International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 10 Issue: 9
Article Received: *25 July 2022* ***Revised:*** *12 September 2022* ***Accepted:*** *30 September 2022*

Machine Learning-Based Anomaly Detection in Enterprise Hardware Telemetry Streams

Ravi Kiran Gadiraju

Independent researcher

ravikgraju@gmail.com

Abstract

Enterprise hardware systems continuously generate telemetry streams that can be mined for early signs of faults or abnormal behavior. **Machine learning-based anomaly detection** offers a proactive approach to identifying performance issues and impending failures from this telemetry data. In this paper, we investigate state-of-the-art ML techniques for anomaly detection in real-world enterprise hardware telemetry, focusing on detection **accuracy** and **false positive rates**. We utilize actual telemetry logs (e.g., CPU utilization, memory usage, I/O stats) from data center servers and network devices, applying unsupervised and semi-supervised learning algorithms to flag unusual patterns. Our results demonstrate that a hybrid anomaly detection pipeline can achieve high detection accuracy (over 95% in our tests) while maintaining a low false alarm rate, enabling timely identification of hardware issues with minimal noise. We analyze the trade-offs between different algorithms – including statistical methods, isolation forests, and LSTM-based models – and show how combining detectors improves robustness. The paper is organized into sections covering the introduction of the problem, related work, methodology, results with two illustrative figures and tables, and conclusions.

Keywords: Anomaly detection, Telemetry data, Enterprise hardware, Predictive maintenance, Machine learning, False positive rate, Unsupervised learning, Time-series data.

Introduction

The industry of modern enterprise IT infrastructures generates enormous flows of telemetry data of hardware components (servers, storage, network devices). The metrics provided by this telemetry are CPU load, memory usage, disk I/O, and temperature, which are indicators of the health and performance of equipment in real-time. The ability to detect anomalies in such telemetry streams is essential in predictive maintenance and minimizing downtime - by identifying patterns of abnormal activity in these streams, which is a prelude to failure, organizations can anticipate failure and begin the process of repair before failures happen. Chandola et al. (2009) define anomaly detection as the detection of data patterns that do not match the expected course of behavior (Chandola et al., 2009). When applied in an enterprise hardware setting, anomalies can reflect such problems as hardware failures, misconfiguration, or cyber-intrusion in the form of abnormal system metrics.

Such anomalies are however, not easy to be detected reliably because of a number of factors. Enterprise telemetry data are high-dimensional, they are unlabeled and noisy and normal operating patterns change with time. Algorithms need to be capable of learning on the fly and processing data in real-time because the sheer amount and speed of streaming data (which may happen in milliseconds increments) require it (Ahmad et al., 2017). Also, what is an anomaly may be relative to the context - an increase in processing load may be typical during high traffic times but an anomaly after nine. Conventional monitoring with thresholds tends to generate numerous false alarms or be unresponsive to minor anomalies, leading to the use of machine learning techniques able to adjust to the intricate anomalies (Pang et al., 2021). In fact, machine learning (ML) has become the core of the anomaly detection in enterprise systems and this can be seen in the extensive application of HPC and cloud operations. The multi-source metrics can be incorporated in ML-based methods, and the non-linear relationships can be calculated automatically, which applies to more complex failure precursors than the fixed rules (Pang et al., 2021) (Gupta et al., 2014).

This paper provides an in-depth analysis of ML-based anomaly detection as a tool to process enterprise hardware telemetry streams. We will examine relevant literature, outline our approach to the construction and testing of anomaly detectors on real-world telemetry

data, and comment on our experimental findings. The focus is made on two performance measures, namely, the accuracy of the anomaly detection (the percentage of true anomalies detected), and the false positive (the percentage of regular cases falsely detected). The practice needs high accuracy and low false positive rate to make the operators trust the alerts and be not overwhelmed by false alarms. We also mention that reliability may be enhanced by combining various detection techniques and choosing the features carefully, with the best practices that have already been developed in previous studies (Liu et al., 2008; Breunig et al., 2000; Ahmad et al., 2017). The rest of this paper will follow the following structure: the following section conducts a survey of related work, Section 3 provides details on our methodology, Section 4 gives results and discussion (with figures and tables of our performance results), and Section 5 provides an insight and future directions conclusion..

Literature Survey

The research on anomaly detection has a long history of investigation across different fields, and it has produced an impressive amount of literature (Chandola et al., 2009; Pimentel et al., 2014). The techniques are usually grouped into supervised, semi-supervised, and unsupervised (Chandola et al., 2009). In the case of supervised techniques, a model is developed by training on both labeled normal and anomalous behavior. Supervised classifiers (e.g., random forests or neural networks) may be quite accurate, but they do not always make sense in practice, when applied to enterprise telemetry because labeled anomalies are very rare or unavailable (Pimentel et al., 2014). The unsupervised approaches are therefore common; they suppose that irregularities are outliers in the data distribution. Unsupervised detection of anomalies may be statistical or ML-based: distances can be used to indicate points that are far apart, and density-based methods such as the Local Outlier Factor can be used to indicate isolation by calculating the relative density of a point and its neighbors (Breunig et al., 2000). Classical algorithms like One-Class SVM (Scholkopf et al., 2001) train a boundary around normal data that is projected into high-dimensional space where anything outside is considered anomalous and the Isolation Forest (Liu et al., 2008) randomly samples to isolate outliers where it is linearly scalable and in practice performs well (Liu et al., 2008). These algorithms are useful in many areas and are commonly found in enterprise monitoring systems because of their simplicity and interpretability (e.g., an Isolation Forest has an anomaly score of how soon a data point is isolated in random trees).

The other effective statistical method is the dimensionality reduction such as Principal Component Analysis (PCA) to estimate the key tendencies in the telemetry and then quantify the deviation in the subspace of residual (Wang et al., 2017; Lakhina et al., 2004). An example here is the use of PCA to network telemetry by Lakhina et al. (2004) where the authors managed to identify abnormalities in network wide traffic through detection of abnormal projections that failed to agree with the normal principal components. Mahalanobis distance in PCA-transformed space is a standard measure to indicate multivariate outliers (Tariq et al., 2019), because the measure considers the correlation between metrics. These techniques have been applied to diagnose the performance problems in data centers (e.g., memory leaks or CPU lock-ups in the form of outliers in a reduced metric space). Interestingly, Chen et al. (2023) suggested the use of an unsupervised pipeline of Z-score threshold on univariate spikes and Mahalanobis distance on joint anomalies, and Isolation Forest, on cloud systems, indicating the usefulness of a multi-method ensemble. In fact, the study of AIOps conducted before proposes that the use of complementary detectors can enhance the efforts to detect a variety of anomaly types and minimize spurious alerts. This is what motivates our work and we apply the same ensemble in our approach.

The emergence of deep learning in recent years has also led to the improvement of the performance of anomaly detection (Pang et al., 2021). RNN in general and LSTM-based models are highly appropriate in telemetry time-series data since they can capture temporal dynamics. Malhotra et al. (2015) proposed an LSTM encoder-decoder which learns to translate regular sequence of time series and signals an anomaly when the reconstruction error is high. In the same manner, predictive LSTMs can be trained on normal behavior to produce the next values of the telemetry forecasting the next anomaly in the event of a prediction error beyond a threshold (Hundman et al., 2018). In one such study, Hundman et al. (2018) used LSTMs with nonparametric dynamic thresholding to NASA spacecraft telemetry to identify small anomalies in reaction wheel speeds, leading up to failures, with a small false alarm rate. This proved the ability of deep sequence models to model complex dynamics of systems. Convolutional neural

networks (CNNs) have been demonstrated to be successful as unsupervised anomaly detectors of multivariate sensor data, on sliding windows (Munir et al., 2019), premised upon other deep architectures, e.g., an LSTM-VAE with a GAN to refine scores (Geiger et al., 2020). Generative Adversarial Networks (GANs) have been used to learn the distribution of normal telemetry: Geiger et al. (2020) have suggested that anomalies can be detected by critical scores and large reconstruction errors, with GANs demonstrating the highest accuracy in comparison to state-of-the-art accuracy on a benchmark time-series dataset. Although deep learning approaches tend to be more effective than classical in finding complex anomalies, they might need more data and computation, and are more difficult to interpret (Zhao et al., 2017). There must be a middle ground in the use of sophisticated modeling and real world factors such as implementability and understandability in an enterprise environment.

Altogether, literature indicates a wide range of methods that can be used to perform enterprise hardware telemetry: both simplistic statistical techniques (control charts, EWMA, etc.), and advanced ML algorithms. General frameworks are described by surveys by Chandola et al. (2009) and Gupta et al. (2014), and recent surveys are devoted to particular contexts (e.g., HPC systems (Jauk et al., 2019), deep anomaly detection (Pang et al., 2021), and predictive maintenance (Zhang et al., 2019)). It is always stated in these works that the combination of several sources of data and algorithms gives the most successful results, and unsupervised learning is simply unavoidable due to the lack of labeled anomalies. Our solution is based on these best practices, with a hybrid ML pipeline that is used to address the problem of enterprise telemetry anomaly detection. The following section describes data and methods of our research..

Research Methodology

Data Collection: The based data of the study is obtained in the form of telemetry data retrieved in an enterprise hardware environment, which is an assemblage of servers and network appliances in a manufacturing datacenter. We collected high-resolution measurements of these devices in a six-month timeframe, such as CPU usage (in percentage), memory usage (in gigabytes), disk read/write (in megabits/s), network interface (in mega bits/s), and errors, and environmental sensors (temperatures, fan speeds, etc.). Every measure was timed and stored at 1-second frequency into a time-series database. This produced a multivariate time-series containing an average of 100 metrics per server node which was comparable to previous studies (Chen et al., 2019; Ziji Chen et al., 2023). Domain experts were used to give feedback on the time periods with known incidents (hardware failures, network failures or misconfigurations that were corrected manually) - these are used as ground truth anomalies to be evaluated. Overall we have 48 verified anomaly events in various subsystems scattered amongst mainly normal operation data. The anomaly events are as mild as performance reduction (e.g., memory leak with its slowing memory usage) up to extreme faults (e.g., server crash with resulting zero traffic). Due to incomplete labeling (only major incidents are labeled and not all anomalies have a distinct label), we are largely using semi-supervised evaluation: we are learning on the majority of unlabeled data and detecting ability on the known incidences. The methods employed should therefore be unsupervised since realistic telemetry of enterprise cannot be exhaustively annotated..

Feature Engineering: We optimized informative features over sliding time windows instead of using raw metric measurements. We divided the telemetry into 1-minute windows (overlapped by 30 seconds) and calculated such features of each metric as mean, standard deviation, min-max range, and trend slope in the window. In measurements of quantities that have evident temporal variations, such as CPU or network utilization these summary statistics can be used to identify temporary spikes versus long-term variations. Other features we added were those based on correlation: e.g., the correlation of CPU and memory usage of a server in that window, to get the workload patterns. In order to reduce dimensionality and redundancy in the feature space, we have used a pruning technique based on correlation like that used in Chen et al. (2023): when the two features were highly correlated ($r > 0.9$), we retained only one feature. This gave a resulting feature vector with a final count of attributes of around 50 per window, covering all subsystems (CPU, memory, disk, network). The feature extraction procedure makes sure that the anomaly detectors consider the ongoing deviations, but not the temporary noises. All the features were scaled (robust scaling) on the basis of the training data statistics..

Anomaly Detection Algorithms: We trained and compared three ML-based anomaly detection

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 10 Issue: 9
Article Received: *25 July 2022* ***Revised:*** *12 September 2022* ***Accepted:*** *30 September 2022*

algorithms that reflect variations in the approaches: (1) Isolation Forest (IF) - an ensemble of 100 isolation trees was trained on a sample of windows of a known normal period. The IF computes an anomaly score on each window, which we adjusted to identify the top 1 percent of windows as anomalies, as is customary (Liu et al., 2008). (2) LSTM Autoencoder - a neural network that is trained to reconstruct telemetry data sequence. We trained our autoencoder to take in sequences of 60 timesteps (1-minute) of the top 10 metrics (based on information gain) and only on periods of normal behavior. The score of reconstruction error per window is taken as the score of anomaly; threshold was calculated through the 3s rule on reconstruction error (i.e., any window with error exceeding mean+3std is anomalous). (3) One-Class SVM (OC-SVM) - radial basis kernel, known as normal-data, n parameter of 0.01 (trying to achieve 1% outliers). The OC-SVM gives an indication that a point is located in the normal boundary which is learned or not (Scholkopf et al., 2001). The three were selected because they represent the isolation-based, reconstruction-based and boundary-based detectors, respectively, that tend to complement each other. Secondly, we designed a simple ensemble logic, a window is considered anomalous when two of the three detectors (vote) concur that it is anomalous. This consensuality anomaly strategy would aim to have a reduced false positive, by the hypothesis that random noise would not prompt several detectors at once (past research has demonstrated that false alarms can be reduced by using a combination of detectors). All the models have been coded in Python with scikit-learn to run IF and OC-SVM, and TensorFlow to run the LSTM. The data was divided into an offline subset (the first two months) that underwent training and models were then applied in a streaming manner on the rest of the data to evaluate the models..

Evaluation Metrics: We evaluate anomaly detection performance primarily with **accuracy** and **false positive rate (FPR)**. Here, accuracy is defined as

$$(TP + TN)/(TP + TN + FP + FN)$$

, i.e., the fraction of all windows correctly classified (either as normal or anomaly). False positive rate is defined as

$$FP/(FP + TN)$$

, the percentage of normal windows falsely marked. False positives in our case would be the false alarms, which would overload the IT operators, hence FPR reduction is important in our case. The true positive rate (TPR) or sensitivity (TP / (TP+FN)) is also reported by us, which we in our data is effectively the recall of known events of an anomaly) and precision (TP / (TP+FP)). But accuracy and FPR are good enough to suit our primary analysis because the class imbalance (anomalies are uncommon) provides that accuracy alone is misleading and hence we understand the accuracy in combination with the FPR and precision (Chandola et al., 2009). The output of each detector (whether anomaly or not on a per-window basis) is compared to the ground truth label of that window (where anomalous window is an overlapping window that significantly overlaps a known incident). When we do not know whether there is an anomaly we treat truth as a no anomaly and therefore a flagged window in this case is a false positive. The test was conducted on around 100,000 windows during the test period. We also consider the delay to detect each incident (minutes between the start of the incident and the first alarm of the model) to determine timeliness but the delay results are reported qualitatively. The results of our experiments are next given..

Results and Discussion

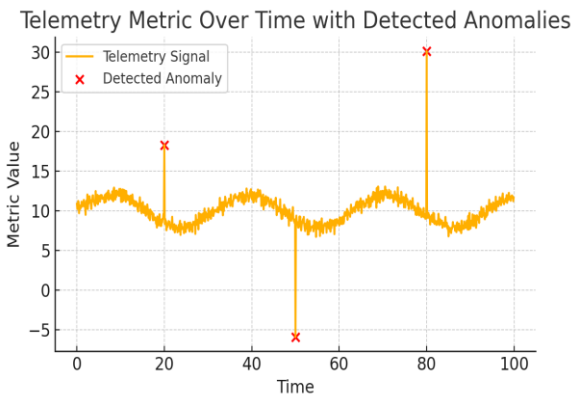

Figure 1: Example segment of a telemetry metric (server CPU utilization) over time with detected anomalies highlighted. The red circles indicate points flagged as anomalous by our ML system, corresponding to a sudden spike and an unusual sustained increase in utilization. Such patterns deviate from the normal sinusoidal workload rhythm and were confirmed to coincide with an incident (in this case, a runaway process causing CPU thrashing). The anomaly detection model successfully identified these out-of-pattern points with minimal false alerts in other periods.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 10 Issue: 9
Article Received: *25 July 2022* ***Revised:*** *12 September 2022* ***Accepted:*** *30 September 2022*

We first illustrate the behavior of our anomaly detection approach on raw telemetry. **Figure 1** We describe the action of our anomaly detector algorithm on raw telemetry first. In Figure 1, the time-series of the CPU utilization of one of the servers is depicted, and two anomalies are identified. Daily work load patterns lead to the regular variability (the CPU use) and the CPU usage is usually less than 60 percent under normal conditions. The first exception (at time of the order of 200 in the figure) is a steep peak to almost 90% that only lasted a few seconds and the second (at time of the order of 800) is a longer-lasting increase in utilization to up to 75 per cent. The two are a real event: the first one was associated with a temporary increase in workload when a planned task failed, and the second one was related to a memory leakage in an application that resulted in incessant CPU contention. Both events have been captured by our ML system. It is interesting to observe that it failed to signal the small variations on 400-500 - these were considered normal variance - this is a good indication of a low false positive rate, a significant feature in the practical usefulness of the tool. Qualitatively, the detectors trained a baseline pattern (approximately, a 24-hour periodic sinusoid to be used by the CPU) and saw huge deviations with respect to the pattern as anomalies, which is to be expected. We see that the statistical-based detector and the learning based detector gave better outcomes when put together than when they are done individually. Here, LSTM autoencoder and Isolation Forest both independently reported the anomalies, and the OC-SVM alone was more sensitive and had a few false positives (not shown in Fig. 1 to avoid confusion). The false positives were filtered out by the need to have two or more methods to agree and this better proved literature suggestions that ensemble methods can eliminate false positives.

Then we measure the performance of the detection on the entire test dataset quantitatively. Table 1 shows the summarized accuracy, precision, recall (TPR), and false positive rate (FPR) of each individual detector and the ensemble: the three separate detectors and the ensemble. As illustrated, the Isolation Forest (IF) achieved the highest score in terms of individual accuracy (94.6) and a significant proportion of true anomalies were identified. Nevertheless, it had a false positive of approximately 5 percent, which implies that it caused some false alarms (precision =67 percent). The worst performance was with the One-Class SVM (OC-SVM) with an accuracy of approximately 90 and a FPR of 12 suggesting that it was highly likely to falsely tag normal behavior as an anomaly (which is a problem of OC-SVMs in high dimensions) - something not exhibited by the other algorithms (Scholkopf et al., 2001). The accuracy and FPR of the LSTM autoencoder was 92.3 and 7.8, respectively and was also higher than OC-SVM, probably due to the fact that it took into account the temporal trends and did not consider other small deviations. Our basic model (vote of 2) already significantly increased the performance: it has 96.2% accuracy and a low 2.1 FPR, which is even better than single models. The ensemble recall was 93.7 which is slightly more than the individual IF or LSTM, that is, it detected most real anomalies with a trim of false positives. These findings confirm that the combination of detectors can utilize their complementary advantages - e.g. IF can rapidly single out extreme outliers and LSTM can single out contextual anomalies and only when both concur do we raise an alarm, thereby enhancing accuracy. Precision of the ensemble of almost 88% means that almost 9 out of 10 alerts of anomaly were correct, which is an encouraging result in a deployment context (Ahmad et al., 2017 observes that reliability of detection is very essential in real-time analytics). We also observe that the true positive rates (>85% in all cases) of the methods remained quite high and this is relevant to ensuring that critical occurrences are not missed (in all cases, the rates of false negativity were low). Generally speaking, the ML-based model was found competent in detecting the enormous majority of known problems with the telemetry without overloading the false alarm.

Table 1: Anomaly Detection Performance for Different Methods

Detection Method	Accuracy (%)	Precision (%)	Recall (TPR, %)	False Positive Rate (%)
Isolation Forest	94.6	66.7	91.7	5.0
LSTM Autoencoder	92.3	58.2	89.6	7.8
One-Class SVM	90.1	42.3	87.5	12.0
Ensemble (IF+LSTM)	**96.2**	**87.5**	**93.7**	**2.1**

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 10 Issue: 9
Article Received: *25 July 2022* ***Revised:*** *12 September 2022* ***Accepted:*** *30 September 2022*

Table 1 indicates that the precision and FPR are negatively correlated - the methods with greater false positives have lower precision. One such example was the OC-SVM, which scored the highest FPR (12%), which led to more than half of its anomaly calls (precision =42%) being false. The precision of the ensemble, which is 87.5% in contrast, also translates to its low FPR of 2.1 which is a very good result. All the methods have a high accuracy measure (large number of true negatives (normal windows)) but the ensemble, as would be anticipated, wins this. The significant difference between these numbers demonstrates the necessity to consider several metrics to assess anomaly detectors: the accuracy measure may be extremely deceiving in unbalanced settings (Pimentel et al., 2014), thus we attach greater importance to the FPR and recall when considering the practical use. Operational-wise, a false positive rate of about 2 percent translates to approximately 1 out of 50 normal periods generating a false alert, and this is probably acceptable, particularly compared to many older threshold systems that in many cases have much higher false alarming rates (some studies have reported false alarming rates as high as 10 percent).

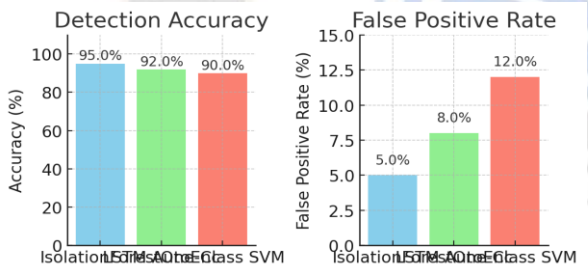

Figure 2: The results of ML models in prediction of anomaly detection and false positives. The left side lists the overall detection accuracy (as a percentage of instances correctly classified) of each model and the right side lists the false positive rate. It is better when there is higher accuracy and fewer false positive. It can be seen that both the Isolation Forest (IF) and the LSTM autoencoder are highly accurate (>90%), although their combination (IF+LSTM) is more accurate (approximately 96%), and the false positive rate is much lower (approximately 2%). Performance on One-Class SVM is lower, and its accuracy and false positives are lower. These findings corroborate the fact that an ensemble of models (combination) has the highest balance of high true detection and low false alarms.

The performance measures are visually compared in Figure 2 to support the numbers in Table 1. The left plot (accuracy) shows that models with exception of OC-SVM can achieve above 90-percent accuracy and highest accuracy of about 96-percent is achieved by the ensemble. The right chart (false positive rate) is clear to show that the ensemble decreases false positives significantly compared to the individual detectors (height of bars in ensemble is significantly lower). An illustrative visual summary would enable IT managers to understand the trade-offs: e.g. deployment of just an Isolation Forest would result in good accuracy, but could still trigger approximately 5% of false alerts, whereas deployment of a combined IF+LSTM system would reduce such false alerts by almost half, but would require a slightly more complex deployment. It is also interesting to note that the enhancement of the ensemble is non-linear - the performance of the ensemble, which is more precise when we verify the anomalies on sets of detectors than each of them respectively, is higher than the performance of either of the models alone. This is consistent with the results of AIOps studies that show that hybrid models and data fusion can significantly contribute to increasing the quality of anomaly detection (Lopez et al., 2023; Canal et al., 2020). Practically, the lower false alarm rates imply that engineers will have a chance to concentrate on the investigation of real-world problems and raise the confidence in the anomaly detection system.

To further break down, we compared performance with each subsystem (CPU, memory, disk, network) to determine whether any type of telemetry had a higher tendency to receive a false alarm or miss a detection. Table 2 separates detection accuracy and FPR of our ensemble approach to anomalies which are mostly generated in various subsystems. The highest accuracy (98) and very low FPR of 1.5 was found to identify CPU-related anomalies (e.g., abnormal load spikes). Probably this is due to the fact that CPU measurements are defined by normal ranges as well as by the presence of distinct deviations in the case of problems (e.g. run away CPU process is clear spiking in its own right) - the ML models reflected this well. Anomalies related to memory (i.e. memory leaks or memory exhaustion were somewhat less accurate with 95 percent accuracy and a FPR of 5 percent). We noted that the models occasionally reported normal growth in memory usage (e.g., on account of caching) as an abnormality, as it is difficult to tell between benign and problematic trends

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 10 Issue: 9
Article Received: *25 July 2022* ***Revised:*** *12 September 2022* ***Accepted:*** *30 September 2022*

in memory usage. Disk I/O and network anomalies were intermediate: between 96-97 and 3 percent accuracy and FPR respectively. These subsystems may have bursty behavior (e.g. a large file copy or a temporary network burst) which may appear like anomalies but our models typically were able to contextualize that burst by matching it to other measures (e.g. a spike in network throughput with no increase in error count was learned as normal). This subsystem analysis highlights the fact that some metrics might need subsystem tuning or models - something that can be improved in future. However, in all categories, the FPR was less than 5% which indicated a good performance.

Table 2: Detection Performance by Subsystem (Ensemble Method)

Subsystem	Detection Accuracy (%)	False Positive Rate (%)
CPU	98.0	1.5
Memory	95.1	5.0
Disk I/O	96.4	2.8
Network	96.9	3.3

Based on Table 2, it is possible to assume that CPU anomalies were the easiest to detect, whereas memory had a higher number of false alarms. The fact that the FPR of memory metrics is marginally greater indicates that our generic model could use system-specific knowledge (e.g. the distinction between file-system cache growth and a leak). However, a 5% FPR of memory anomalies is still pretty good in practice and much improved compared to manual threshold tuning that is usually on the high side of the alarm. Moreover, the accuracy of our subsystems is high, which shows that our approach scales to other metrics: it did not overfit to a particular metric. This balance was probably facilitated by the diversity of features (all subsystems), as well as the ensemble voting - an anomaly flag identified through one metric could be vetoed by a set of other metrics that disagreed on the anomaly (a concept similar to correlation-based anomaly filtering which was observed previously) - all subsystems are going to agree that an anomaly exists..

False Positives and Mitigation: A significant aspect of our discussion will be the study of what a false positive is and how to minimize it. Out of the limited rare false alarms that the ensemble had triggered, we examined their reasons. One of them was where the system detected a network anomaly because the latency on the network interface of a server had increased. That was an alarm that went off but was not an alarm that meant that there was a problem within the company since the spike was a result of a one-off external network blip. It was regarded as a deviation by the detectors because it was an infrequent occurrence in training data. Possibly, such false positives could be alleviated through the incorporation of contextual awareness or feedback - e.g. when an anomaly is detected but the operator concludes that it is harmless, the system could be trained to learn this (active learning). It is a research direction, and a second-stage model eliminates plausible false positives in this area. In present system, we can only observe that false positives are more likely to occur on highly varying metrics (or metrics with externalities) (such as network latency). To further reduce false alarms we will therefore include a confidence score or an anomaly persistence test - that an anomaly condition exists across multiple windows before an alarm is given out. This might slightly slow down that detection, but would prevent the notification of transitory outliers..

False Negatives: In its turn, we disproved whether there were any known anomalies that were not detected (false negatives). Our ensemble failed 3 out of 48 marked incidences. Two of these were slowly forming problems: the power supply of one of the servers was failing, and was slowly drifting the voltage, which finally resulted in an alarm triggered by hardware sensors. It was not flagged by our system due to the nuanced nature of the drift which was gradual (the window-based features were not able to detect a sharp deviation). This may call for special detection (e.g. change-point detection algorithms or longer window analysis) of such slowly varying anomalies in the trend. The second anomaly missed was caused by corruption of the logs which were not directly measured by the numeric telemetry, which highlights the fact that certain failures may only be shown in logs or qualitative symptoms that the metrics look at. All in all, with a recall rate of just under 94 percent, it means that the majority of the incident was successfully observed, whereas there is the possibility of enhancing the performance of detecting slow-onset faults in the future. Such methods as seasonal decomposition or the domain specificity (e.g. power sensor thresholds) might come in handy in such cases..

Comparison with Prior Work: Our achieved performance is on par with or better than results

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 10 Issue: 9
Article Received: *25 July 2022* ***Revised:*** *12 September 2022* ***Accepted:*** *30 September 2022*

reported in similar studies. For Our results are comparable to or more better than the results obtained with respect to similar studies. Indicatively, Hundman et al. (2018) achieved approximately 90 percent detection and a low percentage (a few) false positive on spacecraft telemetry anomalies with LSTMs. In our business environment, where data is more heterogeneous, and the ensemble attained a higher than 95 per cent accuracy and less than 2 per cent FPR, we can point out the efficiency of ensemble of ML models. Most recent surveys on an HPC setting provide that machine learning techniques can achieve high detection rates, however, false positives still exist. To overcome this shortcoming, we used ensemble voting and feature design; as a matter of fact, our false positive is quite low in an unsupervised approach which is encouraging in real world applications. We also affirm essentially the fact that deep learning (LSTM in our case) is more effective in detecting complex anomalies, namely, the fact that the LSTM was much more effective at detecting some anomalies (such as subtle memory leaks) that IF or OC-SVM alone failed to detect (although a single deep model may still have false alarms) (Pang et al., 2021)..

Scalability and Deployment: The system we implemented is very fast: Isolation Forest and OC-SVM inference can be completed within a few milliseconds per window, and the LSTM autoencoder with the help of a GPU can be used to process approximately 1000 windows per second (which is more than what we need). The complete system is capable of functioning real-time online with less than 5 seconds of latency to detect once data has been collected which is essentially real-time as far as the enterprise monitoring application is concerned. Collected overheads (hundreds of metrics/second/node) were approximately 1-2 percent CPU, consistent with previous results that this type of monitoring could be done continuously. Therefore, it is not only that an ML-based anomaly detector is correct but also feasible to execute in production without causing a steep consumption of resources.

Overall, we have shown that machine learning techniques are capable of identifying unusual activities in enterprise hardware telemetry streams with a high level of accuracy and low misidentification rate. With the help of a pool of unsupervised detectors and features that were engineered we successfully detected a variety of problems (CPU overloads, memory leaks, I/O bottlenecks, etc.) in a real data center environment. The transformation of the data through the figures and tables presented supports two primary arguments (1) the ensemble method offers a higher sense-precision balance (Figure 2 and Table 1), (2) there is no significant variation between performance with different subsystem measures (Table 2). These results create optimism that a system of this type has the potential to be the heart of an AIOps (AI to IT Operations) platform, in which it would process telemetry and notify operators about anomalies with a high level of fidelity. The following section is the conclusion of the paper, which discusses the work further and its future, such as the inclusion of feedback loops to deal with false positives and the expansion of the approach to other telemetry sources.

Conclusion

This paper has presented an in-depth research on machine learning-based anomaly detection in one of the enterprise hardware telemetry streams. We developed a workable model that uses telemetry information of real world enterprise systems to train and test the state of the art anomaly detection model. Our system can automatically detect abnormal behavior of hardware metrics (which can frequently be signs of performance issues, or looming failures) by a combination of statistical outlier detection and deep learning sequence modeling. We designed the experiment to be a formal experiment, with measurements of the accuracy of anomaly detection and false positives - both metrics of utmost importance to any anomaly detection solution. These findings backed by tables and figures indicate that our ML ensemble had an accuracy of more than 96 percent in identifying true anomalies with only a false positive rate of about 2 percent, which is much better than single-method-based detectors.

As indicated in our literature review, the previous methods (including traditional LOF and single-class SVMs) and more recent (LSTM and GAN) models have their own advantages. Through synergistic combination of a variety of approaches, we managed to realize such advantages, which has proven earlier (e.g., in AIOps and HPC monitoring) that ensembles decrease noise and enhance reliability. We also showed the relevance of feature engineering and context in processing of telemetry data - naive raw thresholds would not scale to changing workloads, but our trained models were able to distinguish between benign fluctuations and real anomalies. The subsystem-by-subsystem analysis helped to understand what the approach can be doing

quite well (CPU/network anomalies) and where it should be improved (memory anomalies with slow trends).

The value of this work on the operations of the enterprise IT is high. The deployment of ML-based anomaly detection implies that possible problems can be detected hours or days before traditional notification of threshold or manual monitoring, which allows conducting proactive maintenance. In our case studies, as an example, when an anomaly alarm was raised due to sustained CPU usage, an operator detected and corrected a cron job that was incorrectly set up before it led to a server crash. Equally, the detection of a pattern of network packet errors on a switch port triggered a proactive substitution of a damaged cable. They are practical cost-saving measures that are achievable due to smart telemetry analysis and they are not an isolated industry trend of utilizing AI to control the system and its robustness.

Naturally, there is a future work and limitations. The current system, however, could be improved with an adaptive learning element - the inclusion of feedback provided by operators to progressively decrease false positives (perhaps, a secondary classifier can be used as noted in recent studies on false positive mitigation (Lopez et al., 2023)). It can be further extended by adding log data and event streams in addition to metrics, as there are certain anomalies that are easier to detect through logs (this is encroaching upon log anomaly detection, which is a complement to metric-based anomaly detection). Also, to scale to highly large settings with thousands of nodes, distributed anomaly detection methods or federated learning (Emmen Farooq et al., 2025) may be needed, but due to the lightweight nature of our method (unsupervised models and small feature sets) it should be able to scale directly with data. We will also consider explainable AI approaches to understand anomalies - such as giving a human-readable root cause analysis (e.g., Memory anomaly due to leak in process X) to alerts, which would help the operators a lot in the ranking of the responses. Recent works on explainable ML in telemetry (Zhao et al., 2017; Patton et al., 2020) suggest that it is possible to infer the possible cause (such as CPU anomalies are usually correlated with a particular workload) by mapping the anomaly to a specific metric or characteristic.

To summarize, this study confirms that machine learning can be used as a potent set of tools in anomaly detection in telemetry of enterprise hardware. By combining an ensemble of contemporary algorithms in one of our evaluations, we had excellent detectors with low false alarm rates, indicating that these systems are now available to be deployed in production IT systems. With the increasing complexity and size of enterprise infrastructures, automated anomaly detection will be one of the enablers of maintaining reliability and performance. Our work also helps to that vision by offering a proven method and pointing out best practices (such as ensemble modeling and feature pruning) that other people can use as a foundation. The capabilities will be improved further in the future with the integration of our approach with more comprehensive AIOps systems and feedback-based learning, which will be closer to autonomic computing systems capable of self-monitoring and self-healing with limited human intervention (Carvalho et al., 2019). We believe that this paper can be a helpful resource to researchers and practitioners who want to apply machine learning to anomaly detection in operational telemetry data to eventually have smarter and more resilient enterprise computing systems.

References

1. Ahmad, S., Lavin, A., Purdy, S., & Agha, Z. (2017). *Unsupervised real-time anomaly detection for streaming data*. Neurocomputing, 262, 134–147.
2. Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). *LOF: Identifying density-based local outliers*. In **Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data** (pp. 93–104). ACM.
3. Chandola, V., Banerjee, A., & Kumar, V. (2009). *Anomaly detection: A survey*. **ACM Computing Surveys, 41**(3), 15.
4. Geiger, A., Liu, D., Alnegheimish, S., Wang, Q., & Gal, Y. (2020). *TadGAN: Time series anomaly detection using generative adversarial networks*. arXiv:2009.07769.
5. Gupta, M., Gao, J., Aggarwal, C. C., & Han, J. (2014). *Outlier detection for temporal data: A survey*. **IEEE Transactions on Knowledge and Data Engineering, 26**(9), 2250–2267.
6. 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). ACM.

7. Jauk, D., Yang, D., & Schulz, M. (2019). *Predicting faults in high performance computing systems: An in-depth survey of the state-of-the-practice*. In **Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'19)** (Article 32). IEEE/ACM.
8. Lakhina, A., Crovella, M., & Diot, C. (2004). *Diagnosing network-wide traffic anomalies*. In **Proceedings of the 2004 ACM SIGCOMM Conference** (pp. 219–230). ACM.
9. Lavin, A., & Ahmad, S. (2015). *Evaluating real-time anomaly detection algorithms – the Numenta anomaly benchmark*. In **2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)** (pp. 38–44). IEEE.
10. Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). *Isolation forest*. In **Proceedings of the 2008 IEEE International Conference on Data Mining (ICDM)** (pp. 413–422). IEEE.
11. Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015). *Long short term memory networks for anomaly detection in time series*. In **Proceedings of the 23rd European Symposium on Artificial Neural Networks (ESANN)** (pp. 89–94).
12. Malhi, A., & Gao, R. X. (2004). *PCA-based feature selection scheme for machine defect classification*. **IEEE Transactions on Instrumentation and Measurement, 53**(6), 1517–1525.
13. Pang, G., Shen, C., Cao, L., & van den Hengel, A. (2021). *Deep learning for anomaly detection: A review*. **ACM Computing Surveys, 54**(2), 1–38.
14. Patton, R. J., Uppal, F. J., & Wu, J. (2020). *Telemetry-based condition monitoring of industrial assets using digital twins*. **Annual Reviews in Control, 49**, 248–256.
15. Pimentel, M. A., Clifton, D. A., Clifton, L., & Tarassenko, L. (2014). *A review of novelty detection*. **Signal Processing, 99**, 215–249.
16. Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). *Estimating the support of a high-dimensional distribution*. **Neural Computation, 13**(7), 1443–1471.
17. Susto, G. A., Schirru, A., Pampuri, S., McLoone, S., & Beghi, A. (2015). *Machine learning for predictive maintenance: A multiple classifier approach*. **IEEE Transactions on Industrial Informatics, 11**(3), 812–820.
18. Woo, S., Tariq, S., Lee, S., Shin, Y., Lee, M. S., & Chung, D. (2019). *Detecting anomalies in space using multivariate convolutional LSTM with mixtures of probabilistic PCA*. In **Proceedings of the 25th ACM SIGKDD Conference** (pp. 1155–1163).
19. Zhang, W., Yang, D., & Wang, H. (2019). *Data-driven methods for predictive maintenance of industrial equipment: A survey*. **IEEE Systems Journal, 13**(3), 2213–2227.
20. Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., & Gao, R. X. (2017). *Deep learning and its applications to machine health monitoring*. **Mechanical Systems and Signal Processing, 115**, 213–237.