LEAD - Framework for efficient time-series anomaly detection on large scale data using LLMs

Akash Chandrayan [1] **Amir Zidi** [1]
Matthew Reimherr [1 2] **Anis Mjirda** [1]
Abhinav Pradhan [1]

Abstract

Condition-based monitoring (CBM) is essential for maintaining high machine uptime in industrial settings. While existing CBM solutions effectively use time-series data (e.g., thermal, vibration, amperage, etc.), these can be enhanced with LLMs to integrate domain knowledge and generate interpretable summaries. However, LLMs often incur higher latency and cost than traditional methods. We thus propose LEAD (LLM Enabled Anomaly Detection), a two-stage framework. The first stage acts as a screening step, detecting soft anomalies using statistical methods. The second stage leverages LLMs and batches multiple time series in the same prompt to generate final anomalies. We show that the combination of statistical filtering and batching leads to a more efficient and accurate anomaly detection pipeline. Applying LEAD to industrial motor amperage data at Amazon improves precision from 27% (unsupervised deep learning) and 39% (LLM-only) to 72%, while reducing latency 14 fold and token usage by 12 fold compared to an LLM-only baseline. Lastly, we demonstrate that LEAD's accuracy gains from statistical filtering and batching hold even on public datasets.

1 Introduction

Amazon Fulfillment Centers leverage condition-based monitoring (CBM) technologies (like amperage, vibration, and temperature sensors) to detect potential failures on machinery. Anomaly detection algorithms are key to these CBM technologies as they can flag unusual patterns. Current models within Amazon use statistical, machine learning, or deep learning methods. However, these models don't easily incorporate expert knowledge and their output requires extra processing to be understandable for non-technical users.

Recent advances in Large Language Models (LLMs) have shown promising results in time-series forecasting (Gruver et al., 2024; Su et al., 2024; Ansari et al., 2024) and anomaly detection (Alnegheimish et al., 2024; Dong et al., 2024; Zhou & Yu, 2024; Zhuang et al., 2024) tasks. Particularly, they have been shown to improve the quality of detected anomalies by incorporating domain knowledge and enhancing interpretability (Liu et al., 2024). While these findings are impressive, the high costs and latency associated with LLMs pose challenges for their adoption on large-scale datasets (Su et al., 2024; Alnegheimish et al., 2024; Xu & Ding, 2024). Given the scale of streaming datasets at Amazon, a better framework is needed to leverage the potential of LLMs while balancing cost and latency.

We present a framework that combines lightweight anomaly detection algorithms with LLMs for thorough analysis. Our main contributions in the paper are - 1) develop a two-stage framework for univariate time-series anomaly detection by leveraging simple statistical techniques and zero-shot prompting, 2) extend zero-shot prompting with a batching strategy that reduces the latency of the LLM inference *and* improves accuracy, 3) perform ablation studies and comparison with other baseline models, and 4) show that key benefits from our framework generalize to a public benchmark dataset.

2 Background and Problem definition

Background: Variable Frequency Drives (VFDs) are a key component of equipment at Amazon Fulfillment Centers (FCs). VFDs regulate the amperage drawn by motors and provide important signatures related to the health of the overall asset. While amperage can fluctuate due to variance in throughput of packages and the start-up inertia of motors, non-transient changes in amperage signal potential defects with the asset (see Appendix A.1 for examples).

Problem Statement: Our primary objective is to detect anomalies in streaming amperage data, so that issues can be flagged for technician review daily. To achieve this, we

[1]Reliability Maintenance Engineering, Amazon [2]Department of Statistics, Pennsylvania State University. Correspondence to: Akash Chandrayan <kschn@amazon.com>, Matthew Reimherr <mreimherr@psu.edu>.

Proceedings of the 1st ICML Workshop on Foundation Models for Structured Data, Vancouver, Canada. 2025. Copyright 2025 by the author(s).

evaluate each time series using a rolling window approach. Each window is divided into two segments: a historical segment and a latest segment. This segmentation allows for a statistical comparison between the most recent data and the corresponding historical baseline. Based on this comparison, the latest segment can be flagged for potential anomalies, which is further evaluated using LLM.

Formally, given a univariate time-series, $X_k = \{x_{k,1}, x_{k,2}, x_{k,3}, x_{k,t}, .., x_{k,T}\}$ of data points, we want to determine if the latest P data points, collectively can be considered as an anomaly or not. Here k represents the kth VFD and x_t represents an hourly aggregated value of amperage at hour t. We split the time-series into historical data and the latest data (that needs to be evaluated). We denote this split of X_k into $X_{k,historical}$ and $X_{k,latest}$ such that $X_{k,historical} = \{x_{k,1}, x_{k,2}, x_{k,3}, .., x_{k,W}\}$, where W is the window size for historical data, and $X_{k,latest} = \{x_{k,W+1}, x_{k,W+2}, x_{k,W+3}, .., x_{k,W+P}\}$ represents the period of interest. The anomaly detection task is now to determine if $X_{k,latest}$ is anomalous given $X_{k,historical}$. For our use-case, as this is streaming data, the anomaly evaluation needs to happen daily for every time-stream X_k with a rolling window of eight days with the latest one day ($P = 24$) to be tested against rest of the historical seven days ($W = 7 \times 24$).

3 LLM Enabled Anomaly Detection (LEAD) Framework

Unlike time-series forecasting where every point requires prediction, anomaly detection focuses on irregularities, with most data being normal which does not need a thorough analysis using an LLM. We thus develop a two-stage framework (Figure 1) where the first stage filters the data using statistical anomaly detection methods and a second stage that leverages output from first stage and passes to an LLM for anomaly detection. The final output are anomalies detected via the LLM along with explanations for anomalies in an easily readable free-text format.

Stage 1: Statistical Filtering: The purpose of this stage is as a screening step to reduce the data load that passes to LLMs for processing, while retaining most potential anomalies. To keep the first stage as simple as possible, we only leverage statistical approaches in this framework.

At this stage, we compare $X_{k,historical}$ and $X_{k,latest}$ using non-parametric methods like Wasserstein Distance, suitable for multimodal real-world data. These tests are sensitive to the scale of the data, so appropriate scaling is crucial to get normalized score for all k time-series. We have employed max normalization ($x'_{k,i} = x_{k,i}/max(X_k)$), which preserves the relative gaps between data points across different time-series. Most of the non-zero amperage data lacks significant variations, making standard deviation-based normalization inappropriate because it could exaggerate differences. Similarly, min-max normalization distorts the data gaps by forcing it into the [0,1] range, potentially inflating the gaps, which is not desirable for our use-case.

The output of Stage 1 is a simple binary classification based on rules of the statistical criterion chosen (Wasserstein Distance in this case):

$$y_k = \begin{cases} 1 & \text{if } W_1(X'_{k,\text{historical}}, X'_{k,\text{latest}}) > \epsilon \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } k \in \{1, \ldots, N\} \tag{1}$$

where N is the number of time-series that need to be processed. In our context, this is number of VFDs on a site. As data is normalized, a threshold ϵ can be set as a percentile (p) of W_1 across all VFDs for a given window if large number of time-series are present (our case).

Only M time-series that have y_k= 1 (considered as soft anomalies) will be passed to the second stage for final anomaly detection by the LLM. While we have chosen Wasserstein Distance as the statistical filter, any suitable choice of statistical measure that suits the data and anomalies can be chosen.

Stage 2: LLM Anomaly Detection with Batching : This stage uses LLMs for zero-shot prompting, following a structure similar to that described in (Alnegheimish et al., 2024; Liu et al., 2024). We designed the prompt as shown in Figure 6 that has task, data description, judgment rules based on domain knowledge and specific instruction for the LLM to follow. All the time-series that have been classified as soft anomalies from Stage 1 are passed to the LLM using this prompt for further evaluation. We prompt using raw time-series data without any normalization to preserve semantic meaning and discuss this aspect in more detail in Appendix A.2.

Batching Time-series while Prompting: To optimize the latency of the pipeline, we batch B time-series for anomaly detection in a single prompt. While naive batching degrades performance on natural language tasks (Cheng et al., 2023), our studies show that on time-series anomaly detection tasks, even naive batching improves results across multiple LLM providers. We explain this observation and provide a theoretical intuition through an illustrative study using synthetic data across four LLMs (Appendix B).

4 Experiments on Amperage Data from VFD

4.1 Data Description and Evaluation Criteria

We selected two Amazon FC sites for backtesting, and analyzed hourly aggregated amperage data from 1,241 VFDs

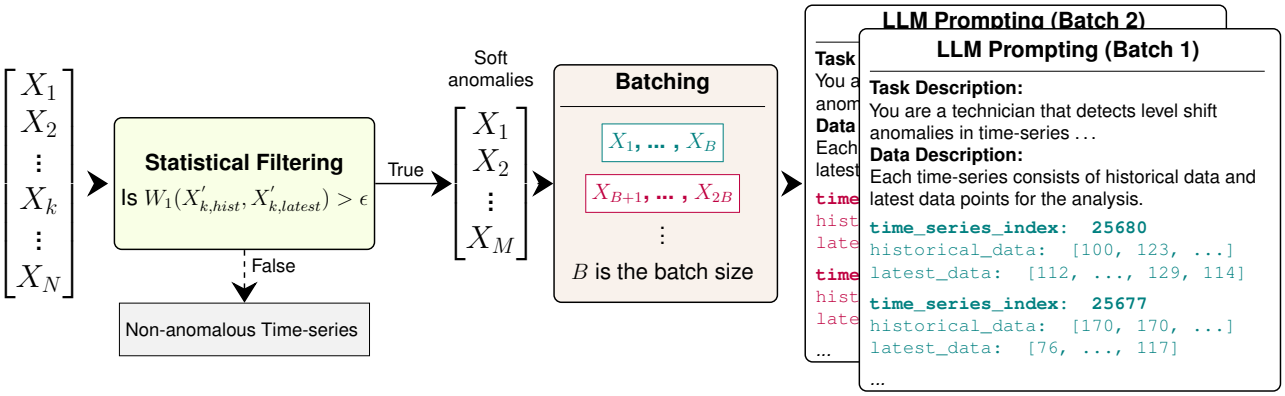


Figure (1) Overall LEAD Framework

containing 893K data points. As no labeled anomalies are present for amperage data, we utilize historical work orders that denote breakdowns and corrective actions taken by site as a proxy for labels. We evaluate all models in their ability to detect anomalies in amperage within seven days of the work performed by technicians. We describe calculation of Precision (P_{det}) and Recall (R_{det}) in more detail in Appendix A.3. Since we want to minimize unnecessary maintenance actions due to false alarms, we prioritize precision over recall and use $F_{0.5}$ as our primary metric (similar to (Luković et al., 2025; Hundman et al., 2018)). Settings for individual hyper-parameters is detailed in Appendix A.4.

4.2 Results

In Table 1, we benchmark our framework against common anomaly detection models. LEAD model shows the best overall performance on both sites. Advanced methods (Appendix A.5) do not provide high precision on VFD data as they require post-processing and domain knowledge to filter trivial anomalies which is already incorporated in LEAD framework's prompt 6. Secondly, we compare LEAD framework ($B = 15, p = 0.75$) with LEAD's Stage 1 ($p = 0.75$) and Stage 2 models ($B = 15, p = 0$). Note that using statistics-only model (Stage 1) or LLM-only model (Stage 2) has a much lower precision compared to both DBSCAN-OCSVM model and LEAD. Our findings corroborate the high false positive rates observed in LLM-only models in (Alnegheimish et al., 2024). This shows that filtering trivial anomalies through Stage 1 of LEAD helps improve the precision significantly and provide better $F_{0.5}$ score. Along with better accuracy, LEAD also has 12 times lower token usage and is 14 times faster compared to LLM-only models without batching as shown in Appendix A.7.

Ablation Studies: We conducted several ablation studies on Site 1's data as shown in Appendix A.8 and highlight two studies in the next section.

Table (1) Performance of various baseline models

Model / Framework	Site 1			Site 2		
	P_{det}	R_{det}	$F_{0.5}$	P_{det}	R_{det}	$F_{0.5}$
LEAD	**0.72**	0.14	**0.39**	**0.75**	0.25	**0.54**
LEAD Stage 1[1]	0.14	**0.37**	0.16	0.29	**0.76**	0.33
LEAD Stage 2[2]	0.39	0.25	0.35	0.51	0.45	0.49
DBSCAN-OCSVM[3]	0.67	0.02	0.09	0.63	0.06	0.37
TranAD	0.20	0.17	0.19	0.39	0.24	0.34
TadGAN	0.18	0.26	0.19	0.40	0.35	0.39
LSTM AE	0.27	0.09	0.19	0.34	0.13	0.25
Conv AE	0.24	0.09	0.18	0.29	0.21	0.27
Feedforward AE	0.23	0.10	0.18	0.36	0.23	0.32
VRNN	0.27	0.10	0.20	0.33	0.22	0.33
Supervised NN	0.27	0.10	0.20	0.39	0.23	0.34

[1] Statistics-only model utilizing Wasserstein Distance
[2] LLM-only model utilizing zero-shot prompting
[3] Custom model optimized for Precision on Amperage Data

4.2.1 Impact of Batching on $F_{0.5}$ and Latency

As shown in Figure 2, an increase in the batch size improves the performance of LLMs. Further, as LLM processes multiple time-series in the same prompt, latency is significantly reduced (6x). Batching thus has dual positive effects: reducing latency and improving accuracy. We believe that having multiple time-series in its context helps the LLM distinguish anomalous patterns better than providing a stand-alone time-series.

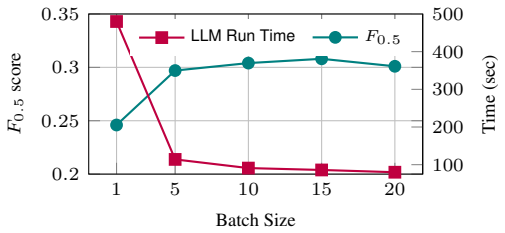


Figure (2) Impact of Batch Size on $F_{0.5}$ score and LLM Inference Time per window. $p = 0.85$

4.2.2 Impact of Shuffling and Voting

We have observed that LLM can reveal slightly different anomalies for a given prompt. This can signal which data points LLM is confident in recognizing as anomalies. We use a strategy of prompting in multiple rounds (r) with shuffled time-series inputs, classifying time-series as anomalous only if it meets a minimum vote threshold (v). Table 2 shows that Precision improves with increasing v for $r = 6$, while $F_{0.5}$ scores remain mostly consistent (except when $r = v$). This allows for setting precision thresholds and choosing r and v based on business needs while maintaining $F_{0.5}$ score. The full table with 1-6 shuffling rounds is in Appendix A.8.4.

Table (2) Impact of v with $r = 6$ ($B = 10$, $p = 0.85$)

	Minimum Votes (v)					
Metric	1	2	3	4	5	6
Precision	0.52	0.64	0.69	0.75	0.77	0.78
Recall	0.11	0.10	0.09	0.09	0.08	0.06
$F_{0.5}$	0.30	0.31	0.30	0.30	0.28	0.23

5 Case Study on Public Data

The LEAD Framework was designed for anomaly detection on streaming sensor data from Amazon's FCs. To understand applicability of our framework more broadly, we tested the LEAD framework on two public datasets from NASA with known ground truth anomaly ranges (Hundman et al., 2018). The data includes two sub-datasets: SMAP and MSL. It contains 82 time-series and have 105 anomalies including contextual or point anomalies. Experimental settings for this study are detailed in Appendix C.1.

5.1 Results

As shown in Figure 3, having statistical filtering through Stage 1 of LEAD ($\epsilon = 15$), improves performance compared to an LLM-only model ($\epsilon = 0$). Similarly, batching through Stage 2 improves performance, further highlighting that LEAD works as expected even on public dataset.

Comparison with State-of-the-art models : Multiple benchmarks exists for time-series anomaly detection task on NASA's SMAP and MSL data. We reproduce the benchmark study done in (Alnegheimish et al., 2024) and compare it with our results in Appendix 13. Note that to align with benchmark study we use the same methodology of F_1 score calculation. LEAD performs reasonably well given minimal parameter optimization and prompt engineering as it is able to beat or come close to 8 and 6 baselines (out of 11) for SMAP and MSL data respectively. This can be explained by the higher gains achieved from batching in SMAP data (0.46 → 0.62) (as SMAP has 55 time-series vs 27 for MSL) compared to MSL data (0.4 → 0.47) as shown in Figure 3.

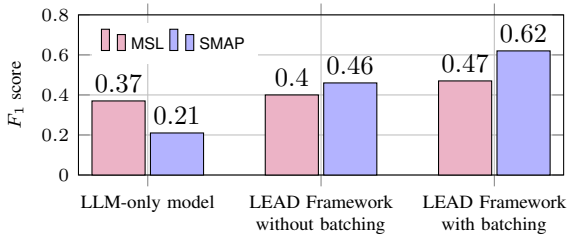
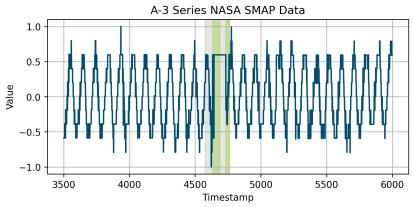
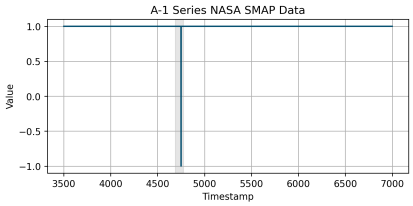


Figure (3) Comparison of LLM-only model ($\epsilon = 0, B = 15$), LEAD Framework without batching ($\epsilon = 15, B = 1$) and LEAD Framework with batching ($\epsilon = 15, B = 10$)

Success and Failure Cases on NASA Data: In Figure 4, we show that while LEAD was successful at detecting contextual anomalies, it missed point anomalies due to Stage 1's Wasserstein Distance filtering, which is not very sensitive to one-off outliers (hence used for our application on VFD data). The framework could be enhanced by incorporating additional statistical measures like Gaussian Mixture Models or Control Charts to better detect such anomalies.



((a)) LEAD detects contextual anomaly



((b)) Point anomaly was undetected by LEAD

Figure (4) Sample success and failure on NASA Data

6 Conclusion and Future Work

In this paper, we propose LEAD, a framework for detecting anomalies in univariate time-series using a hybrid approach that combines statistical measures and LLMs. Through ablation studies on proprietary Amazon and public datasets, we show that LEAD improves both accuracy and efficiency over LLM-only baselines. These improvements stem from both statistical filtering in Stage 1 to identify soft anomalies and batching multiple time-series in Stage 2. Future research can explore more advanced batching strategies based on Stage 1 anomaly scores (derived from statistical measures) and extend the framework to multivariate time series, where batching occurs naturally due to the data structure.

References

Alnegheimish, S., Nguyen, L., Berti-Equille, L., and Veeramachaneni, K. Large language models can be zero-shot anomaly detectors for time series?, 2024. URL https://arxiv.org/abs/2405.14755.

Ansari, A. F., Stella, L., Turkmen, C., Zhang, X., Mercado, P., Shen, H., Shchur, O., Rangapuram, S. S., Arango, S. P., Kapoor, S., Zschiegner, J., Maddix, D. C., Wang, H., Mahoney, M. W., Torkkola, K., Wilson, A. G., Bohlke-Schneider, M., and Wang, Y. Chronos: Learning the language of time series, 2024. URL https://arxiv.org/abs/2403.07815.

Cheng, Z., Kasai, J., and Yu, T. Batch prompting: Efficient inference with large language model apis, 2023. URL https://arxiv.org/abs/2301.08721.

Dong, M., Huang, H., and Cao, L. Can LLMs serve as time series anomaly detectors?, 2024. URL https://arxiv.org/abs/2408.03475.

Geiger, A., Liu, D., Alnegheimish, S., Cuesta-Infante, A., and Veeramachaneni, K. Tadgan: Time series anomaly detection using generative adversarial networks. *2020 IEEE International Conference on Big Data (Big Data)*, pp. 33–43, 2020. doi: 10.1109/BigData50022.2020.9378139.

Gruver, N., Finzi, M., Qiu, S., and Wilson, A. G. Large language models are zero-shot time series forecasters, 2024. URL https://arxiv.org/abs/2310.07820.

Hong, G. and Suh, D. Supervised-learning-based intelligent fault diagnosis for mechanical equipment. *IEEE Access*, 9:116147–116162, 2021. doi: 10.1109/ACCESS.2021.3104189.

Howard, J. and Gugger, S. Fastai: A layered api for deep learning. *Information*, 11(2):108, February 2020. ISSN 2078-2489. doi: 10.3390/info11020108. URL http://dx.doi.org/10.3390/info11020108.

Hundman, K., Constantinou, V., Laporte, C., Colwell, I., and Soderstrom, T. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '18, pp. 387–395. ACM, July 2018. doi: 10.1145/3219819.3219845. URL http://dx.doi.org/10.1145/3219819.3219845.

Liu, J., Zhang, C., Qian, J., Ma, M., Qin, S., Bansal, C., Lin, Q., Rajmohan, S., and Zhang, D. Large language models can deliver accurate and interpretable time series anomaly detection, 2024. URL https://arxiv.org/abs/2405.15370.

Luković, V., Jovanović, , Đurašević Pešović, S., Pešović, U., and Đorđević, B. Solid-State Drive Failure Prediction Using Anomaly Detection. *Electronics*, 14(7): 1433, April 2025. ISSN 2079-9292. doi: 10.3390/electronics14071433. URL https://www.mdpi.com/2079-9292/14/7/1433.

Ren, H., Xu, B., Wang, Y., Yi, C., Huang, C., Kou, X., Xing, T., Yang, M., Tong, J., and Zhang, Q. Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery ; Data Mining*, KDD '19, pp. 3009–3017. ACM, July 2019. doi: 10.1145/3292500.3330680. URL http://dx.doi.org/10.1145/3292500.3330680.

Su, J., Jiang, C., Jin, X., Qiao, Y., Xiao, T., Ma, H., Wei, R., Jing, Z., Xu, J., and Lin, J. Large language models for forecasting and anomaly detection: A systematic literature review, 2024. URL https://arxiv.org/abs/2402.10350.

Sørbø, S. and Ruocco, M. Navigating the metric maze: A taxonomy of evaluation metrics for anomaly detection in time series, 2023.

Tatbul, N., Lee, T. J., Zdonik, S., Alam, M., and Gottschlich, J. Precision and recall for time series, 2019.

Tuli, S., Casale, G., and Jennings, N. Tranad: Deep transformer networks for anomaly detection in multivariate time series data. *Proc. VLDB Endow.*, 15:1201–1214, 2022. doi: 10.14778/3514061.3514067.

Xu, R. and Ding, K. Large language models for anomaly and out-of-distribution detection: A survey, 2024. URL https://arxiv.org/abs/2409.01980.

Zamanzadeh Darban, Z., Webb, G. I., Pan, S., Aggarwal, C., and Salehi, M. Deep learning for time series anomaly detection: A survey. *ACM Computing Surveys*, 57(1): 1–42, October 2024. ISSN 1557-7341. doi: 10.1145/3691338. URL http://dx.doi.org/10.1145/3691338.

Zhang, Y., Chen, Y., Wang, J., and Pan, Z. Unsupervised deep anomaly detection for multi-sensor time-series signals, 2021. URL https://arxiv.org/abs/2107.12626.

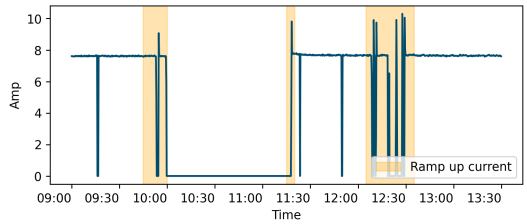
Zhou, Z. and Yu, R. Can llms understand time series anomalies?, 2024. URL https://arxiv.org/abs/2410.05440.

Zhuang, J., Yan, L., Zhang, Z., Wang, R., Zhang, J., and Gu, Y. See it, think it, sorted: Large multimodal models are few-shot time series anomaly analyzers, 2024. URL https://arxiv.org/abs/2411.02465.

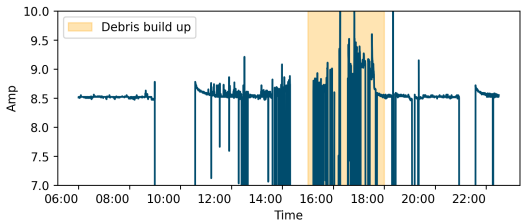
A Supplementary material for Case Study on Amperage Data

A.1 Transient and non-transient anomalies

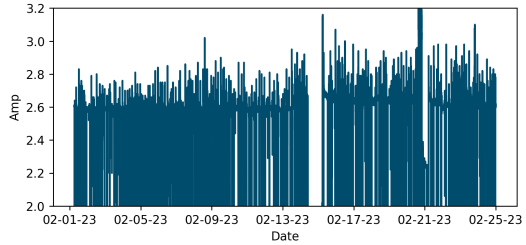
While VFDs are expected to draw a constant amperage if package throughput is the same, in practice regular fluctuations (transient-anomalies) can be observed in VFD data which could be due to multiple reasons including changing throughput, ramp up torque needed to start a motor, or overcoming friction that builds up during a jam when packages get stuck in equipment – see Figure 5(a), 5(b) for examples. Non-transient anomalies on the other hand could be hidden under these normal operating noise and need to be isolated through data processing and anomaly detection on specific patterns. (Figure 5(c), 5(d)).



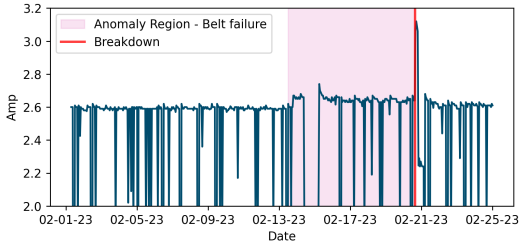
((a)) Ramp up current during start of the motor leads to spikes that are normal



((b)) Transient rise in amperage observed during debris build up



((c)) Anomaly not visible due to intra-day noise



((d)) Anomalous amperage visible after data aggregation

Figure (5) Intra-day transient fluctuations compared to non-transient issues

A.2 Prompt and time-series representation used for case study on Amperage Data

Prompt template used for zero-shot LLM anomaly detection used in Stage 2 for the LEAD framework is shown in Figure 6. In Table 3 we show impact of removing Level of change and Steps components of the prompt. The analysis shows that these components provide necessary domain knowledge and LLM based model would lose significantly on precision without such information in the prompt.

Time-Series Representation in the Prompt: Although normalization was necessary in Stage 1, we do not pass normalized data to the prompt, as it would strip semantic meaning and limit the LLM's ability to leverage domain knowledge or generate technician-friendly anomaly explanations. Instead, we use indices from Stage 1 and format raw data as arrays for LLM input. In our case study, raw amperage data from VFDs—already in integers and scaled by 100 (i.e., 100 = 1 amp)—requires no further quantization. For time-series with fractional values, however, quantization is needed. In such cases, scaling and truncating decimals, as proposed in (Gruver et al., 2024), is preferred. We include the scaling factor in the prompt to retain semantic meaning for interpretability.

Prompt for VFD Amperage Data

Task Description: You are a technician that detects level shift anomalies in time-series data from sensors in industrial setting. You can analyze multiple independent time-series in batch.
Data Description: Each time-series is a sensor reading sequence coming from motor's amperage captured from the VFDs. Each value is for one hour of data. The data is expected to have fluctuations and lots of noise. The values are in amperage but multiplied by 100. So, 104 mean 1.04 Amps.
Judgment Rules:
1) A data point is anomaly if it deviates by at least 10 points or 0.1 amps compared to historical range(s). Note that historical data may be multimodal.
2) Majority of data must be anomalous for the whole day to be classified as anomaly. Point anomalies do not make whole of the latest data anomaly.
3) Whole day anomalies are extremely rare and only happen when there is a significant change and should be identified with absolute certainty.
Level of change: 1) 0-0.1 amps (Not an Anomaly) 2) 0.1-0.5 amps 3) 0.5-1.0 amps 4) >1.0 amps
Strict steps to follow:
1) Analyze each time-series independently for anomalies in the `latest_data` array relative to the `historical_data` array.
2) Count the number of anomaly data points in `latest_data` array
3) Classify `latest_data` as anomalous only when anomaly data points are significantly different from historical pattern and show a clear shift in level on a persistent basis
4) Re-assess the classification of `latest_data` as anomaly. Ensure that you are not calling out the `latest_data` as anomaly because of few point anomalies. There must be a persistent level shift.
5) Provide explanation only when anomaly is detected.
6) In LLM explanation, remember to convert values from raw data to amps by dividing by 100. So, for example, 204 becomes 2.04 amps
7) For each time-series, provide the output in strict JSON format with the following structure:

```
"[
    {
    "time_series_index": <index>,
    "anomaly": "Yes/No",
    "anomaly_type": "Level Shift Up or Level Shift Down",
    "anomaly_explanation": "<provide quantitative explanation>",
    "level of change":  <Based on buckets above>
    }
]"
```

Strictly return a valid JSON list of objects and no additional content or explanation outside JSON.

Figure (6) Prompt for zero-shot anomaly detection on VFD Data

Table (3) Performance Comparison of Different Prompt Versions (Site 1, $p = 0.85, B = 15$)

Prompt	$Precision_{detection}$	$Recall_{detection}$	$F_{0.5}$
Full Prompt	0.70	0.095	0.308
Prompt with No Level Change Information	0.61	0.081	0.265
Prompt with No Level Change and Step wise process	0.47	0.089	0.253

A.3 Precision and Recall on work order Data

We lack labeled anomalies and use work orders, which are primarily of three types: breakdown (BRKD), corrective maintenance (CM), and follow-up preventive maintenance (FPM) as proxy labels. Our model aims to detect anomalies before breakdowns, but CM and FPM can alter the asset's properties, affecting amperage. For example, a belt change during FPM may cause slight (but permanent) variation in amperage. These events thus become potential detection candidates for model evaluation as they represent non-transient anomalies. In practice, however, we suppress detections during CM or FPM, recognizing them as proactive actions taken by technicians.

Evaluation Criteria: Defining precision and recall for anomaly detection in time-series is complex due to differing interpretations of successful detection. Point-based precision is less relevant, as we expect lead time before breakdowns, favoring range-based metrics. Studies (Sørbø & Ruocco, 2023; Tatbul et al., 2019) address these challenges, but we lack methods to categorize predicted anomalies into ranges or predefined real anomaly ranges. We therefore adapt range-based precision and recall by using work order data.

Precision is defined as the ratio of successfully detected anomalies within *h* days before a work order to the total number of anomalies generated by the model. We introduce two settings: Predictive, rewarding only pre-work order detections, and Detection, rewarding any detection within *h* days around the work order. Due to the higher frequency of CM/FPM compared to BRKD work orders, Precision/Recall is naturally higher in the Detection setting.

Mathematically, given N_p as total anomalies flagged by the model, we define $N_{h,p,before}$ as the number of anomalies flagged in h-days window before a work order. We can then define precision as:

$$Precision_{predictive} = \frac{N_{h-before,p}}{N_p}, \quad Precision_{detection} = \frac{N_{h-any,p}}{N_p}$$

Recall is defined as the ratio of successful work order events identified within h-days window of the work order to total work order events (N_{wo}). Here as well we define two variants as below:

$$Recall_{predictive} = \frac{N_{h-before,wo}}{N_{wo}}, \quad Recall_{detection} = \frac{N_{h-any,wo}}{N_{wo}}$$

We set $h = 7$ days across evaluations on VFD Data.

A.4 Hyper-parameters for the framework

Below we list the key hyper-parameters for the LEAD Framework. The ranges mentioned below were used for the case study on VFD Data.

- ϵ - We set this parameter based on p^{th} percentile of Wasserstein Distance (W_1) for the data for each rolling window. This ensures that top-p fraction of data as judged by the Stage 1 statistical criteria is passed to the LLM. We vary p in our experiments from 0.65 to 0.95
- **Window,** W - We set window, $W = 7 \times 24$ as look back period for historical data.
- **Period,** P - We set window, $P = 24$ for the latest period of interest.
- **Batch Size,** B - Number of inference time-series chained in one prompt. We vary this between 1, 5, 10, 15 and 20.
- **LLM Settings** - To minimize the variability in output we set $temperature = 0, top_k = 1, top_p = 1$ across all LLMs

Table (4) Performance metrics for work order types across sites

Site	Work order type	Prediction			Detection		
		Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
Site 1	Breakdown	0.08	0.08	0.08	0.12	0.11	0.12
	Total	0.29	0.06	0.16	0.72	0.14	0.39
Site 2	Breakdown	0.03	0.17	0.07	0.12	0.17	0.13
	Total	0.36	0.12	0.26	0.75	0.25	0.54

A.5 Baseline Models

Unsupervised Models: We tested multiple deep learning based methods - LSTM, Convolutional, and Feedforward autoencoders as well as Variational Recurrent Neural Networks (VRNNs) (Zhang et al., 2021; Zamanzadeh Darban et al.,

2024; Ren et al., 2019). These unsupervised models learn to reconstruct normal data patterns and identify anomalies through high reconstruction errors. VRNNs combine the benefits of variational methods with recurrent architectures to model both uncertainty and temporal dependencies. LSTM-based autoencoders capture long-term temporal patterns, while convolutional architectures efficiently extract local features. We further expanded our investigation to include Transformers for Anomaly Detection (TranAD) and Time Series Anomaly Detection using Generative Adversarial Networks (TadGAN). TranAD ((Tuli et al., 2022)) leverages the self-attention mechanism of transformers to capture complex temporal dependencies and global patterns in the data without the sequential constraints of recurrent architectures. TadGAN ((Geiger et al., 2020)) combines the power of adversarial training with autoencoder architectures, where the discriminator learns to distinguish between normal and anomalous patterns while the generator aims to reconstruct normal behavior. This adversarial approach provides a more robust anomaly detection framework by learning both reconstruction and discrimination-based features.

Results show that all the unsupervised models do not provide a good out-of-the-box precision and potentially require additional post-processing based on expert knowledge and to filter out false positives.

Semi-supervised: Our custom model (DBSCAN-OCSVM) used for amperage anomaly detection at Amazon is a semi-supervised model and has been optimized for precision through additional post-processing. The model first utilizes DBSCAN to reduce the noise in the data, followed by a one class SVM to construct the decision boundary for flagging anomalies. Additional post-processing parameters were introduced in the pipeline that were derived from the OCSVM model output. The hyper parameters of the SVM, DBSCAN and post-processing parameters were tuned based on historic anomalies to optimize for precision.

Supervised Model: We also explored supervised deep learning ensemble models (Howard & Gugger, 2020), (Hong & Suh, 2021) for anomaly detection, using known breakdown occurrences as labels. This approach aimed to directly learn patterns associated with breakdowns. However, the highly imbalanced nature of the dataset, with rare breakdown events, posed significant challenges. Moreover, the limited availability of accurate labels for anomalies restricted the model's ability to generalize to unseen failure modes.

A.6 True Positive, False Positive and False Negative Cases

In this section, we present four illustrative cases that highlight true positive, false positive and false negative classifications of anomalies detected by the LLM model, along with the generated explanations.

Case 1: An anomaly was detected during follow-up preventive maintenance (FPM). Notably, no signal was present in the data before the FPM, and the model accurately detected the change on the day the maintenance was performed.

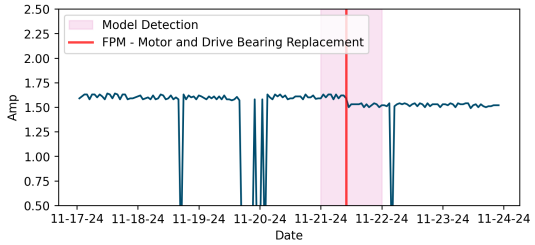
Case 2: The model identified an anomaly five days before an actual breakdown, as indicated by a clear spike in amperage, representing a true positive detection.

Case 3: A level shift in amperage data was detected, but no corresponding work order was available. This scenario might indicate that either the work was performed but not logged, or the machine's increased throughput led to higher amperage. These cases are classified as false positives since the reason for the change cannot be verified in back-testing and requires live monitoring.

Case 4: A false negative example occurred where a Photo-eye issue caused a breakdown, yet no signal was present in the amperage data.

All LLM-generated explanations provide concise, human-readable summaries, and these will integrate into work orders when the model is deployed. The LLM effectively converted raw data (multiplied by 100 and in integer format) into real amperage values, eliminating the need for further post-processing.

Case 1 - VFD 26388

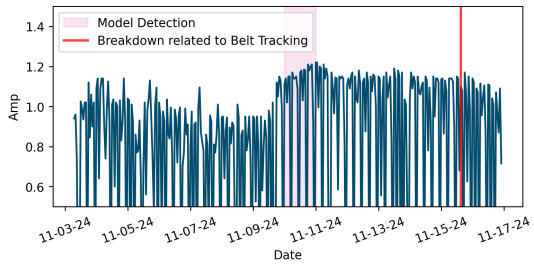


LLM Anomaly Explanation

The data has shifted from a historical range of 1.58–1.65 amps to a persistent lower level of 1.50–1.54 amps. This represents a clear and maintained downward level shift of approximately 0.1 amps.

Classification: True Positive

Case 2 - VFD 25680

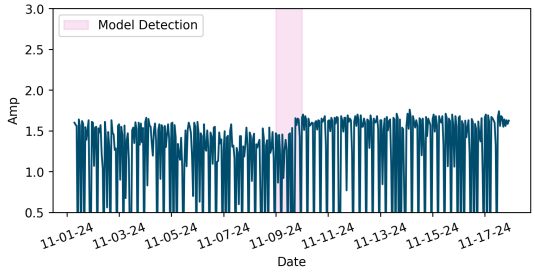


LLM Anomaly Explanation

The latest data shows a persistent upward shift to around 1.15–1.25 amps compared to historical levels which were typically between 0.85–1.10 amps.

Classification: True Positive

Case 3 - VFD 25682

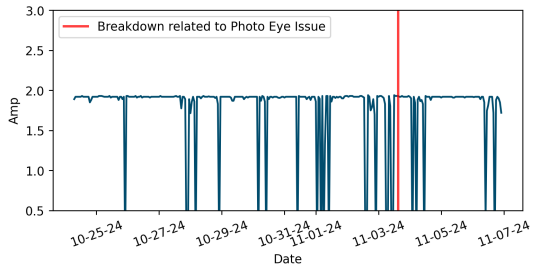


LLM Anomaly Explanation

The latest data shows a sustained shift up to around 1.65–1.70 amps compared to historical range of 1.35–1.60 amps.

Classification: False Positive

Case 4 - VFD 86610



Classification: False Negative

A.7 Detailed Cost and Latency Analysis

In Table 5, we show a comparison of total token usage count and latency of our proposed framework and compare it with the zero-shot LLM-only models, both with and without batching. All estimates for the analysis are based on token usage and latency for Claude 3.5 Sonnet v2 used via Bedrock. The analysis shows that our proposed framework is 14 times faster compared to the LLM-only model (without batching) and achieves a runtime that is comparable to current models in production. Similarly, the LLM token usage in LEAD Framework is 12 times lower compared to the LLM-only model without batching and 7 times lower compared to the LLM-only model with batching. While the numbers can vary based on the choice of parameters B and p, the reduction in token usage and latency would still be significant for any reasonable choice of these parameters that is best for accuracy.

Table (5) Comparison of LLM-only Models and the LEAD Framework

Metric	LLM-only Model without Batching ($p = 0$, $B = 1$)	LLM-only Model with Batching, ($p = 0$, $B = 15$)	LEAD Framework ($p = 0.85$, $B = 15$)
Input tokens per inference	1200	9644	9644
Output tokens per inference	63	836	836
No of inferences per day	1241	82.7	12.4
Total token usage	1,567,383	867,045	130,057
LLM Latency per site/day	21 min	11 min	1 min 26 sec

A.8 Ablation Studies on Amperage Data for Site 1

All ablation studies have been performed on *Claude 3.5 Sonnet v2* for its consistent results (details on comparison with other LLM models is presented in Appendix A.8.3).

A.8.1 Impact of Stage 1 parameter

In Table 6, we show how the results would have been under different choices of threshold ϵ for Stage 1. We set this parameter by varying the percentile p of the data that is passed to LLM. We notice that as p increases to the 90th percentile, we see a drop in both precision and recall. Furthermore, note that setting $p = 0$ is equivalent to an LLM-only model.

Table (6) Performance metrics for different values of p for same Batch Size ($B = 15$)

p	$Precision_{detection}$	$Recall_{detection}$	$F_{0.5}$
0.65	0.606	0.133	0.355
0.70	0.609	**0.151**	0.379
0.75	0.715	0.141	**0.394**
0.80	**0.725**	0.100	0.322
0.85	0.700	0.095	0.308
0.90	0.588	0.077	0.252
0.95	0.529	0.036	0.141

A.8.2 Detailed results for Ablation Studies on Batching

As shown in Table 7, an increase in the batch size improves the performance of LLMs by increasing the precision significantly. As an LLM processes multiple independent time-series in the same prompt, latency is also significantly reduced (6x). Batching thus has dual positive effects: reducing latency and improving accuracy.

Table (7) Performance metrics for different values of Batch Size (B), $p = 0.85$

B	$Precision_{detection}$	$Recall_{detection}$	$F_{0.5}$	Latency (sec)
1	0.677	0.069	0.246	480
5	0.633	0.095	0.297	114
10	0.757	0.089	0.304	91
15	0.700	0.095	0.308	86
20	0.693	0.092	0.301	80

A.8.3 PERFORMANCE UNDER DIFFERENT LLM MODELS

We tested several LLM models available in AWS Bedrock and found that *Claude 3.5 Sonnet v2* (`anthropic.claude-3-5-sonnet-20241022-v2:0`) delivered the best performance. Unlike Cohere, Mistral, and Llama models, which frequently encountered JSON decoding errors, the Claude models performed without such issues. Moreover, the Claude models consistently adhered to the instructions provided in the prompt. Detailed results are shown in Table 8.

Table (8) Performance Metrics of LLM Models

LLM Model	$P_{detection}$	$R_{detection}$	$F_{0.5Score}$	**Comments**
Claude 3.5 Sonnet v2	0.72	0.14	0.39	Good performance overall. No JSON Decoding Error, model is able to follow instructions and able to do appropriate scaling (by 100) when providing explanations
Claude 3 Sonnet	0.25	0.15	0.22	No JSON decoding error; follows instructions on unit conversion while providing anomaly explanation, however has high false positives.
Cohere Command R+	0.19	0.03	0.09	Frequent JSON decoding error, Model does not follow instructions well and still produces explanation for non-anomalous time-series; provided explanations have incorrect units of numbers
Llama 3.2 11b Instruct	-	-	-	LLM unable to understand instruction. Provides output like - *result = detect_anomaly(26278, historical_data, latest_data)*
Mistral 7b Instruct	-	-	-	Incorrect units and percentage in explanation. Example - *The latest data shows a persistent level shift of approximately 0.134 amps (13.4%) compared to the historical data*

A.8.4 IMPACT OF REPEATED PROMPTING BY USING SHUFFLING AND VOTING

Although the ablation studies above are based on single runs with the same parameters, we have observed that multiple iterations with identical input to the LLM can yield slightly different outputs (Table 9). To leverage this non-homogeneity in output to extract high confidence answers, we adopt a strategy of repeatedly prompting LLMs across multiple round (r) and classify anomalies only when they have been voted as anomalies a minimum of v times. Results are shown in Table 10

Table (9) Performance Metrics for Different Runs, $B = 20, p = 0.85$

Run	$Precision_{detection}$	$Recall_{detection}$	$F_{0.5}$
Run 1	0.693	0.092	0.301
Run 2	0.728	0.105	0.333
Run 3	0.704	0.092	0.303

Table (10) Effect of Shuffle Rounds and Minimum Votes on Precision, Recall, and $F_{0.5}$ ($B = 10, p = 0.85$)

Shuffle Rounds	Precision						Recall						$F_{0.5}$					
	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
1	0.63						0.09						0.28					
2	0.61	0.77					0.10	0.08					0.30	0.28				
3	0.57	0.73	0.78				0.11	0.09	0.07				0.30	0.29	0.26			
4	0.54	0.70	0.77	0.78			0.11	0.09	0.08	0.07			0.31	0.30	0.29	0.26		
5	0.51	0.65	0.72	0.79	0.82		0.11	0.10	0.09	0.08	0.06		0.29	0.30	0.30	0.29	0.24	
6	0.52	0.64	0.69	0.75	0.77	0.78	0.11	0.10	0.09	0.09	0.08	0.06	0.30	0.31	0.30	0.30	0.28	0.23

B Performance of Batch Prompting across multiple LLM Providers on Synthetic Data

To assess whether batching-related gains are specific to Claude Models, we generated synthetic data and tested it across different LLM providers. The dataset consisted of 500 normal data points followed by 50 points with a synthetically injected anomaly. The first nine series (S1-S9) contained trivial anomalies (e.g., anomalies passing the Stage 1 statistical filter in the LEAD Framework but considered minor), while only S10 featured a major anomaly with a level shift in the seasonal pattern. Please refer to Figure 7 for visualization of the series.

Each model provider was tested in two modes using the prompt in Figure 8 : 1) '**Batch**' where all ten time-series data was passed in the same prompt and LLM outputs a JSON with all time-series in one invoke. 2) '**Single**' where each time-series was passed one by one to the prompt. All LLMs were set to minimum `temperature`, `top_p`, and `top_k` settings to minimize variations in output.

Results in Table 11 show that all the LLMs perform better at detecting major anomaly (and ignoring trivial noise) when data is passed in 'Batch' mode compared to prompting in 'Single' mode. Both, Claude 3.5 Sonnet and Cohere Command R+ perform much better in batch mode at ignoring 'trivial' anomalies. This illustrative experiment shows that batching related accuracy improvements are not particular to any LLM providers but is a better strategy to prompt for anomaly detection tasks across LLMs as they seem to benefit from other time-series in the context. A more detailed discussion is provided in the next section B.1.

B.1 Intuition behind improvement in Precision from Batching

The observation that batching multiple time-series improves precision in LLM-based anomaly detection, as opposed to single time-series prompting, can be supported by considering how LLMs process information and leverage context. The core idea is that a batch provides a richer context that allows the LLM to make more informed and robust judgments, leading to less false positives.

- Let X_k denote the k-th time-series from the M time-series flagged as 'soft anomalies' by Stage 1.
- Let R represent the set of rules and instructions provided in the prompt.
- Let A_k be the event that X_k contains a true anomaly.

LLMs, especially in a zero-shot setting, establish a baseline of 'normal' behavior based on the input data and the provided rules R. When processing a single time-series X_k, the LLM's understanding of normalcy is limited to the characteristics of X_k itself (and its historical data included in the prompt segment for X_k) and its pre-trained knowledge, as guided by R.

Single Series Prompting A variation within X_k might be statistically rare for that specific series but could be a common type of fluctuation or operational mode when considering a broader set of similar entities. If this variation is unusual for X_k in isolation, the LLM might incorrectly flag it as an anomaly, leading to a False Positive (FP).

The probability might be relatively high if X_k exhibits even trivial anomaly compared to historical data:

$$P(\text{LLM flags } X_k \text{ as anomalous} \mid X_k, R, \neg A_k)$$

Batched Series Prompting When a batch of B time-series $X_1, \ldots, X_B$ is provided, the LLM gains access to a richer, contemporaneous sample of behaviors. It can implicitly learn or infer a more robust and representative 'normality distribution' from the batch itself. If several series in the batch exhibit similar types of trivial anomalies (as they have passed Stage 1 statistical filter), the LLM can better calibrate its threshold for what constitutes a deviation significant enough to be an anomaly according to R. A pattern in X_k that might have seemed anomalous in isolation could be contextualized as part of the normal operational variance observed across the batch. Thus, each batch could provide the LLM with multiple examples of both trivial and major anomaly patterns, creating an implicit few-shot learning setup.

The probability, therefore, is likely to be lower than with single prompting for such trivial anomaly cases:

$$P(\text{LLM flags } X_k \text{ as anomalous} \mid X_k, \{X_j\}_{j \neq k, j \in B}, R, \neg A_k)$$

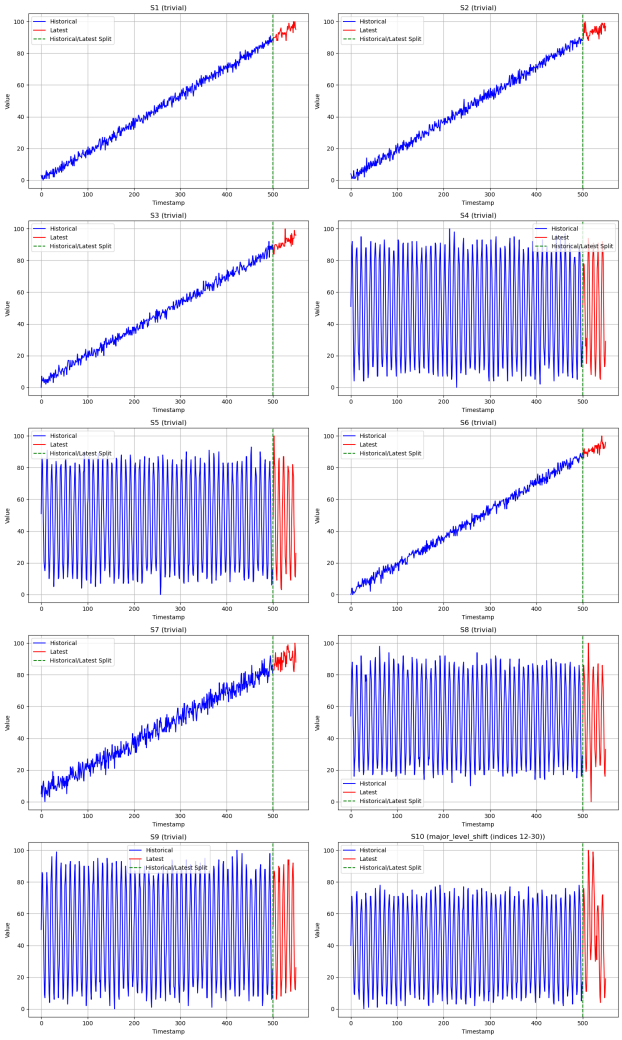


Figure (7) Synthetic data with Normal (blue) range and Anomaly Range (Red). Series S1-S9 have trivial anomalies and Series S10 has a major anomaly.

Table (11) Comparison of Anomaly Detection across Model Providers in 'Batch' and 'Single' mode on Synthetic Data. 'Yes' implies model detected time-series as anomaly and 'No' implies model detected no anomalies. Only S10 has non-trivial anomaly that needs to be detected

Time Series	**Anomaly**	**Maverick 17B**		**Mistral 7B**		**Cohere Command R+**		**Claude 3.5**	
		Batch	**Single**	**Batch**	**Single**	**Batch**	**Single**	**Batch**	**Single**
S1	No	Yes	No	Yes	Yes	No	Yes	No	No
S2	No	Yes	Yes	Yes	Yes	No	Yes	No	Yes
S3	No	Yes	Yes	No	Yes	No	Yes	No	No
S4	No	No	Yes	No	Yes	No	Yes	No	No
S5	No	No	Yes	No	Yes	No	Yes	No	No
S6	No	Yes	Yes	No	Yes	No	Yes	No	Yes
S7	No	Yes	Yes	Yes	Yes	No	Yes	No	No
S8	No	Yes	Yes	No	Yes	No	Yes	No	No
S9	No	No	No	No	Yes	No	Yes	No	Yes
S10	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Anomaly Count	1	7	8	4	10	1	10	1	4

Prompt for Synthetic Data

Task Description: Task Description: You are an expert at anomaly detection in patterns of time-series data from sensors. You can analyse multiple time-series in batch.
Data Description: Each time-series is a sensor reading that has both historical_data and latest_data.
Judgment Rules:
1) 1. A data point is anomaly if it deviates compared to historical pattern. Note that historical data may have seasonal pattern or trends.
2) Anomalies are when the latest data has a different pattern than historical values.
3) Do not call data points as anomalies if they follow the pattern but have slightly different peaks or troughs
4) Do not call indices as anomalies unless there is a clear and significant deviation. Ignore smaller deviations as anomalies are EXTREMELY rare
4) Always give range of anomalies, if there is a single anomaly point, add padding of 5 indices
Strict steps to follow:
1) Analyze each time series independently for anomalies in the `latest_data` array relative to the `historical_data` array.
2) Identify indices in the `latest_data` that might be anomalous (look for both distribution changes and pattern changes)
3) Provide explanation only when anomaly is detected.
4) For each time series, provide the output in strict JSON format with the following structure:

```
"[
    {
        "time_series_index": <index>,
        "anomaly_explanation": "<provide quantitative explanation>",
        "anomaly_indices": [<indices>],
        "anomaly": "Yes/No"
    }
]"
```

Strictly return a valid JSON list of objects and no additional content or explanation outside JSON.

Figure (8) Prompt for zero-shot anomaly detection on Synthetic Dataset

C Supplementary Material for NASA Data

C.1 Experiment Setup

We utilize a very similar setup as used in Amazon's VFD data. For the NASA data, we set the period, P, of analysis to be 70 (based on the downlink frequency) as mentioned in the original paper (Hundman et al., 2018). We set the window, W, to be 700 to maximize historical window context for LLM while minimizing anomalies to fall in the first window (only 1 anomaly out of 105 anomaly ranges occur before index 700). For threshold on Wasserstein distance, we set the value of ϵ to be 15 (see Ablation study in Appendix C.2). All the data was scaled between 0 to 1 using min-max scaling and then quantized to take integer values from 0 to 100. The LLM prompt used for the studies is shown in Figure 9 and time-series from both SMAP and MSL data were combined in the same pool to maximize opportunities for batching. The LLM model used is *Claude 3.5 Sonnet v2*. We use the same evaluation criteria of Precision and Recall as used in the original paper (Hundman et al., 2018) and benchmark study (Alnegheimish et al., 2024), where range-based metrics are calculated based on overlap of ground truth with detected anomalies.

Prompt for NASA Data

Task Description: You are an expert at anomaly detection in patterns of time series data from sensors. You can analyse multiple independent time series in batch.
Data Description: Each time series is a sensor reading from NASA Space rovers on Mars
Judgment Rules:
1) A data point is anomaly if it deviates compared to historical pattern. Note that historical data may have seasonal trends.
2) Anomalies are when the latest data has a different pattern than historical values. 3) Do not call data points as anomalies if they follow the pattern but have slightly different peaks or troughs
4) Do not call indices as anomalies unless there is a clear and significant deviation. Ignore smaller deviations as anomalies are EXTREMELY rare
5) Always give range of anomalies, if there is a single anomaly point, add padding of 5 indices
Strict steps to follow:
1) Analyze each time series independently for anomalies in the `latest_data` array relative to the `historical_data` array.
2) Identify indices in the `latest_data` that might be anomalous (look for both distribution changes and pattern changes)
3) Provide explanation only when anomaly is detected.
4) For each time series, provide the output in strict JSON format with the following structure:

```
"[
    {"time_series_index":<index>,
    "anomaly_explanation": <provide quantitative explanation>,
    "anomaly_indices": [indices],
    "anomaly": "Yes/No"}
]"
```

Strictly return a valid JSON list of objects and no additional content or explanation outside JSON.

Figure (9) Prompt for zero-shot anomaly detection on NASA Dataset

C.2 Ablation study for variations on ϵ on NASA data

In Table 12 we show the variation of F_1 score across NASA datasets at different ϵ. Overall, $\epsilon = 15$ shows the best performance. Figure 10 shows how batching improves performance on SMAP data, particularly by improving the Precision, reinforcing that having multiple time-series in the same prompt help LLMs to ignore trivial anomalies.

Table (12) F_1 Scores for Different ε Values for $B = 10$

ε	MSL	SMAP	Overall Data
5	0.423	0.303	0.341
10	0.426	0.473	0.455
15	0.472	0.618	**0.509**
20	0.450	0.500	0.478
25	0.377	0.387	0.383
30	0.387	0.326	0.351

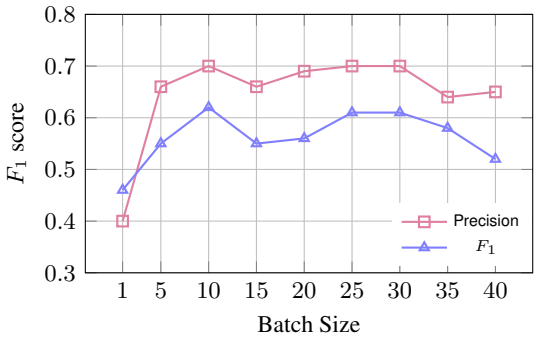


Figure (10) Batching improves F_1 scores for SMAP data primarily through increased Precision

C.3 Comparison of LEAD results on NASA dataset with available baselines

In Table 13 a comparison of LEAD Framework's results is shown with other baseline methods. The numbers for other baseline methods have been reproduced from the study (Alnegheimish et al., 2024)

Table (13) Model Performance (F_1) on NASA Datasets with Difference from LEAD

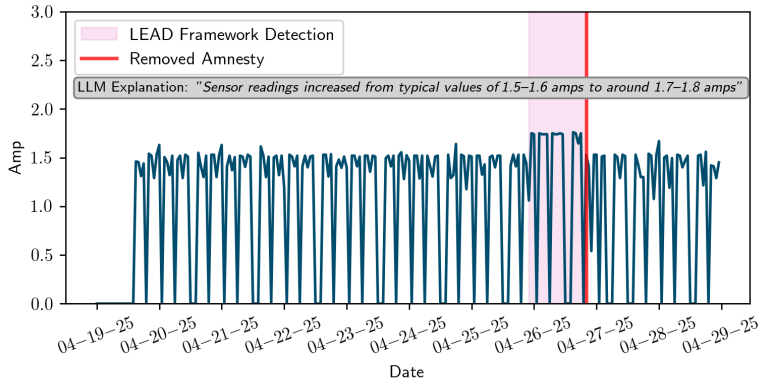
Model	MSL	SMAP	Diff w/ LEAD (MSL)	Diff w/ LEAD (SMAP)
AER	0.587	0.819	-19.6%	-24.5%
LSTM DT	0.471	0.726	0.2%	-14.9%
LEAD (ours)	0.472	0.618	0.0%	0.0%
ARIMA	0.525	0.411	-10.1%	50.4%
Matrix Profile	0.474	0.423	-0.4%	46.1%
TadGAN	0.560	0.605	-15.7%	2.1%
LSTM AE	0.545	0.662	-13.4%	-6.6%
VAE	0.494	0.613	-4.5%	0.8%
AnomalyTransformer	0.400	0.266	18.0%	132.3%
Moving Average	0.171	0.092	176.0%	571.7%
MS Azure	0.051	0.019	825.5%	3152.6%

D Insights from Pilot

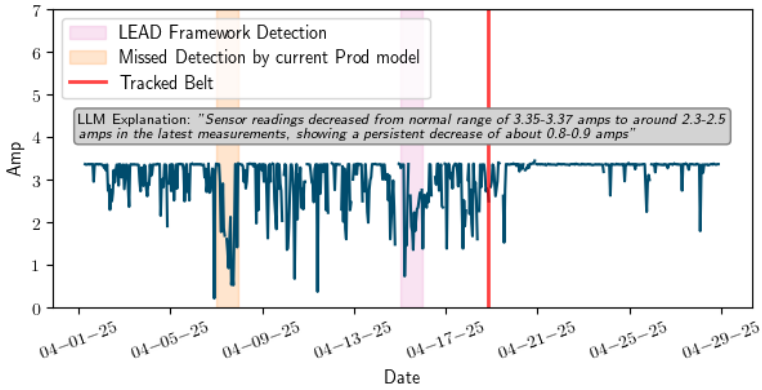
With the improvement in latency made by this framework and better recall than the current production model in back-testing, we piloted LEAD in Week 16 and Week 18 on Site 1 and Site 2 respectively to gather feedback from technicians. While anomaly instances are rare and we are early in our pilot to fully evaluate it, we have already detected six defects through LEAD that were undetected by current production model (Sample detections shown in Figure 11). Preliminary results for Precision are shown in Table 14 and are broadly inline with back-testing performance observed through case study.

Table (14) Performance of LEAD in Pilot

Site	Work orders resolved	True positive work orders	Precision	Period
Site 1	30	24	80.00%	14 Apr – 18 May
Site 2	13	9	69.23%	28 Apr – 18 May



((a)) Amperage rise detection leading to removal of amnesty



((b)) LEAD detected unusual variance in amperage pointing to tracking issue in belt

Figure (11) Sample detection by LEAD in Pilot (that were undetected by current model), please refer to explanations generated by LEAD in gray boxes in the images

E What are suitable use-cases for the LEAD Framework?

From the studies above we can observe that LEAD Framework works best for anomaly detection tasks that have following properties 1) highly imbalanced data so that most trivial time-series can be discarded in Stage 1, 2) have opportunity of parallel processing of large number of time-series (to exploit gains from batching in Stage 2), and 3) require domain level knowledge and out-of-the-box explainability.

The simple design of the framework makes it easy to adapt to different use-cases with minimal changes and achieve comparable results with state-of-the-art techniques. Further, the framework is particularly useful in cold-start problems where not much historical data is present for training deep-learning models.