On Cost-Effectiveness of Language Models for Time Series Anomaly Detection

information

MDPI

Article

On Cost-Effectiveness of Language Models for Time Series Anomaly Detection

Ali Yassine, Luca Cagliero * and Luca Vassio

Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129 Torino, Italy; ali_yassine@polito.it (A.Y.); luca.vassio@polito.it (L.V.)
* Correspondence: luca.cagliero@polito.it; Tel.: +39-011-090-7179

Abstract

Detecting anomalies in time series data is crucial across several domains, including healthcare, finance, and automotive. Large Language Models (LLMs) have recently shown promising results by leveraging robust model pretraining. However, fine-tuning LLMs with several billion parameters requires a large number of training samples and significant training costs. Conversely, LLMs under a zero-shot learning setting require lower overall computational costs, but can fall short in handling complex anomalies. In this paper, we explore the use of lightweight language models for Time Series Anomaly Detection, either zero-shot or via fine-tuning them. Specifically, we leverage lightweight models that were originally designed for time series forecasting, benchmarking them for anomaly detection against both open-source and proprietary LLMs across different datasets. Our experiments demonstrate that lightweight models (<1 Billion parameters) provide a cost-effective solution, as they achieve performance that is competitive and sometimes even superior to that of larger models (>70 Billions).

Keywords: time series anomaly detection; large language models; cost-effective approaches

Academic Editors: Ruobin Gao, Maohan Liang and Xiaocai Zhang

Received: 28 November 2025
Revised: 1 January 2026
Accepted: 7 January 2026
Published: 12 January 2026

1. Introduction

Time Series Anomaly Detection (TSAD) is a well-known data mining problem. In real-life applications, anomalies can indicate cyber-attacks, service failures, system defects, or health-risky conditions [1]. For this reason, anomaly detection has already been applied in financial fraud detection [2], predictive maintenance [3], and healthcare monitoring [4].

To automatically identify the anomalies present in a given time series, the research community has already explored several techniques, including outlier detection, regression, signal processing, stochastic learning, and deep learning [5]. However, many existing approaches require a large number of domain-specific training samples as well as a computationally intensive training process.

Pre-trained Language Models (LMs) are well-established tools for understanding and generating sequential data. By leveraging the transformer architecture [6], LMs have already demonstrated strong performance in foundational time series analysis tasks, such as forecasting [7,8] and TSAD [9–11]. However, learning the specific characteristics of anomalous patterns often requires computationally intensive pretraining and fine-tuning procedures. Both steps become unfeasible when there is a lack of annotated data or computational resources.

Large Language Models (LLMs) are characterized by a more robust model pretraining than traditional LMs, as the number of model parameters increases from millions to

billions [12]. The research community in TSAD has recently explored LLMs zero-shot capabilities to solve TSAD [13]. However, previous work relies on proprietary LLMs that were pretrained on text rather than time series data. Furthermore, their inference cost on long time series is non-negligible.

In this paper, we explore the use of lightweight LMs [7,14], originally designed for time series forecasting, to tackle TSAD. Firstly, we envisage a model-agnostic transfer learning strategy which consists of estimating the residuals between the predicted and actual time series values to detect potential anomalies in the given time series. Then, we compare the performance of pretrained and fine-tuned LMs with that of state-of-the-art LLM-based approaches, including both open- and closed-source models. Finally, we carry out a cost–benefit analysis, weighing predictive performance and model cost in terms of training time, inference time, memory, and, possibly, monetary costs. The ultimate goal is to identify the most cost-effective LM-based TSAD solution.

The main contributions of this paper are as follows:

- We propose a model-agnostic, residual-driven approach for Time Series Anomaly Detection (TSAD) using LMs, reducing the need for extensive training data and computational resources.
- We systematically compare pretrained and fine-tuned LMs with state-of-the-art LLM-based approaches, including both open- and closed-source models.
- We perform a cost–benefit analysis considering predictive performance (F1 score) and efficiency metrics, such as inference time, memory usage, and estimated monetary cost, to identify the most practical LM-based TSAD solutions.
- We demonstrate that lightweight LMs can achieve competitive TSAD performance compared to proprietary LLMs while substantially reducing computational costs, offering practical insights for real-world deployment.

Figure 1 showcases the outcomes of the cost–benefit analysis. It shows the average F1 score [15], which measures the anomaly detection performance, and the inference time, expressed in seconds, which measures the overall cost, on the benchmark dataset. Red bubbles denote the LLMs, whereas the (light) blue ones represent the LMs. In particular, traditional LMs achieve TSAD performance comparable to proprietary LLMs while requiring substantially fewer computational resources.

The rest of the paper is organized as follows. Section 2 reviews the relevant literature on time series forecasting and anomaly detection using LMs and LLMs. Section 3 details our methodology for unsupervised, residual-driven TSAD using lightweight LMs. Section 4 outlines the experimental settings, including the datasets, model configurations, evaluation metrics, parameter choices, and reproducibility considerations. Section 5 presents the experimental evaluation across multiple benchmark datasets, comparing performance, resource usage, and inference cost. Section 6 provides a critical analysis of the results, emphasizing practical implications and highlighting the limitations of this study. Finally, Section 7 concludes the paper with key insights and discusses potential avenues for future research.

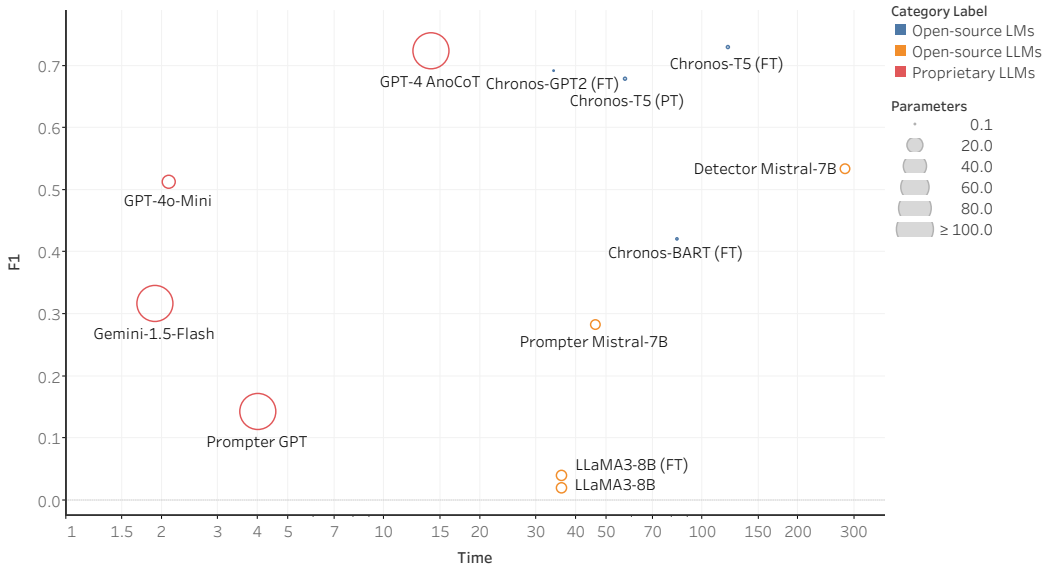

Figure 1. Cost–benefit analysis F1 score (measuring the effectiveness of the TSAD model) vs. inference time (quantifying model efficiency). Each bubble corresponds to a different model (proprietary LLMs are depicted in red, open-source LLMs in blue, open-source LMs in light blue), and the bubble size represents the number of parameters of each model. The TSAD models at the top-left corner are the most cost-effective. The inference times are averaged over the input series and measured via controlled experiments. For all models we reused the same tokenization/preprocessing steps to ensure a fair comparison. For the cost of proprietary models we considered the experimental billing measured by the official service. The hardware settings used in our research are described in Section 4.5.

2. Related Work

2.1. Time Series Forecasting Using Language Models

Several prior works have explored time series forecasting by prompting LLMs with historical values. For example, ref. [16] explores the use of general-purpose LLMs (GPT-3, LLaMA-2) for zero-shot time series forecasting. After careful preprocessing of the numerical series values, it adopts LLMs directly as forecasters, without any additional textual context or prompt engineering, effectively treating next-value prediction as a next-token completion task. In contrast, ref. [17] adopts basic transformations, such as scaling, quantization, and tokenization, to prompt GPT 3.5 [18] and Mistral [19]. PromptCast [20] and Chronos [7] investigate the use of different prompting and quantization strategies to deal with Time Series Forecasting (TSF) using lightweight Language Models (e.g., T5 [21], BART [22]). Metatransformer [23], instead, fine-tunes a transformer architecture for non-textual modalities, including time series.

2.2. Time Series Anomaly Detection Using Language Models

The use of LLMs to detect point- and segment-wise anomalies in time series data has already been explored. For instance, ref. [24] fine-tunes a transformer model for various time series processing tasks, including classification, TSAD, TSF, and few-shot or zero-shot learning. It treats TSAD as a binary classification problem and adds classification layers on top of the transformer network. In [17], the authors adopt a two-step pipeline, where anomaly detection is performed on top of time series forecasts. AnomalyLLM [25] adopts knowledge distillation for TSAD. A student network is trained to mimic the features of a larger LLM-based teacher network. Ref. [26] investigates whether visual LLMs can understand and detect anomalies in time series. They find situations in which LLMs perform better when time series are represented as images and show varying performance

across models. Ref. [27] studies GPT-4 and LLaMA3 for Time Series Anomaly Detection. They show that with prompting and fine-tuning, the models can detect anomalies and provide explanations.

2.3. Position of Our Work

Our study compares the performance of lightweight pretrained and fine-tuned LMs with LLMs for unsupervised TSAD. The aim is to identify cost-effective approaches capable of performing accurate anomaly detection at minimal cost. Similar to [17], we postprocess the output of a time series forecasting module to address TSAD. However, instead of fine-tuned or prompt-based anomaly classification models, we first employ lightweight LMs as forecasting engines and then derive anomaly signals from the residual patterns between predicted and observed values.

Unlike existing strategies based on proprietary LLM prompting (e.g., [13]), which may involve significant computational power, inference time, and financial budget, we focus on cost-effective models that have been originally designed for time series forecasting. The underlying idea is to reuse the pretrained models designed for time series data for TSAD to minimize data curation, training, and inference costs. Benchmarking the performance of open-source and proprietary LMs and LLMs performance on TSAD reveals that model performance depends not only on model scale or access level, but also on the methodological design of the detection pipeline. This highlights the importance of efficient model utilization and pipeline optimization to obtain practical and scalable LM-based time series anomaly detectors.

3. Method

This section describes the methodological step applied to address TSAD using LMs and LLMs. Specifically, Section 3.1 introduces preliminary concepts and formalizes the TSAD problem. Section 3.2 describes the forecasting-then-anomaly detection pipeline using a visual sketch of the pipeline, whereas Section 3.3 reports the algorithmic description. Finally, Section 3.4 details the residual functions used in the present work.

3.1. Preliminaries

Let $T = [t_1, t_2, \ldots, t_n]$ denote a time series of n real-valued observations, and let $T_{i,w}$ represent a subsequence of length w from position i to $i + w - 1$ ($1 \leq i + w - 1 \leq n$).

Given a context window $W = T_{i,w}$ and a forecast horizon h, Time Series Forecasting (TSF) aims to predict the next h future values, i.e., the subsequence

$$[t_{i+w}, t_{i+w+1}, \ldots, t_{i+w+h-1}].$$

Time Series Anomaly Detection (TSAD) identifies points or subsequences that deviate from expected temporal patterns. In an unsupervised setting, anomalies are detected solely from the inherent characteristics of the original time series data and from the knowledge stored in the pretrained model.

Our research goals are twofold: (1) understand whether larger models, which require non-negligible training and inference costs, achieve significantly better performance than lightweight LMs; (2) test the effectiveness of a transfer learning pipeline that reuses traditional LMs pretrained for TSF to tackle TSAD.

3.2. The Proposed Pipeline

We propose to combine an LM pretrained on time series data and fine-tuned for TSF with a residual-driven TSAD strategy (see Figure 2).

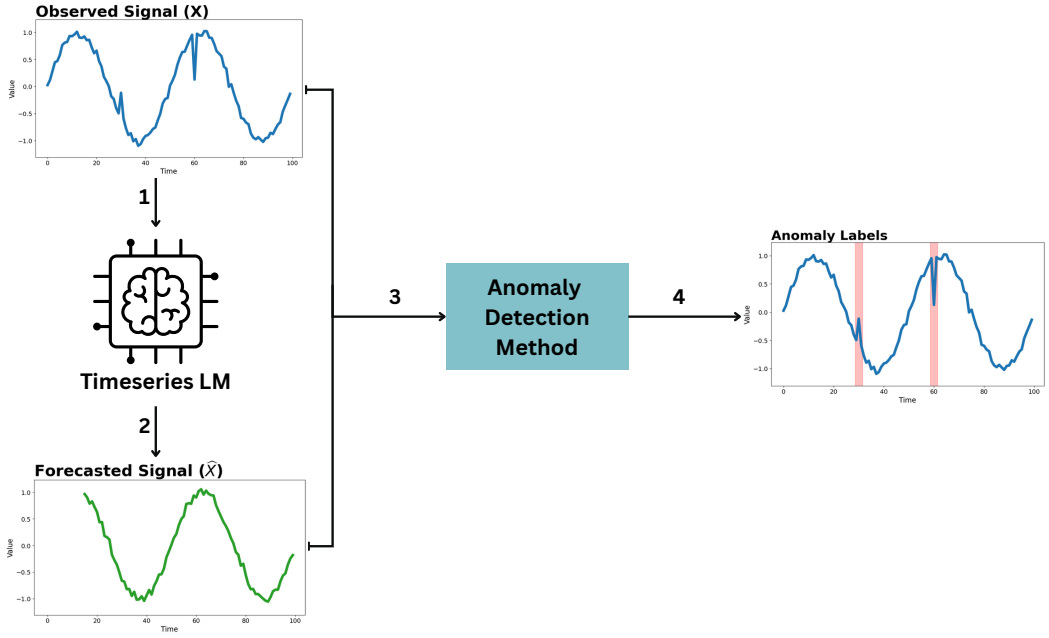

Figure 2. Sketch of the TSAD pipeline. (1) The Observed Signal (X) is processed by a Timeseries LM to (2) generate the Forecasted Signal ($\hat{X}$). (3) Both signals are passed to the Anomaly Detection Method, which calculates residuals to (4) produce the Anomaly Labels. In the final plot, light red shaded regions indicate detected anomalies where the residual exceeds detection threshold.

A more detailed description of each step follows.

Language Model-Based Time Series Forecasting

To generate forecasts for a time series T, we employ Chronos [7], a transformer-based language model specifically designed for time series forecasting. Chronos formulates forecasting as a sequence modeling task by transforming continuous numerical values into discrete tokens. This is achieved through quantization, which discretizes values into a finite set of bins, followed by tokenization, where each value is replaced by the symbol corresponding to its bin. The resulting token sequence can be processed by standard transformer architectures, e.g., [21], trained using the cross-entropy loss. Chronos was pretrained on a large collection of real and synthetic time series, including sequences generated via Gaussian processes, to capture temporal dependencies and seasonal patterns across diverse domains.

Given the time series T, we apply the following steps for time series forecasting:

1. Warm-up context: The first window of length w serves as a warm-up. No forecasts are generated for this initial window, as it provides the initial context $T_{1,w}$ to predict the first forecast horizon h.
2. Sliding-window forecasting: We slide a window of size w across the series with step size s. At each position, the model generates forecasts for horizon h based on the current context window. Forecasts are produced for all time steps beyond the warm-up window (see Figure 3). Importantly, the data in the context (observed) window are always true values; forecasts do not overwrite or replace observed data.
3. Handling overlapping forecasts: Depending on the forecast horizon h and a step size s, some future time steps may fall within multiple forecasted intervals if $s < h$. In such cases, the final forecast for a time step is computed as the average of all predictions generated for that timestamp:

$$\hat{t}_i = \frac{1}{k} \sum_{j=1}^{k} \hat{t}_i^{(j)},$$

where k is the number of overlapping forecasts for time step i, and $\hat{t}_i^{(j)}$ is the j-th forecast for that step.

4. Residual computation: Once forecasts are obtained, residuals are calculated as

$$r_i = |t_i - \hat{t}_i|,$$

quantifying the deviation from the expected temporal pattern and serving as input to residual-based anomaly detection methods.

This approach ensures that forecasts are generated for all time steps after the initial warm-up window, enabling robust residual computation and anomaly detection while leveraging the sequential modeling strengths of transformer-based LMs.

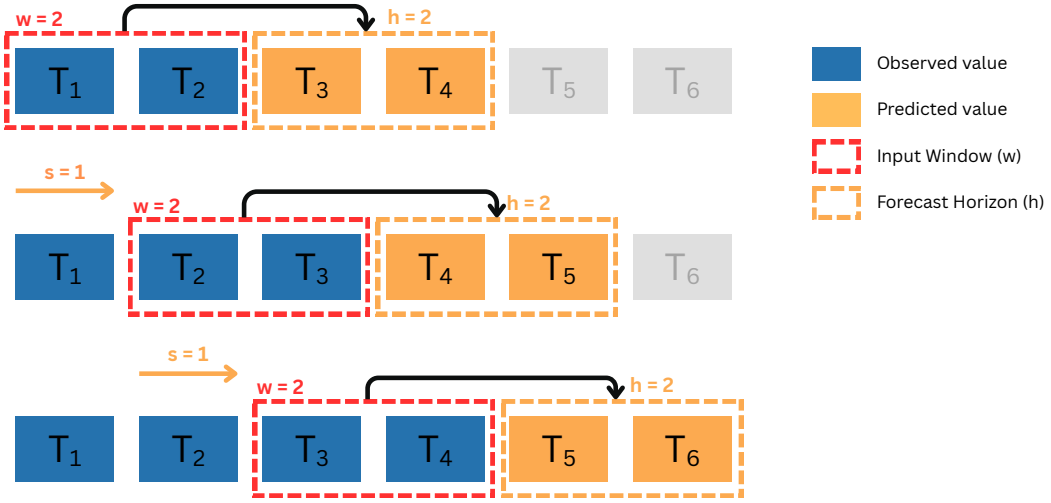

Figure 3. Sliding window approach for time-series forecasting. The diagram shows how observed values (blue) are used to generate predicted values (orange). The process is defined by three parameters: the window size (w) for input data, the horizon (h) for the prediction length, and the step size (s), which specifies how many time steps forward the window moves between consecutive forecasts.

3.3. Time Series Anomaly Detection

We apply the TSAD procedure described in Algorithm 1, which relies on a specific anomaly detection method m and a method-specific thresholding function $f_m(\cdot)$. Each method transforms the residuals, applies its thresholding function, and labels as anomalies the points exceeding the threshold. Standard statistical measures such as standard deviation (STD), interquartile range (IQR), median absolute deviation (MAD), and Robust Z-Score were computed following standard definitions [28]. Methods specific to anomaly detection, including Percentile-Based Filter (PBF), Extreme Value Detection, Relative Difference (RD), and Consensus-Based Detection, were implemented as described below, with some threshold parameters selected based on preliminary experiments to balance sensitivity, robustness, and computational efficiency.

Algorithm 1 Unified Anomaly Detection Framework

Require: Residuals $R = [r_1, \ldots, r_n]$, detection method m

$R' \leftarrow \text{Transform}(R, m)$ ▷ Preprocessing step

$threshold \leftarrow f_m(R')$ ▷ Thresholding function

$\mathcal{E}_m \leftarrow \{i \mid R'_i > threshold\}$ ▷ Mark it as anomaly

return $\mathcal{E}_m$

3.4. Residual Functions

3.4.1. Standard Deviation (STD)

The standard deviation-based method operates in the log space to handle heavy-tailed residual distributions. Residuals are first transformed as

$$R' = \log_{10}(R + 10^{-10}).$$

The anomaly threshold is computed as

$$threshold = \mu_{R'} + 2\sigma_{R'},$$

where $\mu_{R'}$ and $\sigma_{R'}$ denote the mean and standard deviation of the transformed residuals.

3.4.2. Interquartile Range (IQR)

The interquartile range method aims to improve the robustness of the estimate to moderate deviations and outliers. Let Q_1 and Q_3 denote the 15th and 85th quantiles of R, with $IQR = Q_3 - Q_1$. The detection threshold is defined by

$$threshold = Q_3 + 2.5 \cdot IQR.$$

Residuals are smoothed using a three-point moving average prior to thresholding.

3.4.3. Median Absolute Deviation (MAD)

The MAD-based method computes

$$median = \text{median}(R), \quad MAD = b \cdot \text{median}(|R - median|),$$

where b = 1.4826 and the dynamic threshold is defined as follows:

$$threshold = median + 4 \cdot \log\left(1 + \frac{MAD}{median}\right) \cdot MAD.$$

3.4.4. Percentile-Based Filter (PBF)

The percentile-based filter computes a dynamic upper bound

$$threshold = \text{percentile}(R, p), \quad p \in [99.5, 99.9],$$

adjusted automatically based on the ratio of the 99th to 99.9th percentiles.

3.4.5. Robust Z-Score

The robust z-score is calculated as

$$z_i = \frac{|r_i - median(R)|}{MAD(R) + 10^{-10}},$$

with anomalies flagged when $z_i > 3.5$.

3.4.6. Extreme Value Detection

Residuals are analyzed in the log space:

$$R' = \log_{10}(R + 10^{-10}), \quad threshold = median(R') + 4k \cdot MAD(R'),$$

where $k = 1 + \frac{\text{kurtosis}(R)}{10}$ adjusts for tail heaviness.

3.4.7. Relative Difference (RD)

For time series with strong trends, anomalies are detected when

$$\frac{r_i}{\hat{t}_i} > 0.3,$$

i.e., residuals exceed 30% of the predicted value.

3.4.8. Consensus-Based Detection

To aggregate the outputs of individual detection methods, we employ a weighted consensus-based approach. In this framework, each detector identifies potential anomalies, and the final decision is reached based on the weighted agreement across the ensemble. A final anomaly is signaled only if this sum exceeds a predefined threshold τ. The comprehensive procedure for this consensus mechanism is detailed in Algorithm 2.

To account for varying levels of detector reliability, we assign a weight w_m to each method based on its empirical performance across benchmarks. These weights, alongside the decision threshold τ, are as follows:

- Standard Deviation: 3.0
- Percentile: 3.0
- Interquartile Range (IQR): 2.0
- Relative Difference: 2.0
- Z-score: 1.0
- Extreme Value: 0.5
- Median Absolute Deviation (MAD): 0.5
- Threshold (τ): 6.0

Algorithm 2 Consensus-Based Anomaly Detection

Require: Detected index sets $\mathcal{E}_m$, method weights w_m, threshold τ
1: Initialize consensus set $\mathcal{E}_{\text{cons}} \leftarrow \emptyset$
2: **for** each data point i **do**
3: $\quad vote\_score \leftarrow \sum_m w_m \cdot \mathbf{1}_{\{i \in \mathcal{E}_m\}}$
4: $\quad$ **if** $vote\_score \geq \tau$ **then**
5: $\quad\quad \mathcal{E}_{\text{cons}} \leftarrow \mathcal{E}_{\text{cons}} \cup \{i\}$
6: $\quad$ **end if**
7: **end for**
8: **return** $\mathcal{E}_{\text{cons}}$

The threshold in the vote score is chosen to ensure that a data point is considered anomalous only if multiple methods agree on it, reducing the impact of false positives from any single method while capturing points with strong consensus.

4. Experimental Settings

4.1. Datasets

We evaluate our methodology on 492 series belonging to three widely used benchmark collections, namely the Numenta Anomaly Benchmark (NAB) https://github.com/numenta/NAB/tree/master/data latest access: 6 January 2026 , the Yahoo Webscope S5 dataset https://webscope.sandbox.yahoo.com latest access: 1 November 2025, and the NASA MSL and SMAP datasets https://www.kaggle.com/datasets/patrickfleith/nasa-anomaly-detection-dataset-smap-msl latest access: 28 November2025. These datasets cover diverse domains and include both point-wise and subsequence anomalies, making them suitable for comprehensive benchmarking.

- NAB contains 58 univariate time series from domains such as server metrics, network traffic, and IoT sensors. Each series includes labeled point anomalies and exhibits challenges such as seasonality, concept drift, and noise.
- Yahoo S5 includes 367 univariate time series across four subsets, representing a mixture of real and synthetic data. Labeled anomalies include point-wise, collective, and contextual anomalies [5], capturing spikes, drops, and behavioral shifts.
- NASA (MSL & SMAP): comprises the Mars Science Laboratory (MSL) and Soil Moisture Active Passive (SMAP) datasets. Each series contains univariate sensor readings with labeled anomalies representing complex system failures. These datasets were introduced and used in [29] for spacecraft anomaly detection.

To ensure reliable evaluation, we run separate tests on the benchmark datasets by excluding the series with flawed annotations, boundary anomalies, or excessive noise before running the experiments according the time series recommendations reported in [30]. The results look comparable to the ones achieved on the original datasets, which are considered by prior works, and do not change the key findings of our research. More details about the time series cleaning process can be found in [31]. A summary of the datasets used in our experiments is reported in Table 1.

Table 1. Summary of the datasets used in the experiments.

Dataset	Number of Series	Average Number of Anomalies per Series	Series Length ($Mean \pm Std$)
NASA	80	1.28	8715 ± 3121
Yahoo S5	367	5.87	1562 ± 140
NAB	45	2.09	6088 ± 3150

4.2. Language Models

We compare the performance of (1) a LM pretrained for TSF, i.e., Chronos [7], (2) three transformer-based language models fine-tuned for comparison: *T5-Large* https://huggingface.co/amazon/chronos-t5-large latest access: 1 November 2025, *BART* https://huggingface.co/facebook/bart-large latest access: 1 November 2025, and *GPT-2* https://huggingface.co/openai-community/gpt2 latest access: 1 November 2025. Among these, only T5-Large had been pretrained within the Chronos framework [7] on large-scale, temporally diverse time series data. In contrast, BART and GPT-2 were not pretrained for time series forecasting, serving instead as baselines to evaluate the impact of domain-specific pretraining. All the tested LMs have less than 1 Billion parameters (T5-large: 770 million, BART-large: 406 million, GPT-2 large: 762 million). This setup allowed for a controlled comparison of architectures with distinct pretraining histories under a unified fine-tuning protocol. More details about the tested models are given below:

1. T5-large is a transformer-based encoder–decoder language model suitable for sequence-to-sequence tasks. Its encoder–decoder architecture allows it to capture complex temporal dependencies in time series data. We fine-tuned a Chronos-pretrained T5-large checkpoint on the target datasets, leveraging prior knowledge from large-scale time series. Fine-tuning enables the model to adapt to dataset-specific patterns, improving forecast accuracy and supporting residual-based anomaly detection.
2. GPT-2 is an autoregressive transformer-based language model originally developed for natural language generation. Its architecture models sequential dependencies, making it applicable to time series forecasting tasks once adapted to numerical sequences. Although it was not pretrained for time series data, its ability to model long-range dependencies enables it to capture temporal dynamics effectively.

3. BART is an encoder–decoder transformer architecture that combines bidirectional encoding with autoregressive decoding. While originally designed for text-to-text generation tasks, its sequence modeling capabilities make it suitable for forecasting applications. Unlike T5-Large, BART was not pretrained within the Chronos framework, but it serves as a useful baseline to assess the benefits of domain-specific pretraining.

4.3. Evaluation Metrics

We fine-tune all models using a 2-fold cross-validation setup [32], ensuring that each model is exclusively evaluated on unseen data during inference and minimizing the risk of data leakage. The fine-tuning procedure was kept consistent across models to maintain comparability. All models are trained using the Adam optimizer with standard hyperparameter settings, ensuring stable convergence and generalization performance.

To assess TSAD performance we adopt the F1 score, which averages precision and recall, and is well-suited for evaluating anomalies that may span multiple timestamps [33]. F1 score is particularly appropriate because time series anomalies are often rare and imbalanced, and both false positives and false negatives are deemed as relevant. In particular, our evaluation considers both partial and complete identification of anomalous events as correct detections. This approach accounts for the inherent uncertainty in anomaly boundaries and provides a fair assessment of the ability of the model to capture abnormal patterns in the time series.

4.4. Parameter Settings

For the Chronos T5-large model, we explored multiple configurations of context window size (w), forecast horizon (h), and step size (s). These ranges were chosen based on preliminary experiments on a validation set to balance forecast accuracy, anomaly detection sensitivity, and computational efficiency. For example, forecast horizons $h \in \{16, 32, 64\}$ were selected to cover short-term to medium-term predictions, which are relevant to capture anomalies at different temporal scales. Similarly, context window sizes w and step sizes s were selected to provide sufficient historical context while keeping computational costs manageable.

All experiments were conducted under controlled conditions to ensure reproducibility and consistency across datasets and models. Each time series is segmented into overlapping windows of length w (context window) to predict the next h steps (forecast horizon). The step size s determined the frequency of forecast generation.

We explored a grid of parameter values:

$$w \in \{32, 64, 128, 256\}, \quad h \in \{16, 32, 64\}, \quad s \in \{16, 32, 64\}.$$

Unless otherwise specified, the default configuration was $w = 256$, $h = 64$, and $s = 16$.

Warm-up window and overlapping forecasts

The procedure for generating forecasts was as follows:

1. The first window served as a warm-up period, providing the initial context for the model without generating forecasts.
2. Subsequent windows slid across the series by step size s, potentially producing overlapping forecasts.
3. For timestamps with overlapping forecasts, the final prediction was computed as the average of overlapping values.

4.5. Hardware and Computational Resources

All experiments were executed on a workstation equipped with an NVIDIA RTX A6000 GPU, 125 GB of system memory, and an Intel(R) Core(TM) i9-10980XE CPU @ 3.00 GHz. The total computational budget for all experiments was approximately 370 h, including both GPU and CPU computations.

4.6. Reproducibility

All codes and dataset preprocessing scripts are publicly available to ensure reproducibility. The resources can be accessed only at our GitHub repository: https://github.com/aliyassine26/VIGI (accessed on 6 January 2026).

5. Experimental Results

This section compares the TSAD abilities of LMs, LLMs, and baseline methods across multiple datasets. To perform a fair cost–benefit analysis, Table 2 shows the F1 scores, inference time, memory usage, and cost for various models evaluated on the Yahoo S5 dataset. Fine-tuning the Chronos pipeline substantially improves performance, with the T5-Large variant achieving an F1 of 0.730 compared to 0.679 when pretrained. Importantly, the GPT2 variant of Chronos performs roughly on par (F1 = 0.692), indicating that the effectiveness comes from the pipeline design rather than the choice of T5-Large as the underlying model. Since Chronos pretraining is available only for T5-based models, GPT and BART variants are fine-tuned without pretraining. Lightweight Chronos models demonstrate efficient inference and low memory usage compared to large LLMs.

Table 3 presents the aggregated performance of the Chronos variants on the Yahoo S5 dataset, summarizing Precision, Recall, and F1 scores across all subsets. This aggregation highlights the consistent benefits of fine-tuning over zero-shot settings and complements the detailed per-model results reported in Table 2.

Table 2. Evaluations of the LM/LLM effectiveness (F1 score) and efficiency (Inference Time, Cost, Memory) on the Yahoo S5 Dataset. The inference times are averaged over the input series and measured via controlled experiments. For all models we reused the same tokenization/preprocessing steps to ensure a fair comparison. For the cost of proprietary models we considered the experimental billing measured by the official service. The hardware settings used in our research are described In Section 4.5.

Model	Approach	Model	Setting	F1	Inference Time (s)	Cost ($)	Memory (GB)
	Pretrained	Chronos *	T5-Large	0.679	57	–	2.5
LMs	Fine-tuned	Chronos *	T5-Large	**0.730**	120	–	2.5
	Fine-tuned	Chronos *	Bart-Large	0.421	83	–	1.5
	Fine-tuned	Chronos *	GPT2	0.692	34	–	1.1
LLMs	Pretrained	LLaMA3-8B [27]	Zero-shot	0.018	36	–	22
	Fine-tuned	LLaMA3-8B [27]	LoRA	0.039	36	–	22
	Pretrained	PROMPTER Mistral-7B [17]	Zero-shot	0.283	46	–	30
	Pretrained	PROMPTER GPT-3.5-turbo [17]	Zero-shot	0.143	4	1.69	–
	Pretrained	DETECTOR Mistral-7B [17]	Zero-shot	0.534	280	–	30
	Pretrained	LLMAD Llama-3-70B [34]	AnoCoT (Few-shot)	0.490	–	–	–
	Pretrained	LLMAD GPT-3.5 [34]	AnoCoT (Few-shot)	0.204	–	–	–
	Pretrained	LLMAD GPT-4 [34]	No CoT (Few-shot)	0.621	–	–	–
	Pretrained	LLMAD GPT-4 [34]	CoT (Few-shot)	0.645	–	–	–
	Pretrained	LLMAD GPT-4 [34]	AnoCoT (Few-shot)	0.724	14	0.260	–
	Pretrained	Gemini-1.5-Flash [26]	Zero-shot Vision	0.317	1.9	0.001	–
	Pretrained	GPT-4o-Mini [26]	Zero-shot Vision	0.513	2.1	2.146	–
	Pretrained	Internvlm-76B [26]	Zero-shot Text-S0.3-PAP	0.534	6	0.017	–

* Indicates the best-performing TSAD method. – Indicates unavailable or non-applicable entries.

Open-source zero-shot LLMs, such as LLaMA3 and PROMPTER, achieve poor F1 scores, indicating that generic language models struggle to detect domain-specific anomalies without fine-tuning. Instead proprietary LLMs, including GPT-4 AnoCoT and GPT-4o-Mini, perform better, but high model size or license type does not guarantee top-quality performance; for example, PROMPTER GPT underperforms despite being a proprietary model. Methodological choices, such as prompt design, few-shot setups, or fine-tuning, are crucial to achieve strong performance.

While proprietary models can achieve competitive F1 score, they often require substantial computational resources and monetary cost, which may limit practical deployment. In contrast, the Chronos pipeline, whether using T5-Large or GPT2, balances competitive detection performance with low memory footprint and fast inference, demonstrating the potential of lightweight, task-adapted models for real-world applications.

Table 3. Aggregated performance of Chronos variants on the Yahoo S5 dataset. Results are reported as *Mean* ± *Std* across all subsets for Precision, Recall, and F1 score.

Method	Precision	Recall	F1 Score
Chronos Zero-shot	0.743 * ± 0.175	0.716 ± 0.155	0.679 * ± 0.134
Chronos Fine-tuned	**0.804 ± 0.133**	**0.7165 ± 0.162**	**0.730 ± 0.168**
GPT Fine-tuned	0.775 * ± 0.178	0.692 ± 0.162	0.692 * ± 0.183
BART Fine-tuned	0.455 * ± 0.206	0.538 * ± 0.250	0.421 * ± 0.207

* Indicates statistical significance at $p < 0.05$ compared to Chronos Fine-tuned using a paired t-test. Note: The best-performing method per subset was selected prior to aggregation.

5.1. Further Experiments

Table 4 presents F1 scores of various models on the Yahoo S5, NASA, and NAB datasets, including conventional statistical methods (e.g., ARIMA, MP), deep learning approaches (e.g., LSTM AE, VAE, TadGAN), specialized LLM-based anomaly detection methods (PROMPTER, DETECTOR), and the Chronos pipeline with T5-Large (pretrained and fine-tuned).

Traditional statistical and deep learning models achieve moderate performance across datasets. Methods such as AER and LSTM DT are generally consistent but show variability depending on dataset's characteristics. Zero-shot LLM-based approaches, including PROMPTER and DETECTOR, demonstrate highly variable F1 scores; for example, DETECTOR performs fairly well on Yahoo S5 and NASA datasets but poorly on some NAB subsets. Proprietary models like PROMPTER GPT, despite their large number of model parameters, underperform in several cases, highlighting that the methodology, rather than model size or licensing, determines success.

Our pipeline, including T5-Large, achieves competitive F1 scores across all datasets. Fine-tuning improves performance in Yahoo S5 and NAB datasets, particularly on subsets A2, A3, and AdEx, while pretrained variants perform better on certain NASA subsets (e.g., MSL and SMAP). These results indicate that dataset characteristics strongly influence the relative effectiveness of pretrained versus fine-tuned models.

Overall, the empirical evaluation demonstrates that task-specific adaptation and careful pipeline design are more critical than model scale or license type. Lightweight models like Chronos provide robust performance across heterogeneous time series datasets, offering an efficient and practical alternative to large proprietary LLMs, which may incur high computational costs and variability in effectiveness.

Table 4. F1 scores of different models on Yahoo S5, NASA, and NAB datasets.

Model	Yahoo S5			NASA			NAB				*Mean* ± *Std*
	A1	A2	A3	A4	MSL	SMAP	Art	AWS	AdEx	Traf	
AER	**0.799**	**0.987**	**0.892**	0.709	**0.587**	**0.819**	**0.714**	**0.741**	0.690	**0.703**	**0.764 ± 0.108**
LSTM DT	0.728	0.985	0.744	0.646	0.471	0.726	0.400	0.468	0.786	0.585	0.654 ± 0.168
ARIMA	0.728	0.856	0.797	0.686	0.525	0.411	0.308	0.382	0.727	0.467	0.589 ± 0.183
MP	0.507	0.897	0.793	**0.825**	0.474	0.423	0.571	0.440	0.692	0.305	0.593 ± 0.188
TadGAN	0.578	0.817	0.416	0.340	0.560	0.605	0.500	0.623	0.818	0.452	0.571 ± 0.149
LSTM AE	0.595	0.867	0.466	0.239	0.545	0.662	0.667	**0.741**	0.500	0.500	0.578 ± 0.163
VAE	0.592	0.803	0.438	0.230	0.494	0.613	0.667	0.689	0.583	0.483	0.559 ± 0.150
AT	0.571	0.565	0.760	0.576	0.400	0.266	0.414	0.430	0.500	0.371	0.485 ± 0.132
MAvg	0.713	0.356	0.647	0.615	0.171	0.092	0.222	0.408	**0.880**	0.157	0.426 * ± 0.259
MS Azure	0.280	0.653	0.702	0.344	0.051	0.019	0.056	0.112	0.163	0.117	0.250 * ± 0.235
PROMPTER MISTRAL [17]	0.194	0.235	0.338	0.336	0.160	0.154	0.370	0.268	0.000	0.135	0.219 * ± 0.108
PROMPTER GPT [17]	0.143	0.078	0.157	0.195	0.049	0.110	0.154	0.194	0.133	0.133	0.135 * ± 0.044
DETECTOR [17]	0.615	0.828	0.376	0.363	0.429	0.431	0.400	0.362	0.727	0.480	0.501 ± 0.157
Chronos Pretrained	0.543	0.789	0.747	0.570	0.553	0.508	0.092	0.238	0.387	0.428	0.485 ± 0.201
Chronos Fine-tuned	0.533	0.866	0.874	0.647	0.510	0.389	0.346	0.141	0.392	0.509	0.521 ± 0.217

* Indicates statistical significance at $p < 0.05$ compared to Chronos Fine-tuned using a paired t-test. Note: Scores represent mean values; the final column reports aggregate *Mean* ± *Std* across all datasets.

5.2. Benchmarking Anomaly Detection Methods

Table 5 presents a comparison of multiple anomaly detection methods integrated within the Chronos pipeline, evaluated on Yahoo S5, NASA, and NAB datasets. Both pretrained and fine-tuned T5-Large models are considered, allowing us to observe the impact of model adaptation on different AD techniques.

Table 5. Comparison of anomaly detection methods using Chronos T5-Large Pre-Trained (PT) & Fine-Tuned (FT) on the Yahoo S5, NASA, and NAB datasets. Results are reported as F1 scores.

Method	Model	Yahoo S5			NASA			NAB				*Mean* ± *Std*
		A1	A2	A3	A4	MSL	SMAP	Art	AWS	AdEx	Traf	
Consensus	FT	0.505	0.859	0.828	0.609	0.329	0.378	0.056	0.151	0.246	0.409	0.437 ± 0.268
	PT	0.528	0.810	0.777	0.586	0.386	0.407	0.000	0.241	0.234	0.373	0.434 ± 0.250
Extreme	FT	0.000	0.000	0.000	0.000	0.041	0.065	0.000	0.000	0.000	0.000	0.011 * ± 0.023
	PT	0.000	0.000	0.000	0.000	0.075	0.140	0.000	0.000	0.000	0.000	0.022 * ± 0.048
IQR	FT	0.459	0.774	0.558	0.324	0.186	0.334	0.010	0.123	0.303	0.402	0.347 * ± 0.220
	PT	0.505	0.630	0.213	0.107	0.195	0.326	0.007	0.199	0.303	0.404	0.289 * ± 0.187
MAD	FT	0.139	0.164	0.294	0.210	0.247	0.206	0.011	0.020	0.081	0.075	0.145 * ± 0.096
	PT	0.140	0.231	0.394	0.261	0.271	0.249	0.022	0.023	0.076	0.086	0.175 * ± 0.124
Percentile	FT	0.533	0.577	0.520	0.482	0.509	0.375	0.000	0.140	**0.392**	0.434	0.396 ± 0.186
	PT	0.540	0.593	0.570	0.484	**0.553**	0.427	0.000	0.150	0.386	0.428	0.413 ± 0.194
RD	FT	0.224	0.345	0.263	0.160	0.353	0.389	**0.251**	0.113	0.033	0.193	0.232 * ± 0.113
	PT	0.224	0.166	0.172	0.115	0.494	**0.508**	0.092	0.150	0.032	0.205	0.216 * ± 0.160
STD	FT	0.495	**0.866**	**0.874**	**0.647**	0.207	0.327	0.056	0.141	0.282	**0.509**	**0.440 ± 0.289**
	PT	**0.542**	0.789	0.747	0.570	0.211	0.360	0.000	**0.238**	0.275	0.422	0.415 ± 0.249
Z-score	FT	0.182	0.412	0.602	0.341	0.221	0.191	0.016	0.036	0.115	0.123	0.224 * ± 0.181
	PT	0.198	0.573	0.676	0.410	0.263	0.214	0.022	0.041	0.089	0.126	0.261 * ± 0.224

* Indicates statistical significance at $p < 0.05$ compared to Standard Deviation Fine-tuned using a paired t-test. Note: Scores represent mean values; the final column reports aggregate *Mean* ± *Std* across all datasets.

The results reveal significant performance variance rooted in the underlying statistical assumptions of each method and the structural nature of the datasets. For instance, Standard Deviation (STD) and Percentile methods consistently outperform others on the Yahoo S5 dataset. This is likely because Yahoo S5 contains many synthetic and real-world anomalies with high magnitudes, which easily exceed global or moving distribution thresholds. In contrast, the Extreme and MAD methods perform poorly across most benchmarks. As shown in the implementation, Extreme utilizes a log-scale transformation and a kurtosis-based penalty, while MAD employs an adaptive spread factor. While these designs are intended to make the detectors robust against heavy-tailed noise, they often result in overly conservative thresholds that fail to trigger on lower-amplitude or contextual anomalies,

particularly in the NAB dataset where anomalies are often subtle temporal shifts rather than spikes.

Furthermore, we observe that the Relative Difference (RD) method achieves its peak performance on the NASA dataset, outperforming even the high-performing STD method in the pretrained setting. NASA datasets consist of spacecraft telemetry where anomalies often manifest as a sudden change in the relationship between variables or a shift in state, rather than absolute value violations. The RD method's focus on residuals relative to the local signal scale makes it better-suited for these level-shift anomalies. Conversely, IQR and Z-score benefit significantly from model fine-tuning on the Yahoo and NAB datasets, suggesting that as the model becomes more confident in its temporal predictions, the resulting residual distributions become more Gaussian, allowing these distribution-based estimators to define clearer boundaries.

Overall, these results suggest that no single anomaly detection method universally dominates. Effectiveness depends on dataset characteristics, anomaly types, and temporal patterns. The Chronos pipeline's flexibility allows practitioners to integrate and select TSAD methods that best suit the data distribution and task requirements, emphasizing the importance of both method selection and task-specific model adaptation.

Ablation Study

We conducted an ablation study to assess the sensitivity of the proposed method to the main TSAD parameter. For the sake of brevity, hereafter we will report an in-depth analysis of the two parameters, which, according to our experiments, mostly influence TSAD performance, i.e., the standard deviation multiplier (used in the Standard Deviation method) and the consensus weighted threshold (used in the Consensus Weighted method). Each parameter is independent and controls the sensitivity of its respective method. The study evaluates the impact of these parameters on the mean F1 score across Yahoo S5, NASA, and NAB datasets.

All parameter values were set a priori on the pretrained model based on preliminary experiments and kept consistent across all datasets. No separate validation folds were used for tuning these thresholds, demonstrating that the methods perform robustly with uniform settings. Figure 4 shows the results: the left-hand side plot depicts how varying the standard deviation multiplier affects mean F1 scores, while the right-hand side one shows the effect of different consensus weighted thresholds. Although the trends differ slightly across datasets, the selected thresholds provide consistently reliable performances.

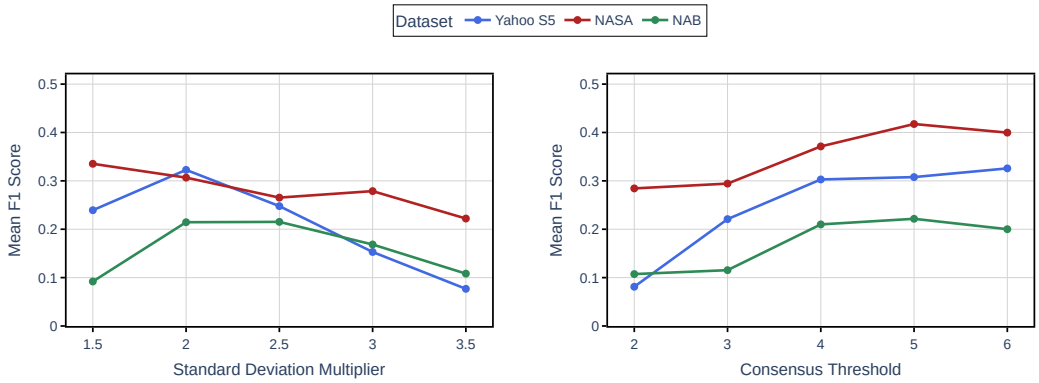

Figure 4. Ablation study on anomaly detection method parameters using the pretrained Chronos-T5 model. Left: Mean F1 score versus standard deviation multiplier. Right: Mean F1 score versus consensus weighted threshold. Results are shown for Yahoo S5, NASA, and NAB datasets.

Overall, this study demonstrates that careful selection of anomaly detection method parameters is important. The consistent trends across datasets justify the choice of uniform thresholds, providing reliable performance while balancing sensitivity and false positives.

5.3. Comparison of Fine-Tuning Strategies and Computational Efficiency

To analyze the computational demand, we compare fine-tuning pretrained language models (LMs) with the LLMs used in prior work [13]. The idea behind this, confirmed by previously reported experiments, is that the fine-tuning step allows LMs to learn the specific patterns hidden in time series data, thus boosting anomaly detection performance. Dong et al. [27] fine-tuned the Meta-LLaMA-3-8B-Instruct model using the parameter-efficient LoRA method [35], updating only the linear layers of the transformer through low-rank adaptation. Their setup involved 1000 synthetic sequences generated by TTGenerator, covering multiple anomaly types and sequence lengths, and was trained for a single epoch. Despite the parameter-efficient approach, fine-tuning LLaMA3-8B required approximately two hours on two NVIDIA H100 GPUs (80 GB each), consuming around 130 GB of GPU memory.

In contrast, our experiments used full-parameter fine-tuning on smaller language models (T5-Large, BART-Large, GPT-2) with K-fold cross-validation on time series datasets [32]. The smaller model sizes enabled full fine-tuning within modest hardware constraints, even with cross-validation, while maintaining competitive performance for forecasting and anomaly detection.

Table 6 summarizes the differences in fine-tuning method, datasets' type, training duration, and GPU memory usage for all models. For the Chronos models, one epoch corresponds to 5000 training steps.

Table 6. Comparison of fine-tuning strategies and computational costs for Chronos models (T5-Large, BART-Large, GPT-2) and LLaMA3-8B. Time and memory usage are reported per model.

Model	Method	Dataset	Epochs	Time	Memory (GB)
BART-Large	Full fine-tuning	Yahoo S5	1	1:26:00	12–16
GPT-2	Full fine-tuning	Yahoo S5	1	0:29:28	6–8
Chronos T5-Large	Full fine-tuning	Yahoo S5	1	2:24:51	24–32
LLaMA3-8B	LoRA	Synthetic	1	2:00:00 (2 GPUs)	130

These results indicate that computational demand is primarily determined by model size rather than by ad hoc fine-tuning method. LLaMA3-8B requires substantial memory and multi-GPU training, even with parameter-efficient adaptation, whereas smaller Chronos models achieve full-parameter fine-tuning on modest hardware. This demonstrates the practicality and cost-effectiveness of using appropriately sized LMs for domain-specific time series tasks while maintaining competitive performance relative to very large models.

6. Discussion

The experiments support our initial hypothesis that lightweight, task-adapted language-model pipelines can deliver competitive TSAD results while enhancing cost-effectiveness compared to very large LLMs. The results also confirm that the pipeline design and the TSAD logic, rather than model scale alone, yield most performance gains. In particular, the fine-tuning of small LLMs is beneficial despite being potentially costly.

There are two noteworthy methodological findings. First, stable, interpretable baselines that generalize across heterogeneous datasets are achieved by combining principled, well-calibrated statistical decision rules (such as percentile and IQR thresholds) with pretrained sequence representations. Second, specific fine-tuning of lightweight language

models results in measurable improvements on various benchmarks, suggesting that adapting to temporal distributions and anomaly characteristics is an efficient way to close the gap with larger models.

The contribution corresponds to a number of values per resource. For comparable detection performance in batch offline scenarios, lightweight model pipelines require less memory, which reduces the overall cost. While real-time or streaming deployment introduces engineering and hosting factors (e.g., API latencies, routing, batching) that are complementary to the core algorithmic claims and are best evaluated in follow-up deployment studies, we intentionally focused on offline benchmark evaluation to ensure reproducibility and an unbiased computational comparison.

While previous studies have emphasized the potential of LMs in zero- or few-shot scenarios, our findings provide an alternative option. In particular, these models are valuable tools, but their effectiveness in structured time series tasks depends largely on how they are adapted and integrated into the overall detection process. In many cases, lightweight language models that are properly pretrained or fine-tuned and paired with efficient detection strategies achieve a better balance between performance and cost than large, general-purpose models.

Limitations

Despite the promising results, several limitations should be acknowledged. First, language model-based forecasting methods may inherit biases from their pretraining data and can exhibit hallucination effects, which may influence forecast accuracy and, in turn, residual-based anomaly detection. Second, our evaluation is limited to univariate time series. While this allows for consistent and reproducible comparisons, it does not capture dependencies across multiple variables that are common in many real-world settings. Third, detection is based on forecasting residuals, which can only be computed after the true observations are available. As a result, the proposed approach is not suitable for zero-latency real-time anomaly detection, but it can be applied in offline or near-real-time settings with a short detection delay. Finally, the performance of the pipeline depends on several design parameters, including the context window length, forecast horizon, and step size. Although these parameters were selected based on standard practice, performance may vary with different choices.

7. Conclusions and Future Research Directions

This paper explores the effectiveness of lightweight LMs for solving the TSAD problem. We propose an approach that combines time series forecasting with residual-based anomaly detection. With the aim of balancing detection accuracy and inference cost, we compare LLMs with lightweight LMs and traditional methods.

The experiments indicate that lightweight LMs with a residual-driven TSAD approach outperform LLMs in zero-shot settings. The pretrained version of Chronos-T5 achieves a F1 score equal to 0.679, whereas its fine-tuned version achieves 0.730. Conversely, the proprietary GPT-4o-Mini reaches an F1 score of 0.513 under vision prompt-only inference. Using the same residual-driven approach, DETECTOR Mistral-7B achieves 0.534 but at a much higher inference cost, while Chronos-T5 maintains superior performance. Beyond their competitive performance, lightweight LMs are more practical for real-world TSAD due to their reduced fine-tuning cost. It is worth noting that using LLMs via API can enable faster inference for single-prompt tasks, but this does not guarantee better anomaly detection results and comes at the cost of monetary expense or high local memory requirements for open-source models.

Overall, our study demonstrates that lightweight language models offer a favorable balance between detection performance and computational efficiency, making them suitable for TSAD across a range of applications.

For practical deployments, model selection may be guided by data availability and cost constraints: pretrained Chronos-T5 is recommended when data is limited, whereas fine-tuned versions improve detection accuracy if sufficient data is available. Key parameters, such as residual window size and detection thresholds, should be adjusted for each dataset to maximize accuracy. This lightweight decision framework provides practitioners with an adaptable approach to balance performance and efficiency across diverse TSAD scenarios.

In our future work, we will (1) study new adaptive strategies that combine anomaly detection methods per time series, (2) extend the current research to multivariate and streaming or drift-aware scenarios, (3) release new testing benchmarks where LMs and LLMs can be fairly tested, and (4) incorporate explainable AI techniques to improve accountability and usability of the approach in practical deployments.

Author Contributions: Conceptualization, A.Y., L.C. and L.V.; methodology, A.Y., L.C. and L.V.; software, A.Y.; validation, A.Y. and L.C.; formal analysis, A.Y. and L.C.; investigation, A.Y., L.C. and L.V.; resources, L.C. and L.V.; data curation, A.Y.; writing—original draft preparation, A.Y., L.C. and L.V.; writing—review and editing, A.Y. and L.C.; visualization, A.Y.; supervision, L.C. and L.V.; project administration, L.C. and L.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The research uses data released by third parties. The code and experimental configurations are publicly available at: https://github.com/aliyassine26/vigi (accessed on 6 January 2026).

Acknowledgments: The research was carried out in collaboration with the SmartData@PoliTo intedepartmental center of Politecnico di Torino, partly using the resources offered by the HPC@PoliTo cluster.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Schmidl, S.; Wenig, P.; Papenbrock, T. Anomaly detection in time series: A comprehensive evaluation. *Proc. VLDB Endow.* **2022**, *15*, 1779–1797. [CrossRef]
2. Paoletti, G.; Giobergia, F.; Giordano, D.; Cagliero, L.; Ronchiadin, S.; Moncalvo, D.; Mellia, M.; Baralis, E. MAD: Multicriteria Anomaly Detection of Suspicious Financial Accounts from Billions of Cash Transactions. In Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2, Toronto, ON, Canada, 3–7 August 2025; pp. 4751–4760. [CrossRef]
3. Buccafusco, S.; Cagliero, L.; Megaro, A.; Vaccarino, F.; Loti, R.; Salvatori, L. Learning industrial vehicles' duty patterns: A real case. *Comput. Ind.* **2023**, *145*, 103826. [CrossRef]
4. Lu, Y.; Wu, R.; Mueen, A.; Zuluaga, M.A.; Keogh, E. DAMP: Accurate time series anomaly detection on trillions of datapoints and ultra-fast arriving data streams. *Data Min. Knowl. Discov.* **2023**, *37*, 627–669. [CrossRef]
5. Zamanzadeh Darban, Z.; Webb, G.I.; Pan, S.; Aggarwal, C.; Salehi, M. Deep Learning for Time Series Anomaly Detection: A Survey. *ACM Comput. Surv.* **2024**, *57*, 1–42. [CrossRef]
6. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is All You Need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS 2017), Long Beach, CA, USA, 4–9 December 2017; pp. 6000–6010.
7. Ansari, A.F.; Stella, L.; Turkmen, C.; Zhang, X.; Mercado, P.; Shen, H.; Shchur, O.; Rangapuram, S.S.; Pineda Arango, S.; Kapoor, S.; et al. Chronos: Learning the Language of Time Series. *Trans. Mach. Learn. Res.* **2024**, *10*, 1–25.

8. Jin, M.; Wang, S.; Ma, L.; Chu, Z.; Zhang, J.Y.; Shi, X.; Chen, P.Y.; Liang, Y.; Li, Y.F.; Pan, S.; et al. Time-LLM: Time series forecasting by reprogramming large language models. In Proceedings of the International Conference on Learning Representations (ICLR), Vienna, Austria, 7–11 May 2024.
9. Kang, H.; Kang, P. Transformer-based multivariate time series anomaly detection using inter-variable attention mechanism. *Knowl.-Based Syst.* **2024**, *290*, 111507. [CrossRef]
10. Wang, X.; Pi, D.; Zhang, X.; Liu, H.; Guo, C. Variational transformer-based anomaly detection approach for multivariate time series. *Measurement* **2022**, *191*, 110791. [CrossRef]
11. Tuli, S.; Casale, G.; Jennings, N.R. TranAD: Deep transformer networks for anomaly detection in multivariate time series data. *Proc. VLDB Endow.* **2022**, *15*, 1201–1214. [CrossRef]
12. Naveed, H.; Khan, A.U.; Qiu, S.; Saqib, M.; Anwar, S.; Usman, M.; Akhtar, N.; Barnes, N.; Mian, A. A Comprehensive Overview of Large Language Models. *ACM Trans. Intell. Syst. Technol.* **2025**, *16*, 1–72. [CrossRef]
13. Alnegheimish, S.; Nguyen, L.; Berti-Equille, L.; Veeramachaneni, K. Can Large Language Models be Anomaly Detectors for Time Series? In Proceedings of the 2024 IEEE International Conferencze on Data Science and Advanced Analytics (IEEE DSAA), San Diego, CA, USA, 6–10 October 2024; IEEE: Piscataway, NJ, USA, 2024.
14. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language Models are Unsupervised Multitask Learners. *OpenAI Blog* **2019** *1*, 9.
15. Keogh, E. Time Series Data Mining: A Unifying View. *Proc. VLDB Endow.* **2023**, *16*, 3861–3863. [CrossRef]
16. Gruver, N.; Finzi, M.; Qiu, S.; Wilson, A.G. Large Language Models Are Zero-Shot Time Series Forecasters. In Proceedings of the Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, New Orleans, LA, USA, 10–16 December 2023.
17. Dong, M.; Huang, H.; Cao, L. Large Language Models Can Be Zero-Shot Anomaly Detectors for Time Series. In Proceedings of the 11th IEEE International Conference on Data Science and Advanced Analytics (DSAA 2024), San Diego, CA, USA, 6–10 October 2024.
18. OpenAI. GPT-3.5 Turbo. 2025. Available online: https://platform.openai.com/docs/models/gpt-3.5-turbo (accessed on 13 October 2025).
19. Jiang, A.Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D.S.; de las Casas, D.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; et al. Mistral 7B. *arXiv* **2023**. arXiv:2310.06825. [CrossRef]
20. Xue, H.; Salim, F.D. PromptCast: A New Prompt-Based Learning Paradigm for Time Series Forecasting. *IEEE Trans. Knowl. Data Eng.* **2024**, *36*, 6851–6864. [CrossRef]
21. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* **2020**, *21*, 1–67.
22. Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; Zettlemoyer, L. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020), Online, 5–10 July 2020; pp. 7871–7880. [CrossRef]
23. Zhang, Y.; Gong, K.; Zhang, K.; Li, H.; Qiao, Y.; Ouyang, W.; Yue, X. Meta-Transformer: A Unified Framework for Multimodal Learning. *arXiv* **2023**, arXiv:2307.10802. [CrossRef]
24. Zhou, T.; Niu, P.; Wang, X.; Sun, L.; Jin, R. One Fits All: Power General Time Series Analysis by Pretrained LM. In Proceedings of the 37th International Conference on Neural Information Processing Systems (NeurIPS 2023), New Orleans, LA, USA, 10–16 December 2023; pp. 1–34.
25. Liu, C.; He, S.; Zhou, Q.; Li, S.; Meng, W. Large Language Model Guided Knowledge Distillation for Time Series Anomaly Detection. In Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI 2024), Jeju, Republic of Korea, 3–9 August 2024; pp. 1–9.
26. Zhou, Z.; Yu, R. Can LLMs Understand Time Series Anomalies? In Proceedings of the Thirteenth International Conference on Learning Representations (ICLR 2025), Singapore, 24–28 April 2025.
27. Dong, M.; Huang, H.; Cao, L. Can LLMs Serve As Time Series Anomaly Detectors? *arXiv* **2024**, arXiv:2408.03475. [CrossRef]
28. Kaptein, M.; van den Heuvel, E. *Statistics for Data Scientists: An Introduction to Probability, Statistics, and Data Analysis*; Springer: Cham, Switzerland, 2022. [CrossRef]
29. Hundman, K.; Constantinou, V.; Laporte, C.; Colwell, I.; Soderstrom, T. Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, London, UK, 19–23 August 2018; pp. 387–395. [CrossRef]
30. Wu, R.; Keogh, E.J. Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress. *IEEE Trans. Knowl. Data Eng.* **2023**, *35*, 2421–2429. [CrossRef]

31. Keogh, E. Irrational Exuberance: Why We Should Not Believe 95% of Papers on Time Series Anomaly Detection. 2025. Available online: https://kdd-milets.github.io/milets2021/slides/Irrational%20Exuberance_Eammon_Keogh.pdf (accessed on 12 December 2025).
32. Cawley, G.C.; Talbot, N.L.C. On over-fitting in model selection and subsequent selection bias in performance evaluation. *J. Mach. Learn. Res.* **2010**, *11*, 2079–2107.
33. Tatbul, N.; Salinas, D.; Zhao, F.; Madden, S. Precision and Recall for Time Series. In Proceedings of the Advances in Neural Information Processing Systems, Red Hook, NY, USA, 10–16 December 2018; Volume 31.
34. Liu, J.; Zhang, C.; Qian, J.; Ma, M.; Qin, S.; Bansal, C.; Lin, Q.; Rajmohan, S.; Zhang, D. Large Language Models Can Deliver Accurate and Interpretable Time Series Anomaly Detection. In Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Toronto, ON, Canada, 3–7 August 2025; pp. 4623–4634.
35. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. In Proceedings of the International Conference on Learning Representations (ICLR), Virtual Event, 25–29 April 2022.