised anomaly detection on multivariate time series. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 3395–3404). Association for Computing Machinery. https://doi.org/10.1145/3394486.3403392
2. Yang, L., Ma, Y., Zeng, F., Peng, X., & Liu, D. (2021). Improved deep learning based telemetry data anomaly detection to enhance spacecraft operation reliability. Microelectronics Reliability, 126, 114311. https://doi.org/10.1016/j.microrel.2021.114311
3. Deng, A., & Hooi, B. (2021). Graph neural network-based anomaly detection in multivariate time series. Proceedings of the AAAI Conference on Artificial Intelligence, 35(5), 4027–4035. https://doi.org/10.1609/aaai.v35i5.16523
4. Elmrabit, N., Zhou, F., Li, F., & Zhou, H. (2020, June). Evaluation of machine learning algorithms for anomaly detection. In 2020 international conference on cyber security and protection of digital services (cyber security) (pp. 1-8). IEEE. https://ieeexplore.ieee.org/abstract/document/9138871/
5. Hundman, K., Constantinou, V., Laporte, C., Colwell, I., & Soderstrom, T. (2018). Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 387–395). https://doi.org/10.1145/3219819.3219845
6. Malhotra, R., & Sharma, A. (2021). Threshold benchmarking for feature ranking techniques. Bulletin of Electrical Engineering and Informatics, 10(2), 1063-1070. https://beei2.org/index.php/EEI/article/view/2752

7. Mozo, A., Ordozgoiti, B., & Gomez-Canaval, S. (2018). Forecasting short-term data center network traffic load with convolutional neural networks. PloS one, 13(2), e0191939. https://ieeexplore.ieee.org/abstract/document/8425798/
8. Sarmadi, H., & Yuen, K. V. (2021). Early damage detection by an innovative unsupervised learning method based on kernel null space and peak-over-threshold. Computer-Aided Civil and Infrastructure Engineering, 36(9), 1150-1167. https://onlinelibrary.wiley.com/doi/abs/10.1111/mice.12635
9. Seth, S., Singh, G., & Kaur Chahal, K. (2021). A novel time efficient learning-based approach for smart intrusion detection system. Journal of Big Data, 8(1), 111. https://link.springer.com/article/10.1186/s40537-021-00498-8
10. Li, G., & Jung, J. J. (2021). Dynamic relationship identification for abnormality detection on financial time series. Pattern Recognition Letters, 145, 194–199. https://doi.org/10.1016/j.patrec.2021.02.004
11. Su, Y., Zhao, Y., Niu, C., Liu, R., Sun, W., & Pei, D. (2019). Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 2828–2837). Association for Computing Machinery. https://doi.org/10.1145/3292500.3330672
12. Wihayati, & Wibowo, F. W. (2021, March). Prediction of air quality in Jakarta during the COVID-19 outbreak using long short-term memory machine learning. In IOP Conference Series: Earth and Environmental Science (Vol. 704, No. 1, p. 012046). IOP Publishing. https://iopscience.iop.org/article/10.1088/1755-1315/704/1/012046/meta
13. Xu, Q., Huang, G., Yuan, Y., Guo, C., Sun, Y., Wu, F., & Weinberger, K. (2018). An empirical study on evaluation metrics of generative adversarial networks. arXiv preprint arXiv:1806.07755. https://arxiv.org/abs/1806.07755