Anomaly Detection in Multivariate Industrial Signals: LLMs, TSFMs, or Classical Deep Learning

Allen Baranov[1], Sarah Alnegheimish[1], Alfredo Cuesta-Infante[2], Weizhong Yan[3], Masoud Abbaszadeh[3], and Kalyan Veeramachaneni[1]

[1] *Massachusetts Institute of Technology, Cambridge, MA, 02139, USA*
baranov@mit.edu
smish@mit.edu
kalyanv@mit.edu

[2] *Universidad Rey Juan Carlos, Madrid, Spain*
alfredo.cuesta@urjc.es

[3] *GE Vernova Advanced Research, Niskayuna, NY, USA*
yan@gevernova.com
abbaszadeh@gevernova.com

ABSTRACT

Large language models (LLMs) offer several distinctive advantages over other machine learning models. First, they are trained as general-purpose models and are readily available, which eliminates the need for task-specific training and allows them to improve rapidly over time. Second, they can be applied directly, without constructing domain-specific or signal-specific models. Third, they are easy to integrate into existing systems and can be deployed without requiring an additional training step. Finally, they are inherently interactive because users can direct them with natural language. In this paper, we investigate whether LLMs can achieve multivariate anomaly detection. To fully exploit the aforementioned benefits, we define a set of guiding principles (such as avoiding pre-learning or representation learning on the signals) to ensure the LLMs remain general-purpose models. Based on these principles, we then propose several algorithmic approaches for building multivariate anomaly detection pipelines. We compare our approaches with two alternatives: (i) classical deep learning pipelines trained specifically for anomaly detection, and (ii) a foundation-model-based approach, in which domain-specific or general purpose time-series foundation models are trained without explicit supervision for anomaly detection but are then used for this purpose. The comparison highlights trade-offs along three key dimensions: anomaly detection accuracy, computational cost, and the amount of domain knowledge required to develop the pipeline. We evaluate our methods through two case studies. The first uses a benchmarking testbed designed for anomaly detection, while the second examines real-world data from wind turbines with known anomalous events.

1. INTRODUCTION

The focus of this paper is unsupervised time series anomaly detection. In this task, a system or set of equipment is monitored via time series. These time series then undergo retroactive unsupervised processing, which flags unusual patterns and abnormal behaviors. These patterns are investigated by operators or domain experts. The investigations generally lead to one of three outcomes: performing a root cause analysis and undertaking corrective action, cataloging the anomaly for future investigation, or false positive (when the equipment is actually behaving properly and the unusual pattern can be explained). This process has enormous relevance for industrial scenarios and is used to monitor wind turbines, ships, aircraft, and other large equipment.

Over the past decade, deep learning has delivered a significant improvement over traditional threshold-based alarm systems for equipment signal analysis (we describe deep learning based methods in Section 2). Threshold-based methods often have high false positive rates, which can contribute to alarm fatigue among operators. Deep learning methods have demonstrated strong performance in identifying complex patterns in sensor data and have reduced false alarms. However, several challenges limit their deployability:

- **Model Scalability Problem** Scalability remains a ma-

jor constraint. Deep learning models must typically be trained on signals collected from the specific equipment under observation. In many industrial systems, each piece of equipment has hundreds of data-generating sensors. To achieve high precision, it is necessary to select a subset of relevant signals and train models using that subset. Users commonly group signals that are associated with a specific functional subsystem, such as a compressor or electrical unit, and train models on these groups. This specialization allows the model to learn signal interdependencies and to effectively detect anomalous patterns. However, because anomaly detection requires training on each group of signals, the number of models can grow rapidly as the number of possible signal groupings increases.

- **Data Drift and Retraining Problem** As equipment ages, sensor signals may drift due to wear, environmental changes, or operational adjustments, and patterns that once indicated abnormal behavior may gradually become normal. Consequently, models must be retrained at regular intervals to maintain accuracy. Determining appropriate retraining schedules is typically left to operators, which adds operational complexity.
- **Model Deployment and Operationalization Problem** Transitioning real-time anomaly detection models from the training phase to deployment often requires specialized infrastructure, integration with operational systems, and software engineering expertise outside the purview of the model developers and equipment operators. As a result, the deployment process often requires coordination with separate engineering teams, which creates delays and operational overhead.

The case for foundation models. Foundation models rely on the same underlying deep learning technologies as DL models, but have capabilities that help alleviate the challenges described above. A foundation model is trained on a large corpus of historical data. Rather than being optimized for a specific task, it learns general patterns across signals and their conditional relationships within different operational contexts. Because of this broad training, foundation models can often be applied directly in real time without undergoing task-specific training for each piece of equipment or signal group. After learning from a diverse dataset that includes multiple contexts and signal combinations, these models can generalize across different physical environments, equipment configurations, and signal groupings. This reduces the need to build and maintain many specialized models. Foundation models also simplify deployment. Rather than being individually integrated into operational systems, which requires dedicated engineering teams for each model, foundation models are typically exposed through standardized APIs that support multiple downstream tasks. This approach enables broader deployment and helps amortize infrastructure and operational costs. Finally, because foundation models are task-agnostic, their retraining schedules are generally decoupled from task-specific monitoring needs. Instead of retraining numerous specialized models at different intervals, the foundation model can be updated periodically using new data. These updates are constrained primarily by available computational resources rather than by the maintenance requirements of individual monitoring tasks.

Due to these advantages, a number of general-purpose time series foundation models have recently been proposed. Rather than being trained for a specific domain or piece of equipment, a general-purpose foundation model can be used for any time series. A second approach relies on domain-specific foundation models, which are trained for a specific domain (for instance, a specific type of wind turbine). Others propose that large language models, which are trained on language data rather than time series, can be repurposed for time series anomaly detection tasks. In this paper, we begin by defining the specific requirements a foundation model must satisfy in order to effectively alleviate the challenges that come with deep learning models. We then explore these three types of foundation models – general-purpose foundation models, domain-specific foundation models, and LLMs – for the task of unsupervised time series anomaly detection. The comparison is not meant to imply that all methods compete under identical conditions: the classical deep learning pipelines are task-specific anomaly detection systems, while the LLM and TSFM pipelines are evaluated under zero-shot or general-purpose constraints. We therefore interpret the results as a trade-off between anomaly detection accuracy and deployment simplicity, rather than as a fully equal optimization contest.

The rest of the paper is organized as follows. Section 2 describes the unsupervised time series anomaly detection problem and the classical deep learning approaches used to detect anomalies. Section 3 describes the types of foundation models that one can utilize for this time series task. Section 4 presents the anomaly detection pipelines we constructed using LLMs, TSFMs and classical deep learning. We describe benchmarks and an industrial case study in Section 5. Section 6 presents the results and discussion. Finally, section 7 presents our conclusion: general-purpose models trade accuracy for deployment simplicity.

2. UNSUPERVISED TIME SERIES ANOMALY DETECTION

Time series exhibit regular patterns. Anomalies are characterized by irregular patterns that deviate from that norm. We sort anomalies into the following categories Alnegheimish (2025):

1. *Point anomalies*. Individual signal values that are unusual relative to normal observations, such as sudden spikes, drops, or impossible readings. These are the anomalies referred to as point anomalies in the benchmark summary and results.

2. *Contextual anomalies*. Signal values that appear normal in isolation but are unusual given the surrounding temporal context or the behavior of other variables. For example, 25 °C is a normal outdoor temperature during warm seasons but anomalous in mid-winter.
3. *Collective anomalies*. Sequences of observations that are anomalous as a group, even if individual values are not clearly abnormal. Gradual drift or sustained regime shifts are examples of this category.

On the other hand, time series also have intrinsic features due to the temporal dependency, which is typically split in four components: 1) *Trend*, the long-run direction of the series, 2) *Seasonal*, the periodic-like pattern, 3) *Cyclical*, longer-term ups and downs with no fixed period, and 4) *Noise*, which should be random after removing the other three systematic components. Due to these, the statistical properties of a time series change over time, which make anomaly detection particularly challenging in this field.

The Unsupervised Time Series Anomaly Detection (UTSAD) pipeline, based on Deep Learning, typically consists of three stages: 1) Pre-processing of the time series, 2) Modeling its behavior and 3) Post-processing the outcome. The pre-processing comprises train-test split, detrending, imputation, normalization and decisions regarding the rolling window that provides inputs to the model. The models are broadly categorized into Prediction-based and Reconstruction-based approaches. In the former, a model is trained to predict future observations from previous patterns. In the latter, the model is first trained to learn a latent low-dimensional representation and then to learn a reconstruction of the input from such a representation. In post-processing, the anomaly score is calculated in terms of the difference (*error*) between the model's output $\hat{x}_t$ and the true observations x_t, for $t = 1, \ldots, T$, where T represents the time horizon. There are three popular error functions: *Point error*, *Area error* and *Dynamic Time Warping error* (DTW). For a time index t, the point error is

$$e_t = |x_t - \hat{x}_t|,$$

whereas the area error is defined as

$$e_t = \frac{1}{2l} \left| \int_{t-l}^{t+l} x \, dx - \int_{t-l}^{t+l} \hat{x} \, d\hat{x} \right|.$$

For the sake of compactness, we refer to Berndt & Clifford (1994) for the DTW calculation. The anomaly score at time index t is then defined by thresholding with a sliding window in the following manner:

$$a_t = \begin{cases} 1 & e_t > \delta_k \\ 0 & \text{otherwise} \end{cases} \quad \text{where} \quad \delta_k = \mu_{w_t} + \kappa \, \sigma_{w_t}$$

where w is the window size, μ_{w_t} and σ_{w_t} are respectively the mean and the standard deviation of the values captured in the window $\{e_t, e_{t+1}, \ldots, e_{t+w}\}$.

Prediction-based methods range from classical models such as Autoregressive Integrated Moving Average (ARIMA) Box & Pierce (1970) Pena et al. (2013), Hidden Markov Models Zucchini & MacDonald (2009) and Functional Data Analysis Torres et al. (2011) to Deep Learning models such as Long Short Term Memory Recurrent Neural Network with Non-parametric Dynamic Thresholding (LSTM-DT) Hundman et al. (2018). These methods are better at finding individual data points outside of the normal region, but trigger more false detections.

In contrast, Reconstruction-based methods assume that anomalous observations are mapped to isolated regions in the low-dimensional space where the representation lives, so they are filtered out in the reconstruction. Classical methods such as PCA Ringberg et al. (2007) have given way to Deep Learning models, including LSTM Autoencoders (AE) Sutskever et al. (2014) Malhotra et al. (2016), Generative Adversarial Networks (GAN) Geiger et al. (2020) and Transformers Xu et al. (2022).

These models identify contextual anomalies better, but are computationally intensive. Yet, there are two reasons for this overtaking of deep learning models. First, the low-dimensional latent space can be seen as a manifold learned from data in a self-supervised manner that can be reversed. Compared with other popular manifold learning methods such as Isomap or T-SNE, which are one-way only (from high-dimension to low-dimension). This is the case when using AE, and the advantage they provide is that they allow further analysis; e.g. if anomalous observations cluster in the latent space they are likely to be novelties rather than anomalies. Second, the model can be seen as a deterministic transformation of a simple joint probability distribution into the complex distribution of *normal* observations. Such is the case of Variational AE, GANs, Flows and Diffusion models.

Recent UTSAD methods also build on deep learning through approaches that use large language models (LLMs) Gruver et al. (2023), Alnegheimish, Nguyen, et al. (2024), Zhou & Yu (2025), Liu et al. (2025).

3. FOUNDATION MODELS

As mentioned previously, there are three possible approaches to leveraging foundation models in this context. The first is the use of general-purpose time series foundation models that are trained on diverse time series datasets and can be applied across multiple domains. The second is the development of domain-specific foundation models that are trained for a particular industry or application, which allows them to capture domain-relevant patterns more effectively. The third approach involves repurposing large language models (LLMs) for time series analysis tasks.

Repurposing large language models for time series analysis offers several advantages. First, if LLMs can indeed effectively address time series tasks without specific retraining, their use eliminates the need to design and train specialized models from scratch, which can be costly. Although LLMs do have a substantial training cost in general, this can be amortized across a wide range of applications.

Second, large language models are widely accessible. Organizations of varying sizes can leverage publicly available models or access them through cloud-based APIs. This accessibility significantly lowers the barrier to adoption and allows both large enterprises and smaller entities to utilize advanced modeling capabilities, perhaps including time series analysis, without extensive machine learning infrastructure or expertise.

As we test each approach, we want to stick to training and deployment principles that preserve the advantages of foundation models. These principles are:

- **No pretraining of the signals before they are fed into the foundation models**. We emphasize that no signal- or domain-specific training should occur before feeding the signals into the foundation models. Such training usually results in "*learned*" parameters that are specific to signals or domains. In all approaches, when using foundation models we only apply preprocessing techniques that do not result in such "*learned*" parameters.
- **No calibration of the foundation model for a particular task or domain** We also do not fine-tune the foundation model with signal- or domain-specific information. This will again result in a requirement that the models be fine-tuned for each set of signals - undermining the scalability of the approach.
- **No dataset-specific fine-tuning of the output.** Similar to the first principle we also only apply post-processing techniques that do not result in "*learned*" parameters that are signal- or domain-specific.

3.1. Time Series Foundation Models

Time series foundation models are large, pretrained models that can capture temporal patterns across many datasets and tasks. Examples include Chronos-2 Ansari et al. (2025) and Cisco Time Series Foundation Model (TSFM) Gou et al. (2025). Instead of being trained from scratch on each individual time series, these models are trained on large collections of time series data, and learn common structures such as trends, seasonality, and correlations. Once trained, the same model can be applied to new datasets without additional training, using zero-shot forecasting.

Time series foundation models are derived from transformers. A time series $X_{1:T}$ is first segmented into fixed-length context windows that serve as inputs to the model. The values in the window are embedded into a latent representation. Self-attention layers then model dependencies across time by allowing each timestep in the context window to attend to other timesteps, which enables the model to capture both short- and long-range temporal relationships. Given the context window, the model produces forecasts for future timesteps $x_{T+1}, x_{T+2}, \ldots$.

These models are usually trained using a self-supervised forecasting objective. During training, the model observes a historical window of the series. It learns to predict future values, and to minimize a loss, such as mean squared error, between predicted and observed values. By repeating this process across large and diverse datasets, the model learns general representations of temporal dynamics that then transfer to new time series.

3.2. LLMs for time series tasks

In SigLLM, two approaches are used for anomaly detection: prompting-based pipelines and detector-based pipelines. For the prompting-based pipeline, an LLM is prompted to directly identify anomalous timestamps in a time series. The time series is first converted into a textual representation, which is given to the model along with instructions asking it to identify anomalies in the sequence. In this pipeline, anomaly detection is performed directly by the model by reasoning over the input sequence.

In the detector-based pipeline, the LLM forecasts future values in the series, rather than directly identifying anomalies. The model predicts the next value in the time series given a historical context window. The difference between the predicted value and the observed value forms a residual error. Large residuals indicate deviations from the patterns captured by the model and are therefore treated as anomalies. As shown in Alnegheimish, Nguyen, et al. (2024), the auto-regressive nature of LLMs allows them to perform next-step time series prediction once the numerical sequence has been converted into a string representation.

For this paper, we focus exclusively on the detector-based pipeline. The original SigLLM detector pipeline is primarily univariate, which limits direct comparison with anomaly detection methods that can use multivariate context. Comparing a univariate LLM pipeline against multivariate TSFM and classical pipelines can therefore understate what an LLM-based approach can do when it has access to the same cross-signal information. Our main implementation contribution is to extend this forecasting-based SigLLM formulation to multivariate inputs by representing cross-signal context as text.

4. ANOMALY DETECTION PIPELINES

In this section, we propose forecasting-based anomaly detection pipelines for unsupervised multivariate time series anomaly detection. These pipelines build on the Detector

pipeline Alnegheimish, Nguyen, et al. (2024) in SigLLM, pretrained time-series foundation models, and pretrained pipelines Alnegheimish, Berti-Equille, & Veeramachaneni (2024) in Orion. Figure 1 summarizes the forecasting-based structure shared by the zero-shot pipelines.

4.1. LLM-Based Forecasting Pipelines

Recent work has explored the use of large language models for time series forecasting by converting numerical sequences into textual representations and leveraging next-token prediction. Approaches such as Time-LLM and LLMTime reformulate forecasting as a language modeling problem, enabling zero-shot or few-shot prediction using pretrained LLMs. Subsequent methods extend this paradigm through prompt engineering and structured representations to better capture temporal dependencies. Despite these advances, prior work has observed that LLM-based forecasting methods often exhibit inconsistent performance and may not reliably outperform specialized time series models.

In this work, we adopt an LLM-based forecasting framework for anomaly detection. Given a sliding window of past observations, the LLM predicts the next segment of the signal, and anomalies are identified based on the residual between predicted and observed values. We consider both univariate and multivariate variants of this pipeline. The univariate pipeline serves as a baseline, using only the target signal, while the multivariate pipeline incorporates additional dimensions to evaluate whether the LLM can leverage cross-variable dependencies.

4.1.1. Univariate LLM-Based Forecasting Pipeline

The univariate LLM-based forecasting pipeline predicts the next segment of the target signal using only its past values. This pipeline ignores all other dimensions and operates solely on a sliding window over the target signal. As a result, it serves as a control for evaluating whether the LLM is able to leverage additional contextual information beyond the target dimension.

We implement this pipeline using the Mistral detector within SigLLM Alnegheimish, Nguyen, et al. (2024), with Mistral-7B-Instruct-v0.2 Jiang et al. (2023) as the underlying model. This model was released on December 28, 2023. There is an updated v0.3 that was released on May 22, 2024. This is a 7B-parameter model with a 32k token context window. We choose this model because it is open-source and therefore can run locally with no API calls.

4.1.2. Multivariate LLM-Based Forecasting Pipeline

The multivariate LLM-based forecasting pipeline is the main extension introduced in this work. It extends the univariate SigLLM detector setting by incorporating multiple input dimensions. Unlike the univariate pipeline, which relies solely on the target signal, this approach provides the LLM with additional contextual signals, enabling it to capture cross-variable dependencies. This creates a more appropriate comparison with TSFMs and classical pipelines that can use multiple signals.

In addition to our own formatting methods, we explore multiple formatting methods proposed in the literature. These include value concatenation, value interleave, and digit interleave, proposed by Chatzigeorgakidis et al. (2024).

Value concatenation (VC) flattens each dimension across time where $X^S = (x_1, x_2, \dots x_T)$ and $x_i = (x^1, x^2, \dots, x^d)$. In our aforementioned example, this will create "$50, 30, 100, 55, 28, 104$".

Value interleave (VI) is similar to VC where there is an additional step to pad values to equal digit length and concatenates all features within a timestamp. That way the separator "," clearly distinguishes one timestamp and another. For example: "$050030100, 055028104$".

Digit interleave (DI) pads values to equal digit length and weaves digits positionally across dimensions. This would create the following "$001530000, 001520584$".

JSON format (JSON) encodes each observation into key:value pairs, where the key is a dimension identifier for each value in the data. Since LLMs comprehend semi-structured input, this allows us to encode dimensions into the sequence "d0:50, d1:30, d2:100, d0:55, d1:28, d2:104".

Each formatting method presents a different way to represent the data, which will result in a different LLM output. Thus, one goal of our experiments will be to find the most effective formatting method for this task–one that achieves both low mean residuals and high anomaly detection accuracy.

We predict all dimensions jointly, but extract only one dimension in the pipeline. An interesting direction for future work will be to extend this pipeline to produce joint multivariate outputs.

4.2. Time Series Foundation Model Pipelines

In addition to the univariate and multivariate LLM-based pipelines, we evaluate several pretrained time series foundation models. These models are trained on large collections of time series data, and can perform forecasting without task-specific training.

Chronos-2 Ansari et al. (2025) is a multivariate pretrained time series foundation model that discretizes continuous values into

tokens before forecasting. The model first normalizes the input series and then flattens observations across dimensions, similarly to the Value Concatenation strategy described above. Chronos-2 quantizes each value according to a fixed binning $z_i = Q(x_i)$. The following transformation shows how inputs are converted to a stream of tokens:

$$
\begin{aligned}
&[X_{t-L}, \ldots, X_t] \\
&\rightarrow \; [x^{(1)}_{t-L}, \ldots, x^{(d)}_{t-L},\; x^{(1)}_{t-L+1}, \ldots, x^{(d)}_{t-L+1},\; \ldots,\; x^{(1)}_{t}, \ldots, x^{(d)}_{t}] \\
&\rightarrow \; [z^{(1)}_{t-L}, \ldots, z^{(d)}_{t-L},\; z^{(1)}_{t-L+1}, \ldots, z^{(d)}_{t-L+1},\; \ldots,\; z^{(1)}_{t}, \ldots, z^{(d)}_{t}]
\end{aligned}
$$

Chronos-2 then uses a transformer to autoregressively generate a token sequence, which is subsequently mapped back to continuous values for final predictions. This formulation allows Chronos-2 to use training techniques similar to those used for language models. The model jointly forecasts multiple time series within the same context window, which enables it to capture cross-signal dependencies.

We also evaluate the Cisco Time Series Foundation Model (TSFM) Gou et al. (2025), a univariate pretrained forecasting model designed specifically to exploit long historical context. Cisco TSFM is based on the TimesFM architecture introduced by Google, and is designed to operate on context windows that contain information on thousands of timesteps. Specifically, the model uses the most recent 512 observations,

$$X_{t-511:t},$$

together with 512 observations sampled from further in the past, at intervals of 60 timesteps,

$$\{X_{t-60i}\}_{i=1}^{512}.$$

This allows the model to incorporate both short-term and long-range temporal context.

These pretrained models serve as strong baselines for evaluating how SigLLM-based approaches perform on forecasting and anomaly detection tasks.

4.3. Classical Deep Learning Pipeline

We use representative pipelines discussed in Section 2, especially AER and LSTM-DT. Table 2 focuses on the methods discussed in the text: AER, LSTM-DT, Chronos-2, Cisco TSFM, and the two SigLLM pipelines. These pipelines represent task-optimized baselines and zero-shot alternatives rather than methods trained under identical conditions.

Source code can be found on the Orion GitHub repository, which is an open-source library containing various verified classical deep learning pipelines. Orion provides standardized and validated implementations of these pipelines, enabling consistent and reproducible benchmarking across datasets Alnegheimish (2025).

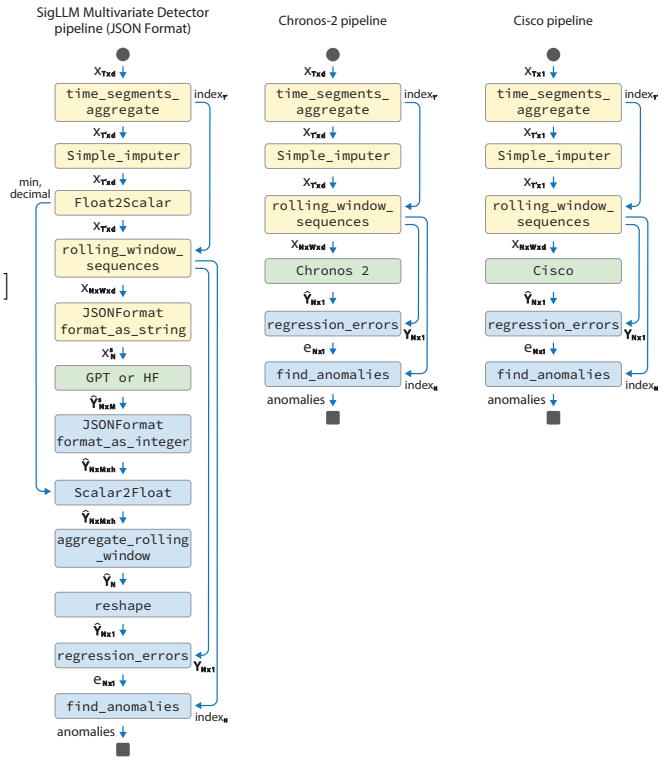


Figure 1. Comparison of forecasting-based anomaly detection pipelines: SigLLM JSON, Chronos-2, and Cisco TSFM. Each pipeline takes time series as input, produces forecasts, and uses the resulting residuals to identify anomalies.

4.4. Persistence Baseline

The persistence baseline is a simple univariate forecasting pipeline used for comparison. Persistence is defined as predicting the last observed value as the future value. This baseline therefore predicts the next value of the target signal without any learning or use of LLMs. Comparing our pipelines against this baseline indicates the extent to which other methods use more than just the last available signal in its predictions.

5. BENCHMARKS AND CASE STUDY

In this section, we discuss the datasets we used to evaluate our models.

5.1. Benchmarking datasets

Table 1 summarizes the Orion benchmark datasets. We use the Orion benchmark suite Alnegheimish, Berti-Equille, & Veeramachaneni (2024) to compare our new pipelines with established approaches. The Orion benchmark suite includes 12 time-series datasets with predefined anomaly intervals that serve as ground truth for evaluation. These datasets originate from several well-known public benchmarks and represent a mixture of real-world telemetry, operational metrics, and synthetic signals. The benchmark suite includes datasets from

Table 1. High-level overview of all 12 benchmark datasets. Figure sourced from Wong et al. (2022)

Datasets	Source	NASA		YAHOO				NAB					UCR
	Name	**MSL**	**SMAP**	**A1**	**A2**	**A3**	**A4**	**Art**	**AdEx**	**AWS**	**Traffic**	**Tweets**	**UCR**
Properties	**Synthetic**	No	No	No	Yes	Yes	Yes	Yes	No	No	No	No	Mix
	# Signals	27	53	67	100	100	100	6	5	17	7	10	250
# Anomalies	**Point (*len=1*)**	0	0	68	33	935	833	0	0	0	0	0	3
	Collective (*len>1*)	36	67	110	167	4	2	6	11	30	14	33	247
# Data Points	**Anomalous Points**	7766	54696	1669	466	943	837	2418	795	6312	1560	15651	49363
	Total Points	132046	562800	94866	142100	168000	168000	24192	7965	67644	15662	158511	19353766

NASA spacecraft telemetry, the Numenta Anomaly Benchmark (NAB), Yahoo's S5 anomaly dataset, and the UCR anomaly archive. Across all datasets, there are 742 time-series signals that contain a total of 2,599 labeled anomalies. Each signal includes annotated anomaly ranges, which makes the datasets suitable for standardized benchmarking.

The NASA datasets consist of spacecraft telemetry from the Mars Science Laboratory (MSL) rover and the Soil Moisture Active Passive (SMAP) satellite. These signals record subsystem health measurements such as temperatures, voltages, currents, and communication metrics. MSL contains 27 signals with 36 labeled anomalies, while SMAP contains 53 signals with 67 anomalies, for a total of 80 telemetry channels and 105 anomalies. These datasets were originally released alongside the LSTM-DT anomaly detection study and have already been divided into training and testing partitions.

The Numenta Anomaly Benchmark (NAB) contains time series derived from operational systems and internet infrastructure. In the Orion benchmark, five NAB subsets are used: artWithAnomaly (6 signals, 6 anomalies), realAWSCloudwatch (17 signals, 30 anomalies), realAdExchange (5 signals, 11 anomalies), realTraffic (7 signals, 14 anomalies), and realTweets (10 signals, 33 anomalies). These datasets include metrics such as cloud infrastructure monitoring signals (e.g., CPU utilization), online advertisement statistics, transportation measurements, and social media activity counts. The artWithAnomaly subset is synthetic, while the remaining datasets contain real operational measurements.

The Yahoo S5 dataset includes four subsets labeled A1 through A4. The A1 subset contains 67 real signals with 178 anomalies derived from production traffic in Yahoo computing systems. The remaining subsets (A2, A3, and A4) are synthetic datasets with 100 signals each, and contain 200, 939, and 835 anomalies respectively. Many of the anomalies in A3 and A4 are only a few time steps long, which is short enough that anomaly detection algorithms struggle to detect them.

The UCR anomaly archive contains 250 time-series signals, each of which contains a single labeled anomaly. These anomalies have been artificially introduced into otherwise normal signals, using transformations such as reversing, smoothing, interpolation, or prolonging sections of the signal. The dataset was originally introduced as part of a data mining challenge designed to create difficult anomaly detection scenarios.

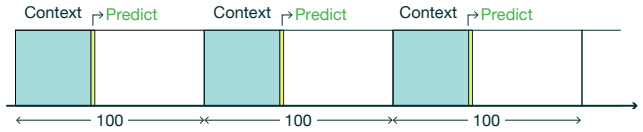


Figure 2. Illustration of the sliding window forecasting setup. Each window uses a fixed-length context to predict the next timestamp, with a stride of 100 between successive windows.

Together, these datasets provide a wide variety of signal lengths, anomaly frequencies, and data characteristics, and allow anomaly detection systems to be evaluated under a variety of conditions.

5.2. Case study: Wind turbine dataset

GE Vernova provided us with an industrial wind turbine dataset containing six features and 127k timestamps. The data was collected in 10-minute intervals from January 1, 2022 to June 28, 2024. We were told that this dataset contains anomalous behavior between March 31, 2024 and May 30, 2024. Our goal was to find a method that can detect this anomalous region.

To ensure a controlled comparison across methods, we fix the context window to 40 timesteps and stride to 100 for all pipelines except Cisco TSFM, whose architecture requires long-range historical context. A stride of 100 and context window of 40 means that we use $X_{t_k:t_k+39}$ to predict X_{t_k+40}, where $t_k = 100k$. A stride of 100 is roughly once-per-day sampling, and a window size of 40 is $6\frac{2}{3}$ hours of context. We use a large stride due to computational limits. This is a disadvantage for methods that benefit from denser scoring, so we include a forecast-horizon sensitivity analysis and discuss the limitations imposed by the large stride in Section 6.3. Figure 2 illustrates the setup.

For Cisco TSFM, we use a stride of 100 and a window size of 30,720. This larger window is used because the model is designed to incorporate long-range historical context. Using a window size of 40 would prevent the model from being able to incorporate any of this context and put it outside of its intended

operating mode. We maintain a fair comparison between time series foundation models and LLM-based pipelines through Chronos-2.

6. RESULTS AND DISCUSSION

In this section, we discuss results on the datasets described in the previous section. For our F1 score calculations, we use range-based evaluation, where a predicted anomaly is counted as a true positive if it overlaps with a ground-truth anomaly interval and as a false positive if it falls entirely outside all ground-truth intervals. Missed ground-truth intervals are counted as false negatives. We also discuss false alarms and detection delay because F1 alone does not capture all operational concerns. Here, false alarm behavior is represented by the false positive component of the range-based evaluation, while detection delay is the time between the start of a labeled anomaly interval and the first overlapping predicted interval. In the wind turbine case study, the stride directly limits this delay because predictions are evaluated only every 100 timesteps.

6.1. Orion Benchmark

Table 2 shows the F1 scores of each pipeline on the Orion benchmark. Figure 3 shows the aggregate F1 scores for selected methods, including AER (the top-performing classical deep learning pipeline), the time series foundation models introduced earlier (Cisco and Chronos-2), and our univariate LLM-based pipeline. The multivariate SigLLM results complete this comparison by testing an LLM-based detector in a multivariate input setting, rather than comparing multivariate TSFM and classical methods only against a univariate LLM baseline.

TSFMs do not perform as well as classical deep learning pipelines, but are close. We can see in Table 2 that specialized anomaly detection pipelines, particularly AER, achieve the strongest and most consistent performance across datasets. LSTM-DT remains competitive but generally does not surpass AER. In contrast, general-purpose foundation models such as Chronos-2 and Cisco TSFM exhibit moderate and highly variable performance, with strengths on certain Yahoo subsets but weaker results on NASA and NAB datasets. This gap should be interpreted in light of the different assumptions: AER and LSTM-DT are task-specific anomaly detection systems, while the TSFM and LLM pipelines are used without anomaly-specific training.

LLM-based approaches fall short compared to deep learning, but are competitive to TSFMs. For this comparison let us consider univariate and multivariate signals separately since we developed a separate multivariate method to use LLMs. For univariate signals and methods, our SigLLM univariate method performs competitively against TSFMs. This is significant considering that LLMs were not designed for this task, while TSFMs were.

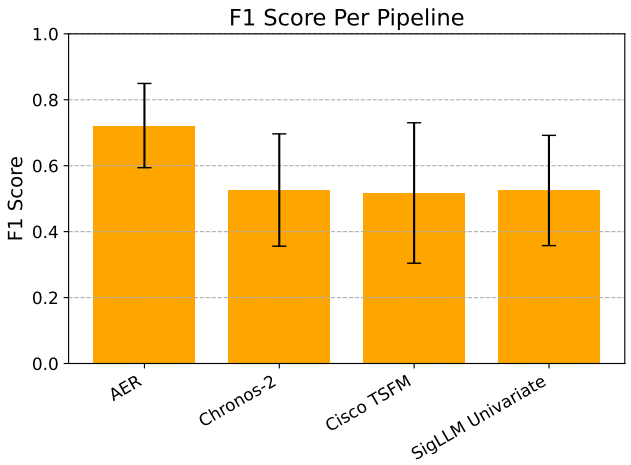


Figure 3. Aggregate F1 score per method for the entire Orion benchmark. Aggregate F1 scores shown are computed as the mean across all available datasets per pipeline. AER performs clearly better than TSFM models and LLM-based models, which perform similarly.

For multivariate, we can only compare the methods using NASA datasets SMAP and MSL, as these two contain multivariate signals. For SMAP, the best classical deep learning technique achieves a F1 score of 0.764 while TSFMs achieve an average of 0.424 and LLM-based methods achieve 0.412.

Overall, these results indicate that while foundation and LLM-based methods offer flexibility and zero-shot applicability, they currently underperform specialized anomaly detection pipelines in both consistency and peak accuracy. At the same time, TSFMs do not necessarily outperform the LLM-based approaches.

LLM-based approaches (SigLLM) show similarly inconsistent behavior, with strong results on select datasets but lower performance elsewhere. However, LLM-based approaches are much more computationally expensive than other pipelines. Each signal takes on the order of seconds or minutes for classical deep learning pipelines but on the order of hours for SigLLM pipelines.

We also observe a clear distinction between point anomalies and contextual anomalies, with contextual anomalies being consistently harder to detect. Figure 4 shows the F1 scores of various methods on point anomalies versus contextual anomalies. AER and Chronos-2 degrade by roughly 25%, while Cisco-TSFM degrades by roughly 50%. In contrast, SigLLM univariate performs similarly across both anomaly types. This suggests that SigLLM's forecasting-based approach is less sensitive to the structural differences between point and contextual anomalies.

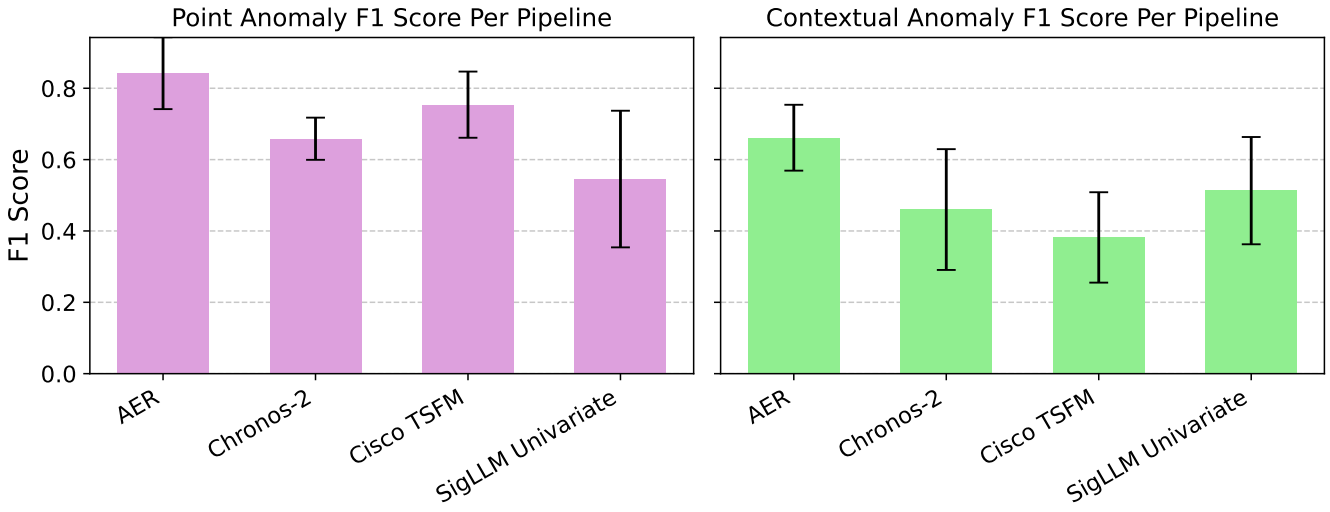


Figure 4. Aggregate F1 scores for point anomalies and contextual anomalies in Orion benchmark.

6.2. Wind turbine case study

The GE wind turbine case study is an extreme case of contextual anomaly, which makes it very subtle and difficult to pick up the anomaly. In Figure 5, we used K-means clustering to show that there is a regime shift in the exact anomaly range, especially in V4. This case study shows that K-means clustering can be effective in cases where anomalies show up as distributional shifts.

The SigLLM multivariate pipeline outperforms the SigLLM univariate pipeline on forecasting and anomaly detection for this dataset. A paired t-test on absolute residuals ($n = 5240$) indicates that the multivariate pipeline produces a significantly lower forecasting error than the univariate pipeline ($t = -4.29, p < 10^{-4}$). The gains are modest in our experiments, though. On average, the multivariate approach has 4.77% lower absolute residuals than the univariate pipeline.

Table 3 reports range-based F1 scores using area error with a window size of 3 across the four wind turbine target signals.

Figure 6 compares the anomaly scores produced by the five forecasting pipelines on the GE Vernova turbine dataset. Each method generates forecasts for the target signals, and the anomaly score is computed using the residual between the predicted and observed values. In the figure, the upper panels show the residuals for each target feature, while the lower panel shows the average anomaly score across all target columns. The highlighted vertical regions correspond to detected anomalies, with darker shading indicating higher anomaly scores. For all pipelines, we use the window parameters “window_size _portion": 0.3, “window_step_size_portion": 0.1, “min_percent": 0.2, and “anomaly_padding": 10. All methods except Cisco operate on sequences of length $n = 1310$ per target column, while the Cisco pipeline processes $n = 971$ samples due to preprocessing constraints discussed above.

The persistence baseline performs particularly well on this dataset. This suggests that much of the short-term signal behavior is dominated by autocorrelation. In such settings, the one-step-ahead forecasting task becomes difficult to improve upon, which limits the magnitude of gains that can be achieved by using multivariate pipelines. It shows that simple pipelines can be good at handling real-world datasets. Nevertheless, the multivariate approach consistently yields small but measurable improvements over the univariate baseline.

One limitation of the SigLLM pipelines is that the context window was constrained to 40 timesteps due to GPU memory limitations during inference. Because multivariate time series must first be serialized into text before being processed by the LLM, the resulting token sequences grow quickly with both the window length and the number of signals. Larger windows therefore lead to substantially higher memory usage and inference cost. Using NVIDIA TITAN RTX GPUs (24GB memory), we found that a window size of 40 provided a practical balance between context length and computational feasibility. In addition, inference took significantly longer when using the SigLLM pipelines versus the foundation model baselines. The LLM-based pipelines required hours to process the wind turbine dataset, whereas Chronos-2 and Cisco TSFM required minutes under the same experimental setup.

Larger language models may improve forecasting accuracy, because they have a greater capacity to capture relationships between signals. In addition, API-based LLMs often support much larger context windows, which could allow the pipeline to incorporate longer historical trends and potentially improve forecasting performance.

6.3. Sensitivity to stride and forecast horizon

The stride of 100 reduces runtime but also reduces temporal resolution. Since the wind turbine data are sampled every 10 minutes, scoring every 100 timesteps means that a detection can be delayed by up to roughly 1,000 minutes relative to the start of an anomalous interval. Smaller strides would improve detection delay and provide denser anomaly scores, but would increase runtime approximately linearly for the LLM pipelines.

We also evaluated multi-step forecasting on the wind turbine dataset to test whether longer forecast horizons produce more useful anomaly scores than one-step prediction. The mean range-based F1 scores across V1–V4 were as follows: SigLLM multivariate achieved 0.125, 0.200, 0.222, and 0.140 at horizons 1, 3, 5, and 10; SigLLM univariate achieved 0.211, 0.146, 0.261, and 0.069; and persistence achieved 0.170, 0.125, 0.244, and 0.049. All three methods performed best around horizon 5, but no method consistently dominated. This suggests that forecast horizon affects residual behavior, but does not remove the difficulty of separating the anomalous interval from normal operation.

6.4. Discussion on Context Window

The effective context window, defined as the amount of past signal a model can use, directly limits what types of anomalies a forecasting-based method can detect. Since anomaly scores come from prediction errors, limited context leads to poor baseline estimates when there are long-term patterns, which is common in time series data.

For LLM-based approaches such as Mistral-7B-Instruct-v0.2, context is defined in tokens, for example 32k tokens. In time series settings, this is misleading. Numerical values must be serialized into text, and each data point expands into multiple tokens. This reduces the number of timesteps that fit in the window. The issue becomes more severe in multivariate settings where multiple features are encoded together. Larger models such as ChatGPT increase the token limit to 128k to 400k, which improves coverage, but the same inefficiency remains and computational cost grows quickly. Computation costs for API-based models like ChatGPT are mostly financial cost, while computation costs for locally run models like Mistral-7B are mostly time cost.

Time Series Foundation Models use more efficient representations. The Cisco Time Series Foundation Model applies a multi-resolution approach that combines detailed recent data with a compressed view of long-range history. The transformer processes a limited number of tokens, but the effective receptive field can span tens of thousands of timesteps. This improves modeling of long-term structure, though distant high-frequency detail is partially lost. In contrast, Chronos-2 operates directly on discretized values with a fixed context such as 2048 timesteps. This avoids tokenization overhead but limits long-range coverage.

In practice, context length determines which anomalies can be detected. Short windows are sufficient for point anomalies and local deviations. Longer context is required for seasonal effects, drift, and other contextual anomalies. The main limitation of LLM-based methods is not the nominal context size, but the inefficiency of representing time series as text, which significantly reduces usable temporal context in practice.

7. CONCLUSION

Under strict "general-purpose foundation model" constraints, LLM-based multivariate anomaly detection pipelines are feasible and slightly improve over univariate LLM baselines, but they remain less accurate and much more computationally expensive than time-series foundation models and simpler baselines on industrial forecasting-based anomaly detection.

These findings suggest a practical tradeoff. Classical models remain the preferred choice for accuracy when task-specific training is feasible. In contrast, foundation models offer a more suitable alternative for zero-shot deployment at the cost of reduced and less consistent performance. LLM-based approaches extend this flexibility further by eliminating the need for time-series-specific modeling entirely, but introduce significant computational overhead and higher variance in results.

Beyond aggregate performance, our experiments highlight an important limitation of forecasting-based anomaly detection. On real-world industrial data, simple persistence-based methods and spatial heuristics such as K-means residuals can outperform more complex forecasting models. In the wind turbine case study, K-means clustering serves as a simple diagnostic that highlights a clear shift in signal behavior during the anomalous period, especially in V4. This suggests that when anomalies appear as distributional shifts rather than forecasting errors, spatial or distance-based methods can be more effective than forecasting-based approaches.

Overall, while general-purpose models provide a compelling direction for scalable anomaly detection, they do not yet match the reliability or accuracy of specialized approaches. The reported F1 scores should also be interpreted with a strong operational caveat: in many industrial monitoring settings, even the better benchmark scores would require additional validation, threshold tuning, and human review before they could be considered usable as alarms. Future progress will likely depend on improving efficiency, extending effective context length, and developing representations that better capture multivariate dependencies without sacrificing the deployment advantages of general-purpose models.

REFERENCES

Alnegheimish, S. (2025). *Machine learning systems for unsupervised time series anomaly detection* (PhD thesis, Massachusetts Institute of Technology). Retrieved from https://dai.lids.mit.edu/wp-content/uploads/2025/08/sarah-alnegheimish-thesis.pdf

Alnegheimish, S., Berti-Equille, L., & Veeramachaneni, K. (2024). Orionbench: Benchmarking time series generative models in the service of the end-user. In *Ieee int. conf. on big data.* doi: 10.1109/BigData62323.2024.10825341

Alnegheimish, S., Nguyen, L., Berti-Equille, L., & Veeramachaneni, K. (2024). Can large language models be anomaly detectors for time series? In *2024 ieee 11th int. conf. on data science and advanced analytics (dsaa)* (p. 1-10). doi: 10.1109/DSAA61799.2024.10722786

Ansari, A. F., Stella, L., Turkmen, C., Zhang, X., Mercado, P., Shen, H., ... Wang, H. (2025). *Chronos-2: A large-scale foundation model for time series forecasting.*

Berndt, D. J., & Clifford, J. (1994). Using Dynamic Time Warping to FindPatterns in Time Series. In *Aaai-94 workshop on knowledge discovery in databases.* Seattle, Washington.

Box, G. E., & Pierce, D. A. (1970). Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American statistical Association*, *65*(332), 1509–1526.

Chatzigeorgakidis, G., Lentzos, K., & Skoutas, D. (2024). Multicast: Zero-shot multivariate time series forecasting using llms. *arXiv preprint arXiv:2405.14748*.

Geiger, A., Liu, D., Alnegheimish, S., Cuesta-Infante, A., & Veeramachaneni, K. (2020). Tadgan: Time series anomaly detection using generative adversarial networks. In *2020 ieee int. conf. on big data (big data)* (pp. 33–43).

Gou, L., Khare, A., Pabolu, P., Patel, P., Ross, J., Shen, H., ... Yang, H. (2025). *Cisco time series foundation model technical report.* Retrieved from https://arxiv.org/abs/2511.19841

Gruver, N., Finzi, M., Qiu, S., & Wilson, A. G. (2023). Large language models are zero-shot time series forecasters. In *Proc. of the 37th int. conf. on neural information processing systems (neurips).* Curran Associates Inc.

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).

Jiang, A. Q., Sablayrolles, A., Roux, A. M., Bamford, C., Chaplot, D. S., Casas, D. d., ... others (2023). Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Liu, J., Zhang, C., Qian, J., Ma, M., Qin, S., Bansal, C., ... Zhang, D. (2025). Large language models can deliver accurate and interpretable time series anomaly detection. In *Proc. of the 31st acm sigkdd conference on knowledge discovery and data mining (kdd)* (p. 4623–4634). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3711896.3737239 doi: 10.1145/3711896.3737239

Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., & Shroff, G. (2016). Lstm-based encoder-decoder for multi-sensor anomaly detection. In *Proceedings of the 2016 international conference on machine learning (icml) anomaly detection workshop.*

Pena, E. H., de Assis, M. V., & Proença, M. L. (2013). Anomaly detection using forecasting methods arima and hwds. In *32nd int. conf. of the chilean computer science society (sccc)* (pp. 63–66).

Ringberg, H., Soule, A., Rexford, J., & Diot, C. (2007). Sensitivity of pca for traffic anomaly detection. In *Proc. of the 2007 acm sigmetrics* (pp. 109–120).

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, *27*.

Torres, J. M., Nieto, P. G., Alejano, L., & Reyes, A. (2011). Detection of outliers in gas emissions from urban areas using functional data analysis. *Journal of hazardous materials*, *186*(1), 144–149.

Wong, L., Liu, D., Berti-Equille, L., Alnegheimish, S., & Veeramachaneni, K. (2022). AER: Auto-Encoder with Regression for Time Series Anomaly Detection. In *2022 ieee int. conf. on big data (big data)* (p. 1152-1161). doi: 10.1109/BigData55660.2022.10020857

Xu, J., Wu, H., Wang, J., & Long, M. (2022). Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy. In *International conference on learning representations.*

Zhou, Z., & Yu, R. (2025). Can llms understand time series anomalies? *Int. Conf. on Learning Representations (ICLR).*

Zucchini, W., & MacDonald, I. L. (2009). *Hidden markov models for time series: an introduction using r*. Chapman and Hall/CRC.

Table 2. Benchmark summary results for the main pipelines discussed in the paper. The SigLLM univariate results are pulled from Alnegheimish, Nguyen, et al. (2024) while the classical deep learning pipelines are pulled from Alnegheimish (2025).

	NASA		UCR	Yahoo S5				NAB				
Pipeline	MSL	SMAP	UCR	A1	A2	A3	A4	Art	AWS	AdEx	Traf	Tweets
						F1 Score						
`AER`	0.657	0.764	0.468	0.782	0.978	0.893	0.716	0.750	0.702	0.733	0.611	0.606
`LSTM DT`	0.455	0.732	0.414	0.740	0.978	0.744	0.638	0.400	0.537	0.759	0.611	0.588
`Chronos-2`	0.496	0.400	0.148	0.677	0.738	0.645	0.574	0.273	0.483	0.688	0.632	0.560
`Cisco TSFM`	0.396	0.448	–*	0.672	0.872	0.817	0.655	0.273	0.333	0.647	0.341	0.235
`SigLLM Univariate`	0.429	0.431	–*	0.615	0.828	0.376	0.363	0.400	0.362	0.727	0.480	0.762
`SigLLM Multivariate`	0.397	0.393	–**	–**	–**	–**	–**	–**	–**	–**	–**	–**

* Results unavailable due to time constraints.
** These datasets are univariate so the multivariate pipeline is not applicable.

Table 3. Range-based F1 scores using area error (window size 3) across four signals. Aggregate F1 is computed from total true positives, false positives, and false negatives across all signals. A predicted range is counted as correct if it overlaps the ground truth anomaly interval.

Pipeline	**V1**	**V2**	**V3**	**V4**	**Aggregate**
			F1 Score		
`SigLLM Multivariate`	0.000	0.000	0.667	0.500	0.308
`SigLLM Univariate`	0.000	0.400	0.000	0.000	0.133
`Chronos-2`	0.000	0.286	0.000	0.000	0.105
`Cisco TSFM`	0.667	0.400	0.000	0.000	0.286
`Persistence Baseline`	0.667	0.500	0.500	0.500	0.533

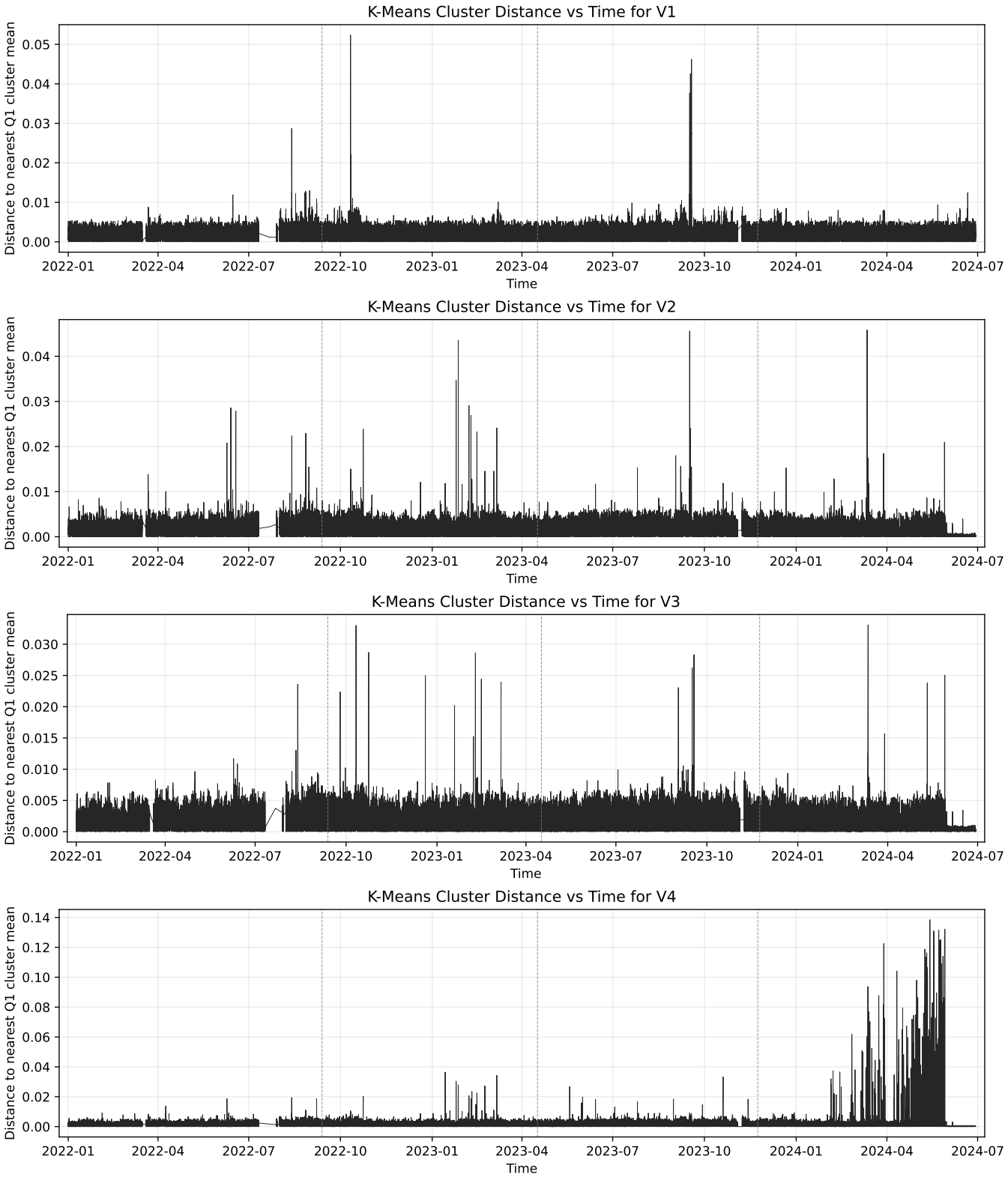
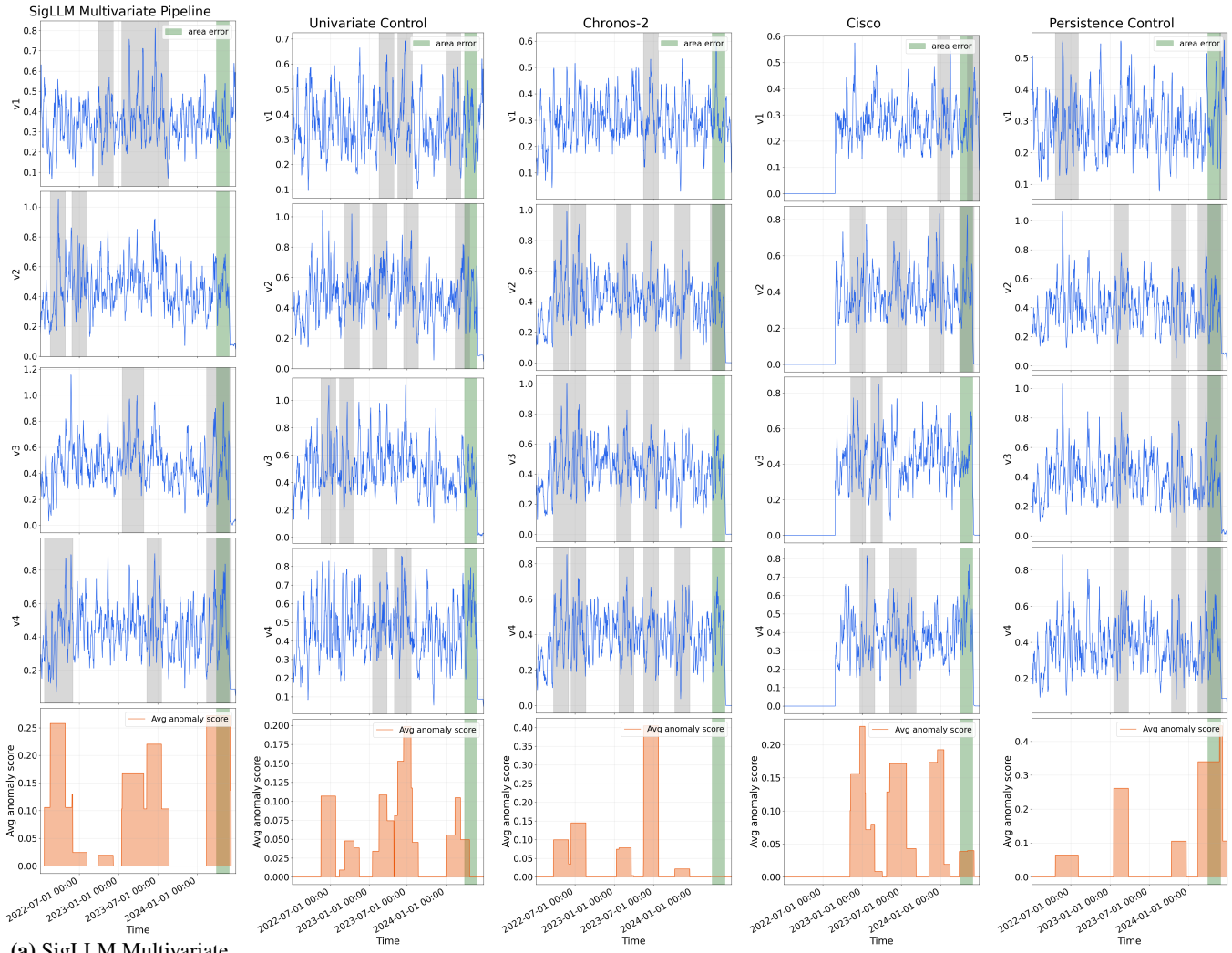


Figure 5. Distance to nearest cluster for signals V1–V4. We fit 1-D K-means on the first 25% of each signal to obtain 100 cluster centers. Each data point is then assigned to its nearest center, and the residual is computed as the distance to that center.



(a) SigLLM Multivariate Pipeline **(b)** Univariate control **(c)** Chronos-2 **(d)** Cisco TSFM **(e)** Persistence

Figure 6. Residual-based anomaly scores produced by the five forecasting pipelines. Upper panels show residuals for each target feature, and the lower panel shows the average anomaly score across all targets. Red vertical bands indicate detected anomalies, with darker shading corresponding to higher anomaly scores.