École Polytechnique Fédérale de Lausanne
Department of Computer Science
Master in Data Science

Large Language Models for Time Series Anomaly Detection and Explanation

Master Thesis

In partial fulfilment of the requirements for the Master of Science EPFL in Data Science (MSc EPFL)

Submitted by:

Salim Cherkaoui

Supervisor(s):
Kalyan Veeramachaneni
Principal Research Scientist, MIT

Alexandre Alahi
Associate Professor, EPFL

Tutor:
Sarah Alnegheimish
PhD EECS Student, MIT

Cambridge, January 2025

Abstract

Large Language Models (LLMs) have recently demonstrated promising capabilities beyond natural language processing, including applications to time series data. In this thesis, we investigate the use of LLMs for the specific task of time series anomaly detection, focusing exclusively on prompt-based approaches without any fine-tuning. Our study explores not only whether LLMs can detect anomalies in time series through carefully designed prompting strategies, but whether they can provide meaningful, human-interpretable explanations for those anomalies.

This study builds on and extends work by Alnegheimish et al. [2023], which explored the zero-shot capabilities of LLMs for anomaly detection in time series. While their work primarily evaluated detection performance, we further investigate the role of prompt design and model behavior, and especially the interpretability of LLM-generated explanations.

To this end, we develop a framework that converts numerical time series into textual representations and systematically prompts LLMs to identify anomalous points. We evaluate our approach on several well-established benchmark datasets for time series anomaly detection, covering a variety of domains and patterns. In addition, we introduce two simple yet effective synthetic datasets, specifically designed with labeled anomalies, to facilitate a controlled evaluation of the explanation capabilities of LLMs.

Our experiments reveal that while large language models are capable of identifying anomalies, state-of-the-art deep learning models remain superior in detection accuracy. Although LLMs could achieve competitive anomaly detection performance on certain datasets, their primary strength may lie in generating natural language explanations that offer interpretability and insights for the detected anomalies. We discuss the advantages and limitations of using LLMs both as anomaly detectors and as explainers, and suggest potential directions for future research.

Acknowledgements

This master's thesis has been an intense, rewarding, and unforgettable journey into the world of research. I feel incredibly grateful for the support, guidance, and kindness I've received along the way.

First and foremost, I would like to thank Kalyan Veeramachaneni for welcoming me into the DAI Lab and offering me the opportunity to complete my thesis at MIT. Working under his direction has been a real privilege. I am especially thankful for his trust, his vision, and for allowing me to explore freely while always providing valuable guidance and perspective.

I am also deeply thankful to Sarah Alnegheimish, who has been my daily mentor throughout this journey. Her feedback — both technical and personal — helped me grow every week, and I truly appreciate the time, patience, and energy she dedicated to my work.

Special thanks to my EPFL supervisor, Alexandre Alahi, for his support and for making this collaboration between EPFL and MIT possible.

To my labmates in the DAI Lab — thank you for creating such a welcoming and inspiring environment. The weekly Tuesday meetings were especially enriching, and I learned a great deal by listening to and exchanging with all of you.

Finally, I would like to thank my family — my parents, Fouzia and Abdelilah, and my sister Sara — for their endless love, encouragement, and support, even from afar. None of this would have been possible without you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in a wide range of natural language tasks, such as text summarization, translation, and text generation Radford et al. [2019], Brown et al. [2020], Sanh et al. [2022], Chowdhery et al. [2023], Wei et al. [2022]. These models have also shown promising abilities beyond text, including image generation, code synthesis, and more Saharia et al. [2022], Koh et al. [2023]. Due to the inherent sequence modeling capabilities of LLMs, researchers have started exploring them in domains traditionally dominated by specialized models—such as time series analysis.

One particularly critical application of time series analysis is anomaly detection, which plays a key role in industries from finance to healthcare. Irregular patterns or outliers can signal fraud, system failures, or other critical events. Conventional approaches to time series anomaly detection typically involve statistical techniques, deep learning models, or domain-specific heuristics, often requiring large labeled datasets and intensive training.

In this work, we investigate whether LLMs, through carefully designed prompting strategies, can serve as effective tools for time series anomaly detection—without fine-tuning. We also explore an equally important question: whether the explanations generated by LLMs for detected anomalies are coherent, meaningful, and provide actionable insights for practitioners.

Unlike prior studies, which emphasize performance metrics alone, our approach gives equal weight to both detection accuracy and interpretability. In particular, this work builds on and extends the recent study by Alnegheimish et al. [2023], which demonstrated the potential of using large language models as zero-shot anomaly detectors for time series. While their work focused on initial detection capabilities, we further explore structured prompting strategies and emphasize the interpretability of the explanations provided by LLMs. To rigorously evaluate their performance, we introduce a custom synthetic dataset with controlled, labeled anomalies—allowing for a fine-grained analysis of both detection

and explanation quality.

1.1 Contributions

The key contributions of this thesis are as follows:

- We explore the application of large language models for time series anomaly detection, focusing exclusively on prompt-based methods without any fine-tuning.
- We evaluate multiple LLM architectures to analyze how model size and structure impact detection performance and explanation quality.
- We introduce one simple synthetic dataset with explicitly labeled anomalies, designed to facilitate evaluation of the interpretability and explanation abilities of LLMs.
- We conduct experiments on several well-established benchmark time series anomaly detection datasets, covering diverse domains.
- Our evaluation includes both quantitative metrics (F1 score, precision, recall) and qualitative analysis of the explanations provided by the models.

Through these contributions, we aim to assess not only the detection capabilities of LLMs, but also their potential to improve model transparency and trustworthiness through natural language explanations.

1.2 Thesis Organization

The thesis is structured as follows:

- **Chapter 1: Introduction** sets the stage for this work by outlining the motivation, goals, and key contributions. It also provides a brief overview of the thesis structure.
- **Chapter 2: Background and Related Work** provides the necessary foundations for this study. It introduces time series concepts, anomaly detection pipelines, and the use of Transformers and LLMs for time series tasks. This chapter also positions our work within existing literature and presents the high-level approach and contributions.
- **Chapter 3: Methodology** details the two main approaches we explore: the forecasting-based approach and the direct anomaly detection approach. It also introduces our LLM-based pipeline for generating anomaly explanations and the rationale behind our design choices.

- **Chapter 4: Challenges** discusses key technical challenges encountered during the research and presents the strategies we employed to overcome them, including input segmentation, enhanced prompting, few-shot learning, and robustness checks.

- **Chapter 5: LLM for Time Series** focuses on how time series data is prepared for use with LLMs. It explains preprocessing steps such as scaling, quantization, tokenization, and the use of rolling windows, as well as prompting strategies like one-shot and few-shot learning.

- **Chapter 6: Synthetic Dataset for Time Series Anomaly Detection** describes the custom dataset we built to evaluate both detection and explanation capabilities in a controlled environment. It outlines the types of anomalies included and the methodology for generating realistic synthetic time series.

- **Chapter 7: Evaluation** presents our experimental setup, including datasets, metrics, and baselines. It compares the performance of LLM-based approaches to existing methods, evaluates explanation quality, and analyzes model behavior.

- **Chapter 8: Conclusion and Future Directions** summarizes the main findings and contributions of the thesis and proposes potential avenues for future work in leveraging LLMs for time series anomaly detection and interpretability.

Chapter 2

Background and Related Work

2.1 Time series and Anomalies

2.1.1 Time series Format

In the context of this thesis, a time series is a sequence of data points indexed in chronological order. These data points are typically collected at regularly timed intervals, and can be univariate (with a single variable per timestamp) or multivariate (with multiple variables per timestamp).

Each time series is built on timestamps that indicate when observations were recorded. In univariate time series, each timestamp is associated with a single numerical value. In contrast, multivariate time series consist of several values per timestamp, each corresponding to a different variable or feature.

Tables 2.1 and 2.2 illustrate examples of univariate and multivariate time series formats.

Table 2.1: Example of a univariate time series.

Timestamp	**Value**
1609459200	75.2
1609462800	76.1
⋮	⋮
1609545600	80.5

While both types of time series have broad applications, this work focuses exclusively on univariate time series.

Table 2.2: Example of a multivariate time series.

Timestamp	Temp	Pressure	Humidity
1609459200	75.2	1012	60
1609462800	76.1	1010	58
⋮	⋮	⋮	⋮
1609545600	80.5	1008	55

2.2 Anomaly Detection Pipelines

Unsupervised anomaly detection pipelines typically follow a common sequence of stages: preprocessing, modeling, and postprocessing Alnegheimish et al. [2022]. As illustrated in Figure 2.2, the goal is to learn the normal behavior of a time series and identify deviations from that behavior.

Preprocessing steps may include rescaling the data, decomposing signals, or transforming sequences into a more convenient format. Modeling refers to training a model to learn temporal patterns from the time series. Finally, postprocessing computes the difference between the values the model expects and the actual values, flagging large discrepancies as potential anomalies.

A variety of models can be used in this pipeline, ranging from traditional statistical models like ARIMA Box and Jenkins [1970] to more recent machine learning-based approaches. We categorize anomaly detection methods into two primary families: classical statistical methods and modern machine learning approaches.

2.2.1 Classical Statistical Methods

Historically, statistical approaches were among the first to tackle time series anomaly detection. One of the simplest of these is static thresholding, which flags a point if it exceeds a fixed upper or lower limit. While appealingly straightforward, this method fails to capture contextual anomalies and dynamic behaviors.

To improve robustness, techniques such as Statistical Process Control (SPC) Montgomery [2019] were developed, flagging points that violate predefined statistical properties. Regression-based models, like ARMA and ARIMA Box and Jenkins [1970], model the trend, seasonality, and residuals of a time series. Anomalies are then identified by analyzing unexpected deviations in residuals. ARIMA, in particular, addresses non-stationarity through differencing, making it a robust tool for modeling real-world time series.

2.2.2 Machine Learning-Based Methods

More recently, deep learning has revolutionized anomaly detection. Two dominant paradigms have emerged:

Prediction-Based Approaches

In this setting, a model is trained to predict future values based on previous sequences. Anomalies are identified when actual values significantly deviate from predictions. Long Short-Term Memory networks with dynamic thresholding (LSTM-DT) Hundman et al. [2018] are prominent in this category. Other examples include Bayesian Networks and Hierarchical Temporal Memory models.

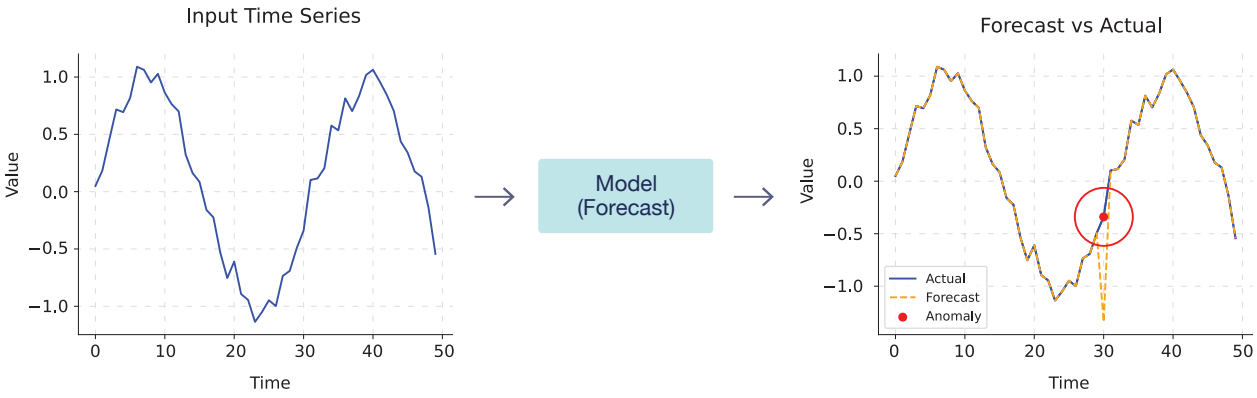

Figure 2.1: Prediction-based anomaly detection pipeline: the left panel displays the input time series, the central block represents the forecasting model, and the right panel shows the forecast overlaid on the actual series, with a red marker highlighting the anomaly.

Reconstruction-Based Approaches

Here, the model attempts to reconstruct the input sequence. If the reconstruction error for certain points is high, those points are considered anomalous. Techniques include LSTM Autoencoders (LSTM-AE), Variational Autoencoders (VAE) Park et al. [2018], and Generative Adversarial Networks (TadGAN) Geiger and El-Yaniv [2020].

These methods follow a general pattern: (1) learn temporal patterns from training data, (2) generate expected outputs, (3) compare with real data, and (4) compute an error signal to detect anomalies.

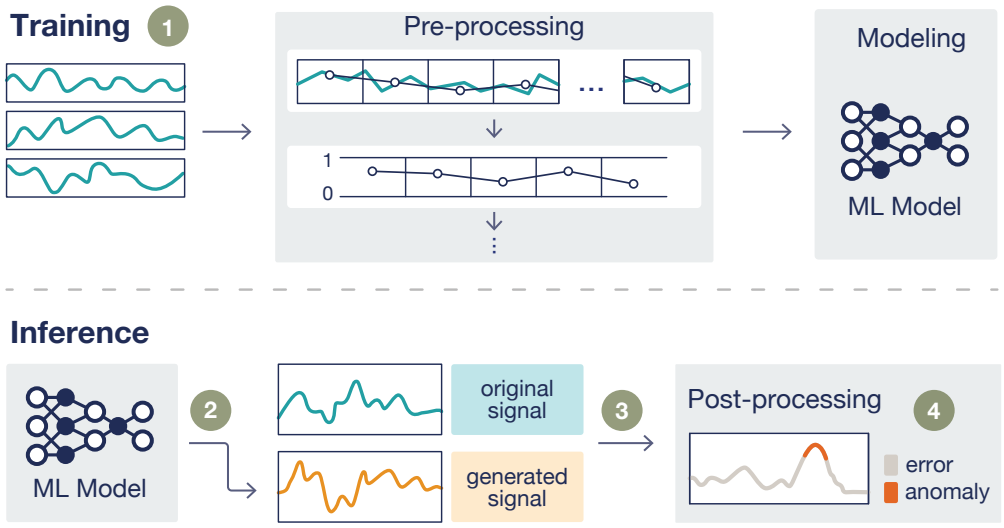

Figure 2.2: Overview of a typical unsupervised anomaly detection pipeline using machine learning. Step 1: The raw time series undergoes a series of preprocessing steps, followed by model training to learn the underlying patterns of normal behavior. (This is often the most computationally intensive stage.) Step 2: The trained model is then used to generate expected values for the input sequence. Step 3: The prediction or reconstruction error is computed by comparing the model's output to the original time series. Step 4: These error values are post-processed to identify points with significant deviations, which are flagged as potential anomalies.

2.3 Transformers for Time Series

Transformers, initially designed for natural language tasks, have shown promise for time series modeling. Their self-attention mechanism allows them to capture long-range dependencies more effectively than RNNs or CNNs.

Several recent works have leveraged transformers for both reconstruction and forecasting tasks in time series:

- **FORECASTPFN** Dooley et al. [2023] uses a pre-trained transformer encoder-decoder architecture on synthetic time series data.
- **TIMEGPT** is trained on a wide variety of real-world time series datasets for general-purpose forecasting.
- **LAG-LLAMA** Rasul et al. [2023] adapts a decoder-only LLAMA-2 model for real time series across diverse domains.
- **CHRONOS** Ansari et al. [2024] parses time series into token sequences, adapting the T5 transformer architecture for pretraining.

While their primary focus is on forecasting, the underlying representations learned by these models can be used for anomaly detection by identifying points where the prediction or reconstruction diverges significantly from the observed values.

2.4 Related Work : LLMs for Time Series

Recent efforts have increasingly explored the use of Large Language Models (LLMs) for time series analysis, particularly for forecasting tasks. However, the application of LLMs to anomaly detection and interpretability remains underexplored. One of the first structured attempts to evaluate LLM capabilities on time series reasoning tasks is the **SigLLM** Alnegheimish et al. [2023] framework.

2.4.1 Forecasting with LLMs

Several works propose transforming forecasting into a language-style task:

- **LLMTIME** Gruver et al. [2023] adapts GPT and LLAMA-2 models for time series forecasting, using windowed sequences formatted as text.
- **PROMPTCAST** Xue et al. [2023] frames forecasting as a prompt-based task, leveraging the zero-shot and few-shot learning capabilities of LLMs.

These approaches demonstrate that pre-trained LLMs can generalize to time series data without additional training, using only carefully crafted prompts.

2.4.2 Foundational Capabilities of LLMs

LLMs, such as GPT Brown et al. [2020] or LLaMA Touvron et al. [2023], fall into the broader category of foundational models—large-scale models pre-trained on diverse data and designed to be adapted to a variety of tasks. These models are capable of learning autoregressive distributions over sequences, enabling tasks like next-token or next-value prediction.

A particularly relevant emergent capability of LLMs is **in-context learning**, where the model performs new tasks simply by being shown examples in the prompt, without any parameter updates. This makes LLMs an attractive candidate for zero-shot or few-shot anomaly detection tasks.

2.4.3 SigLLM: A Framework for Time Series Anomaly Detection with LLMs

SigLLM, introduced by Alnegheimish et al. [2023], is a framework designed to evaluate the ability of large language models (LLMs) to detect anomalies in time series data through prompt-based techniques. It addresses two key challenges: applying LLMs to non-textual time series data, and asking them to identify anomalous *portions* of the input rather than simply generating output or completing sequences.

The SigLLM framework includes a time-series-to-text conversion module, and evaluates two distinct prompting paradigms:

- **Prompter Method:** The LLM is provided with a tokenized version of the time series, and directly prompted to identify which points are anomalous. This setup relies entirely on the model's in-context reasoning abilities.
- **Detector Method:** This approach leverages the forecasting capability of LLMs. The model is used to predict the next value in a sequence window, and anomalies are inferred based on large deviations between the predicted and actual values.

The authors evaluate these methods across 11 datasets and 10 pipeline variations. Their results show that the forecasting-based *Detector* method consistently outperforms the direct *Prompter* approach in terms of F1 score on all datasets. However, both methods fall short of the performance achieved by specialized deep learning models, which still offer superior accuracy (up to 30% higher).

While SIGLLM demonstrates that LLMs can be effective zero-shot anomaly detectors, especially via forecasting, it primarily focuses on detection performance and does not explore explanation generation or interpretability, areas that this thesis seeks to investigate further.

2.5 Positioning and Contributions of This Work

This thesis builds on recent work introduced in SigLLM Alnegheimish et al. [2023], which demonstrated the potential of large language models (LLMs) for time series anomaly detection in a zero-shot setting.

While SIGLLM focused primarily on 0-shot detection performance, our work takes a different approach, emphasizing also the **explanation capabilities** of LLMs and revisiting the potential of the *prompter method*. In particular, we explore whether improved prompting strategies, including carefully structured zero-shot prompts and one-shot examples, can significantly boost the effectiveness of the direct prompting approach.

We hypothesize that, when properly designed, the prompter method could be a powerful and highly practical tool for anomaly detection, especially in settings where interpretability and minimal infrastructure are priorities.

To support this investigation, we design one custom synthetic dataset with fully controlled, labeled anomalies, and construct a structured evaluation to assess:

- Whether LLMs can detect the presence and location of anomalies using only prompt-based inputs (zero-shot and one-shot).
- Whether the natural language explanations they generate align with the ground-truth causes of anomalies.
- How model architecture influence both anomaly detection and interpretability.

2.5.1 Overview of Our Approach

We propose a direct, prompt-based anomaly detection pipeline that avoids fine-tuning and forecasting. Time series are first converted into structured textual formats, which are then used to query the LLM through crafted prompts. While we include a forecasting-based baseline for comparison, we show that forecasting methods often fail to flag anomalies if the model is capable of accurately predicting them—even when they are irregular—thereby weakening their utility in high-risk domains.

In contrast, our focus is on enhancing the capabilities of the prompter method through better prompt engineering and minimal-shot learning. By exploring not only zero-shot but also one-shot examples, we aim to find the true ceiling of what LLMs can achieve in this context, in both detection performance and explanation richness.

2.5.2 Main Contributions

The primary contributions of this thesis are as follows:

- **An improved prompter framework** for anomaly detection in time series, leveraging both zero-shot and one-shot prompting techniques to maximize performance and interpretability without fine-tuning.
- **A critical comparison** between direct prompting and forecasting-based anomaly detection, with evidence supporting the practical advantages of prompting in contexts where interpretability matters.

- **A new synthetic dataset** specifically designed to enable controlled testing of both detection accuracy and explanation quality.
- **An empirical evaluation** across benchmark and synthetic dataset, analyzing the effects of LLM types and prompting strategies on both quantitative and qualitative performance.

Ultimately, this work contributes not only a novel perspective on using LLMs for time series anomaly detection, but also practical insights into how prompting can be engineered to make these models more useful, transparent, and applicable in real-world monitoring scenarios.

Chapter 3

Methodology

In this thesis, we explore the potential of large language models (LLMs) for time series anomaly detection, with a primary focus on prompt-based methods that do not require the model to be fine-tuned. To ground our work in existing research, we first revisit two strategies introduced in the SIGLLM framework Alnegheimish et al. [2023]: the **Prompter** and the **Detector** approaches (Figure 3.1).

The *Prompter* approach directly feeds a time series (converted into a token sequence) into the LLM and asks it to reason over the data to identify anomalous points. This method relies entirely on the model's in-context reasoning abilities to infer deviations from normal behavior.

The *Detector* strategy, in contrast, treats the LLM as a forecasting model. By prompting it to predict the next point in a sequence window, it enables the detection of anomalies through deviations between predicted and observed values.

While both methods highlight the flexibility of LLMs in time series reasoning, our initial experiments reveal that forecasting-based detection might suffer from context fragmentation, particularly for complex anomaly types. These limitations motivated our decision to focus on direct prompt-based anomaly detection techniques, as described in the following sections.

3.1 Forecasting-Based Approach

3.1.1 Setup

In the forecasting-based approach, the LLM predicts the next value in the time series based on a sliding window of previous observations. Anomalies are then identified by calculating the deviation between the predicted and actual values.

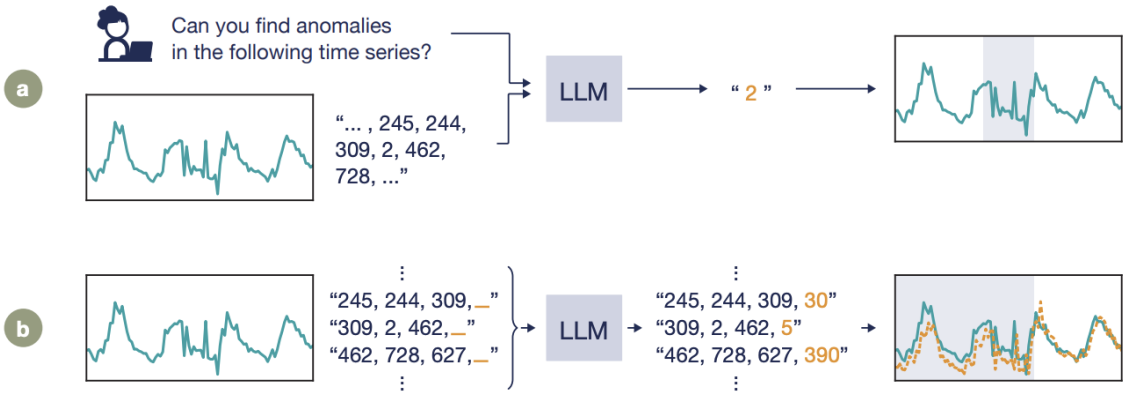

Figure 3.1: Illustration of the two anomaly detection approaches explored in the SIGLLM framework Gruver et al. [2023]. **(a) Prompter**: A direct prompting strategy where the entire time series is tokenized and passed to an LLM, which returns the index or value of an anomaly through reasoning over the sequence. **(b) Detector**: A forecasting-based strategy where the LLM is prompted to predict the next value in a sliding window. Anomalies are identified by comparing forecasted values with actual ones, highlighting large deviations. These two pipelines demonstrate different ways of leveraging LLMs for anomaly reasoning, either through pattern recognition or next-value prediction.

3.1.2 Motivation and Prior Work

This approach aligns well with the autoregressive nature of LLMs and has been shown to perform effectively in prior works, notably SigLLM, where models such as GPT and MISTRAL were used for time series forecasting tasks.

3.1.3 Results and Observations

While the forecasting-based approach initially appeared promising, especially on simpler datasets, we observed several challenges when applying it to anomaly detection:

- The model often performed well in reconstructing time series, including anomalous points, which made deviation-based anomaly identification less effective.
- The forecasting window size proved to be a critical factor; anomalies longer than the input window could not be reliably identified.
- Forecasting-based methods offered limited interpretability, as they focus primarily on numerical predictions without providing explicit explanations for detected anomalies.

These observations are consistent with findings in prior work, and motivated us to investigate alternative approaches.

3.1.4 Limitations of the Forecasting-Based Approach

Reconstruction Challenges with Anomalies To better understand the limitations, we recreated a time series using the same forecasting setup as SigLLM, with a window size of 140 and output size of 1. We also applied ARIMA as a baseline model to illustrate how both simple and complex models can perfectly reconstruct time series, including anomalies, thus reducing the efficacy of deviation-based anomaly detection.

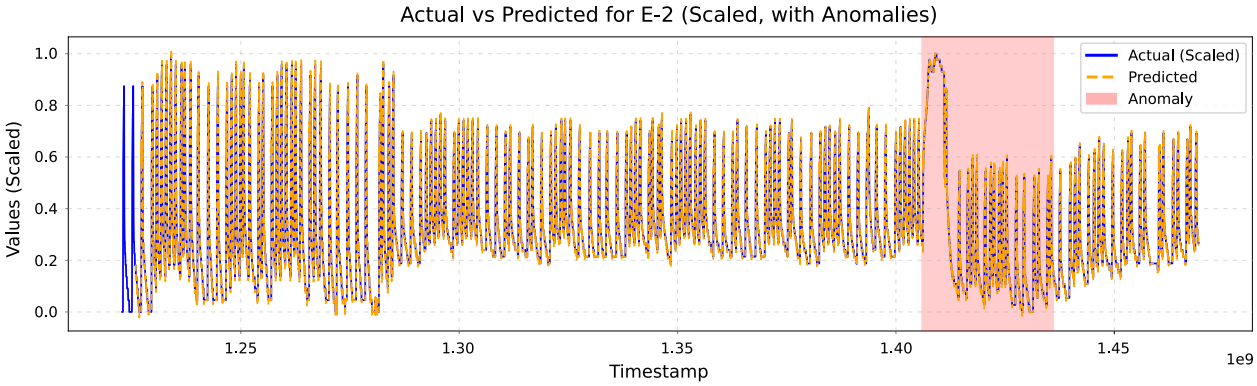

Figure 3.2: Time series reconstruction using ARIMA. Note the perfect reconstruction, including anomalies, undermining the forecasting approach.

Impact of Anomaly Length Another important consideration is the relationship between anomaly length and input window size. Many benchmark datasets contain anomalies that span durations significantly longer than the window size of 140. As a result, the model's input often contains mostly anomalous data, making it difficult for the model to distinguish between normal and abnormal patterns.

Although forecasting-based methods remain a promising direction, especially when combined with fine-tuning and domain-specific adaptations, our focus in this thesis is to investigate whether prompt-based approaches can provide a more interpretable and flexible framework for anomaly detection.

LLMs Prioritize Recent Values in Forecasting Recent studies, such as Liu et al. [2023], have suggested that LLMs tend to focus heavily on the most recent tokens within their context window, often at the expense of earlier information. In cases of time series forecasting, this results in the model primarily relying on the last few observed values to generate the next prediction—effectively ignoring much of the earlier context, especially when the input sequence is long.

This limitation becomes critical when dealing with complex anomalies that unfold gradually or require broader temporal dependencies to detect. In such cases, the model may

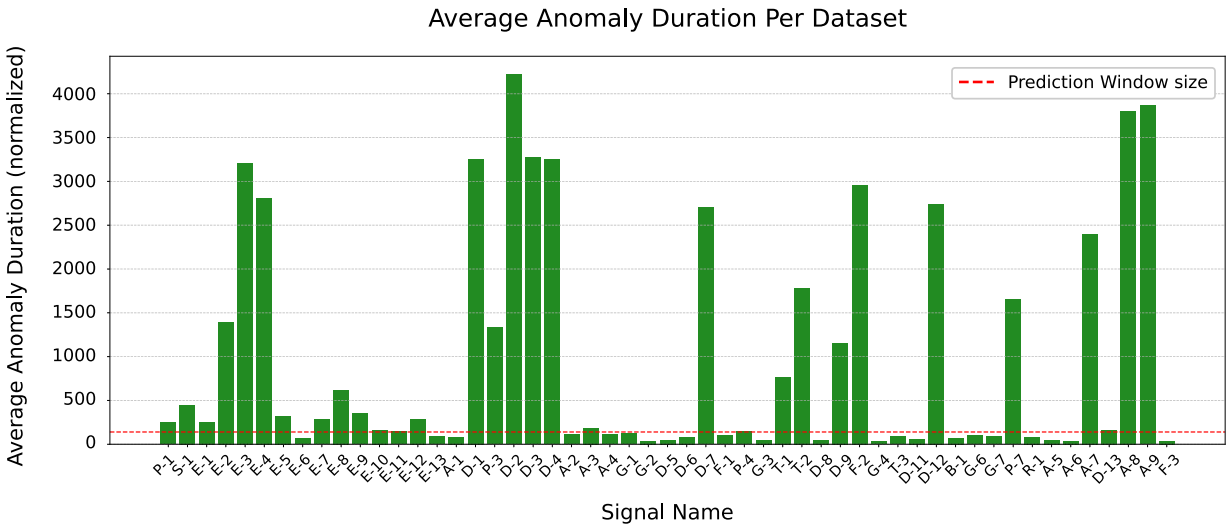

Figure 3.3: Average lengths of anomalies in the dataset. The majority of anomalies are significantly longer than the window size of 140.

miss subtle deviations or fail to differentiate between normal seasonal changes and abnormal fluctuations due to a limited temporal perspective.

One pertinent study, *Time Series Forecasting with LLMs: Understanding and Enhancing Their Preferences* Yang et al. [2024], observes that LLMs excel in predicting time series with clear patterns and trends but face challenges when the data lacks periodicity. It further shows that the preferences of LLMs for certain types of time series are not yet fully understood, and that these models may not effectively utilize longer historical contexts, further limiting their reliability in anomaly detection via forecasting.

Lack of Interpretability Finally, forecasting-based methods offer limited interpretability. While they can produce accurate numerical predictions, they provide no intrinsic explanation for why a certain prediction deviated from the actual value, or why that deviation should be considered anomalous. This lack of transparency is a key drawback for use cases where human oversight or actionable insights are required.

3.2 Direct Anomaly Detection Approach

3.2.1 Setup

In the direct approach, the entire time series (or a significant portion of it) is converted into a textual format and provided as input to the LLM. Instead of forecasting future values, the model is explicitly prompted to identify anomalous points within the sequence. This

method leverages the inherent language understanding and reasoning capabilities of LLMs, and avoids the limitations associated with window-based forecasting approaches.

Prompter Strategy: Core Idea

The *Prompter* strategy directly feeds a tokenized version of the time series into the LLM and asks it to reason over the sequence to identify anomalous values. This relies entirely on the model's in-context learning and pattern recognition abilities.

Difference from SigLLM: Output Control vs Reasoning

In the SIGLLM framework Alnegheimish et al. [2023], the output of the LLM is strictly constrained by limits on the number of output tokens. This restriction forces the model to output only a list of anomaly indices or values, minimizing verbosity and simplifying parsing.

In contrast, our implementation avoids imposing any output token limit. We instruct the LLM to explicitly format its final answer as:

`Anomalies: [x1, x2, ...]` or `No anomalies` / `No anomaly`.

This simple format ensures that results remain parsable, while still giving the model space to reason. Our belief is that restricting output tokens might hinder the model's ability to perform proper reasoning—an idea supported by findings in reasoning literature such as Chain-of-Thought prompting Wei et al. [2022] and self-reflection Zhou et al. [2022], where free-form intermediate steps are shown to improve performance on complex tasks.

Benefits: Interpretability and Diagnostic Insight

Allowing the model to reason freely offers several practical benefits. First, it provides qualitative insight into the model's internal logic and justification, allowing us to analyze its reasoning process. Second, it helps identify failure cases and hallucinations where the model claims to use techniques like ARIMA or differencing, even though these are never provided in the prompt.

This behavior falls within the broader category of LLM hallucinations Ji et al. [2023], where language models produce confident but incorrect statements. Table 3.1 highlights a few examples of such behavior in our experiments.

These insights are not only useful for evaluating model quality, but also for improving future prompting strategies and safeguarding against unintended behaviors.

Table 3.1: Examples of LLM reasonings in unconstrained explanation outputs.

Time Series Type	Hallucinated Explanation Snippet
Level shift	*"Based on ARIMA decomposition, the sudden increase at index 85 does not match the expected seasonal component."*
Gradual drift	*"Applying exponential smoothing revealed that the values between t=50 and t=70 deviate from forecasted trends."*
Outlier	*"Using z-score detection, the spike at t=120 exceeds 3 standard deviations."*

3.2.2 Motivation and Methods

This approach aligns with our objective to explore prompt-based methods without any fine-tuning. Beyond merely detecting anomalies, we place significant emphasis on the interpretability of the LLM's outputs, aiming to generate coherent and human-understandable explanations that provide valuable insights into the detected anomalies.

To systematically investigate the effectiveness of this method, we explore various prompting strategies:

- **Zero-shot prompting:** Providing the LLM with minimal instructions, allowing it to perform anomaly detection based solely on its pre-trained knowledge.
- **One-shot prompting:** Supplying the LLM with one or more annotated examples of time series containing labeled anomalies, guiding the model to generalize and apply similar reasoning to unseen sequences.
- **Reasoning-enhanced prompting:** Integrating advanced reasoning techniques—such as Chain-of-Thought (CoT) Wei et al. [2022] and self-reflection Zhou et al. [2022]—within the prompt to encourage the LLM to articulate intermediate reasoning steps during anomaly evaluation. This approach not only aids in accurate anomaly detection, but also enhances the interpretability of the model's decision-making process.

Additionally, to evaluate how model architecture and scale impact detection and explanation capabilities, we conduct experiments using multiple LLMs, including popular open-source models such as **LLaMA**, **Mistral**, **IBM-Granote**, and **DeepSeek**.

Our evaluation considers both quantitative metrics—such as precision, recall, and F1-score—and qualitative analysis of the explanations generated by each model under different prompting strategies. This comprehensive comparison aims to provide a better understanding of the strengths and limitations of prompt-based LLM approaches for time series anomaly detection.

3.3 LLM-Based Pipelines for Anomaly Explanation

In addition to detecting anomalies, a key focus of this thesis is evaluating the ability of large language models (LLMs) to produce meaningful and human-interpretable explanations for why certain points or segments in a time series are considered anomalous. To that end, we explore three distinct pipelines for structuring the detection and explanation processes. These are illustrated in Figure 3.4.

3.3.1 Motivation for Pipeline Design

Each pipeline represents a different way of leveraging LLMs' capabilities, depending on the target use case. The goal is to compare their trade-offs in terms of performance, interpretability, and modularity.

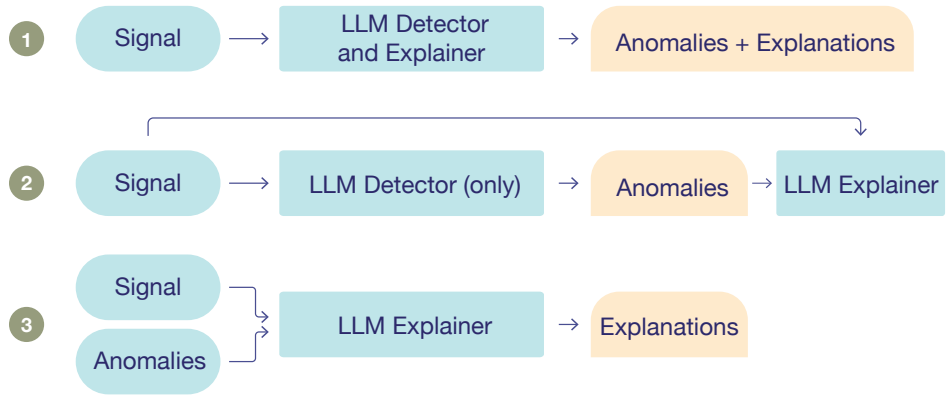

Figure 3.4: Overview of the three explanation pipelines evaluated in this thesis. **Pipeline 1:** A unified approach where the LLM simultaneously detects and explains anomalies. **Pipeline 2:** A dual-stage approach separating detection and explanation, allowing for more control and flexibility. **Pipeline 3:** A dedicated LLM is used to automatically classify the type of each explanation for evaluation purposes.

- **Pipeline 1** integrates detection and explanation in a single prompt. This is simple and efficient, but may conflate tasks and reduce flexibility.
- **Pipeline 2** separates detection and explanation into two stages, allowing independent control over both, as well as the ability to mix-and-match different models or prompt styles.
- **Pipeline 3** introduces a third LLM as an evaluator to automatically assess the quality of explanations, enabling systematic benchmarking.

This modular view of explanation pipelines is useful not only for experimentation, but also for identifying the most scalable and interpretable solutions in practical anomaly detection scenarios.

3.3.2 Explanation Evaluation via LLM

While human evaluation of explanations is ideal, it is not scalable. To address this, we propose an automatic evaluation framework in which an LLM acts as an explanation classifier. Inspired by the approach proposed in Zytek et al. [2024], we propose an automatic evaluation framework in which an LLM acts as an explanation classifier :

- The model-generated explanation.
- The anomaly label

The evaluator LLM is then tasked with verifying whether the explanation aligns with the ground-truth label of the anomaly. Specifically, it:

1. Analyzes the explanation for each anomaly,
2. Compares the predicted type to the true label associated with the anomaly,
3. Outputs a binary alignment score: 1 if the explanation matches the correct anomaly type, 0 otherwise.

This allows us to compute an overall **Explanation Alignment Score** as the proportion of explanations that are consistent with ground-truth anomaly types. This method provides a scalable, automated way to evaluate not only whether LLMs can justify their anomaly detection decisions, but whether those justifications are correct and interpretable from a domain perspective.

3.3.3 Comparison and Evaluation

We evaluate all three pipelines based on the following criteria:

- **Detection Performance**: Using standard precision, recall, and F1-score.
- **Explanation Alignment**: Whether the explanation matches the true anomaly type.

The goal of this comparative evaluation is to determine which approach offers the best balance between interpretability and anomaly detection performance.

Chapter 4

Challenges

Throughout our experiments, we encountered several key challenges that shaped the direction and design of our methodology. These challenges are grouped into the following categories:

4.1 Key Challenges

- **Input Length and Context Window**: LLMs typically operate within a limited context window. Very long time series often exceed this limit, resulting in degraded attention performance. Even when the full sequence fits, the model's attention may become diluted across the input, making it effectively "blind" to certain important regions, especially long stretches of largely normal data that nonetheless contain subtle anomalies.

- **Prompt Design Sensitivity**: The quality and clarity of the prompt play a critical role in the model's performance. Poorly-designed prompts often lead LLMs to ignore relevant patterns or fail to tap into their pre-trained knowledge. Designing prompts that balance clarity, guidance, and flexibility proved to be an iterative and non-trivial process.

- **Consistency and Predictability**: Achieving consistent and predictable outputs from LLMs poses significant challenges. Despite using the same prompts, slight variations in temperature settings, model versions, or token limits often led to different anomaly detection outcomes, complicating evaluation and reproducibility.

- **Contextual Understanding Limitations**: While LLMs excel at general sequence modeling, they lack inherent domain-specific knowledge about time series behavior.

Without explicit guidance, models may miss subtle anomalies or incorrectly classify normal variations as outliers.

- **Evaluation of Explanations**: Building a reliable framework to evaluate the interpretability and quality of LLM-generated explanations was another key challenge. Existing benchmark datasets focus primarily on detection accuracy without providing labeled explanations. To address this gap, we had to design and construct a synthetic dataset with explicitly labeled anomalies and ground-truth explanations, allowing for controlled evaluation.

- **Model Dependency and Variability**: The choice of LLM architecture significantly influenced results. Switching between models—such as LLaMA and Mistral—unexpectedly led to noticeable differences in both detection performance and explanation quality. Understanding and managing this variability became essential for drawing reliable conclusions.

- **Separation of Detection and Explanation Tasks**: An open methodological question we faced was whether to handle anomaly detection and explanation generation simultaneously (i.e., within a single prompt and model pass) or to separate them into distinct stages. Both approaches presented trade-offs in terms of performance, interpretability, and prompt complexity.

4.2 Proposed Solutions

To address the aforementioned challenges, we propose the following strategies:

4.2.1 Segmenting Input Length

We divide long time series into manageable chunks, ensuring each segment contains sufficient context for effective anomaly detection.

4.2.2 Enhanced Prompt Engineering

We developed carefully crafted prompts to provide explicit instructions and contextual cues. For example:

> Here is a time series of daily temperatures. Typical behavior includes steady increases/decreases, peaks, and troughs. Identify any points that deviate significantly from these patterns and explain why.

To optimize performance, we experimented with different styles of prompt formulations, including direct instructions, reasoning steps, and examples.

4.2.3 Incorporating Few-Shot Learning

To mitigate the lack of domain knowledge and improve contextual understanding, we integrated one-shot and few-shot prompting strategies. Providing the LLM with labeled examples of normal sequences helped guide its reasoning.

4.2.4 Exploring Chain-of-Thought Reasoning

We investigated the use of Chain-of-Thought (CoT) prompting techniques to encourage intermediate reasoning steps, enhancing the transparency and interpretability of the model's decisions.

4.2.5 Dedicated Evaluation Framework

To evaluate the LLM's explanation capabilities systematically, we developed a synthetic dataset with ground-truth labeled anomalies and corresponding explanations. This allowed us to conduct a controlled assessment of the model's interpretability and reasoning quality.

4.2.6 Model Comparison and Robustness Checks

Recognizing the variability in results across different LLMs, we systematically compared multiple architectures (e.g., LLaMA, Mistral, IBM, DeepSeek) and model sizes to understand their impact on detection accuracy and explanation quality.

Chapter 5

LLM for Time Series

In this section, we explore how time series data can be represented for input into large language models (LLMs). Time series data can take many forms, and we define a univariate time series as $X = (x_1, x_2, \ldots, x_T)$, where $x_t \in \mathbb{Z}_{\geq 0}$ is the value at time step t, and T is the length of the series. To make the time series LLM-ready, we apply several preprocessing steps: scaling, quantization, rolling windows, and tokenization.

5.1 Time Series Representation

5.1.1 Scaling

Time series data often includes values with varying magnitudes, including both positive and negative values. To standardize the representation and improve computational efficiency, we apply a scaling operation that shifts the range of the time series to non-negative values:

$$x_{st} = x_t - \min(x_1, x_2, \ldots, x_T), \quad X_s = (x_{s1}, x_{s2}, \ldots, x_{sT}), \quad x_{st} \in \mathbb{R}_{\geq 0}$$

This transformation eliminates the need to handle negative values and optimizes tokenization by reducing the number of digits needed. While other scaling methods, such as min-max scaling, could be used, we chose our approach in order to maintain the gaps between data points while simplifying tokenization.

5.1.2 Quantization

LLMs typically work with a finite vocabulary of tokens, but time series values are continuous, so they must be quantized before inputting them to an LLM. We use a rounding method to

map the continuous values into discrete integers. After scaling the time series, we round the values to a fixed number of decimals, which are then converted to integer format:

$$X_q = (x_{q1}, x_{q2}, \ldots, x_{qT}), \quad x_{qt} \in \mathbb{Z}_{\geq 0}$$

For example, a sequence of values like $0.2437, 0.3087, 0.002, 0.462$ becomes $244, 309, 2, 462$. This process ensures that the data is compatible with the tokenization process and is reversible up to a certain precision.

5.1.3 Rolling Windows

Due to the upper limits on input length for LLMs (e.g., MISTRAL has a limit of 32k tokens), time series data that exceeds these limits must be processed in segments. A rolling windows technique is used to segment the time series into manageable chunks, with a predefined window size w and step size:

$$\{X_i\}_{i=1}^{N}, \quad X_i = (x_{i1}, x_{i2}, \ldots, x_{iw})$$

This allows for the handling of large time series while respecting the token input size limitations of LLMs.

5.1.4 Tokenization

Tokenization involves converting the processed time series into a sequence of tokens. Different tokenizers handle numbers differently. For instance, the SentencePiece tokenizer (used by LLAMA-2 and MISTRAL) segments numbers into individual digits, while the GPT tokenizer segments numbers into larger chunks. Empirical evidence suggests that GPT's segmentation of numbers into chunks may hinder its ability to learn time series patterns effectively.

To address this issue, we apply the approach proposed in Gruver et al. Gruver et al. [2023], where spaces are inserted between digits to ensure that each digit is treated as a separate token. For example, the sequence $244, 309, 2, 462$ would be tokenized as 2 4 4, 3 0 9, 2, 4 6 2.

5.2 Summary of Preprocessing Steps

The key steps in preparing time series data for LLM input include:

- **Scaling**: Shifting the data to a non-negative range to optimize tokenization.

- **Quantization**: Rounding continuous values into discrete integers to match the LLM's tokenization format.
- **Rolling Windows**: Segmenting the time series into manageable windows to handle input size limitations.
- **Tokenization**: Ensuring proper tokenization of digits to enable the LLM to learn temporal patterns effectively.

These preprocessing steps allow time series data to be processed efficiently and accurately by LLMs, providing the foundation for anomaly detection tasks.

5.3 One-shot and Few-shot Prompting

One-shot and few-shot prompting are prompting techniques where the model is given a small number of examples before being asked to solve a new instance of the task. These methods enable LLMs to generalize from limited demonstrations, and leverage their powerful in-context learning abilities.

In the context of time series anomaly detection, few-shot learning could involve providing the model with several examples of normal and anomalous segments. The model then learns the distinction between these behaviors and applies that understanding to new data. This is particularly valuable in settings where annotated datasets are small, or where patterns are domain-specific.

> "Here are a few examples of typical and anomalous behaviors in time series data. Based on these examples, identify whether the following sequence contains any anomalies."

While few-shot prompting allows for more comprehensive demonstrations, we choose to focus on a **one-shot prompting setup** in this work. This design choice is motivated by both practical and technical considerations:

- **Practicality for end users**: In many real-world applications, users often have only one clear example of what a "normal" pattern looks like for a given type of signal. Designing prompts that rely on a single reference pattern is thus more aligned with practical deployment.
- **Prompt length efficiency**: Increasing the number of examples in the prompt expands the input context, which can lead to diminished model performance due to context

window saturation and attention dilution Liu et al. [2023]. Keeping prompts concise ensures that the model focuses on the relevant signal characteristics without being overwhelmed by excessive information.

"Here is a representative example of a normal signal pattern. Based on this, analyze the following sequence and identify any deviations that might indicate anomalies."

Thus, our methodology prioritizes one-shot prompts as a balanced compromise between effectiveness, clarity, and efficiency. The complete set of prompt templates used for anomaly detection and explanation tasks can be found in Appendix A.1.

Chapter 6

Synthetic Dataset for Time Series Anomaly Detection

6.1 Motivation and Design Considerations

Anomaly detection in time series is a critical task across a variety of domains, including finance, healthcare, cybersecurity, and industrial monitoring. However, real-world datasets often suffer from two major drawbacks: a lack of comprehensive labeled anomalies, and a limited diversity of abnormal behavior types. Moreover, publicly available datasets rarely include explicit explanations of why certain points are considered anomalous, complicating the evaluation of interpretability-focused models.

To overcome these limitations, we constructed a synthetic dataset in which the type, location, and characteristics of anomalies are fully controlled. This controlled environment enables the assignment of explicit labels for each anomaly, providing a reliable ground truth not only for detection accuracy but also for evaluating the quality of explanations generated by large language models (LLMs).

A key design goal was to make the dataset suitable for prompt-based LLM methods. Specifically, for each segment containing an anomaly, we include a corresponding example of normal behavior to serve as a reference. This setup supports both zero-shot and one-shot learning scenarios, allowing LLMs to compare and contrast normal versus abnormal patterns effectively.

Our dataset design draws inspiration from benchmark collections such as the UCR Time Series Classification Archive [Dau et al., 2018]. By generating a rich variety of anomaly types with detailed labels and explanations, the dataset serves as a comprehensive tool to assess both detection and interpretability capabilities of LLM-based approaches.

6.2 Anomaly Types and Implementation

6.2.1 Types of Anomalies in Time Series

In the context of time series data, an anomaly refers to an observation or segment that deviates significantly from the expected behavior of the data. While domain-specific variations exist, anomalies are broadly classified into two high-level categories:

- **Point Anomalies:** Isolated data points that are significantly different from their neighbors. These could manifest as sudden spikes or abrupt drops.
- **Collective Anomalies:** Sequences or segments of contiguous data points that appear abnormal when considered together, even if individual points might seem normal in isolation. Examples include sustained drifts, abnormal bursts, or unexpected changes in pattern or variance.

These two general categories encompass a wide range of more specific anomaly types, which are commonly observed in real-world applications. To ensure comprehensive evaluation of anomaly detection strategies, we consider multiple fine-grained anomaly types, introduced in the following section.

6.2.2 Anomaly Types and Synthetic Dataset Implementation

To mimic the heterogeneity and complexity of real-world signals, our synthetic dataset includes the following specific anomaly types. These categories align with those shown in Table A.1 and capture a range of temporal behaviors:

- **Sudden Spike:** An abrupt, short-lived upward excursion in the signal. This can occur, for instance, when sensor noise or a brief process malfunction causes the readings to spike unexpectedly.
- **Sudden Drop:** A rapid downward shift in the signal over a short interval. This is analogous to equipment failure or a sharp decline in a measured process variable.
- **Gradual Drift:** A slow, continuous deviation from the original trajectory, simulating phenomena like sensor aging or environmental drift over time.
- **Outlier:** A relatively brief, isolated deviation that does not span multiple contiguous time steps. This is typical of transient anomalies such as momentary glitches or spikes in noise.

- **Level Shift:** An abrupt, lasting change in the baseline or mean level of the signal, akin to a system fault or an operational intervention that reconfigures the process setpoint.
- **Variance Shift:** A sudden change in the overall variability (standard deviation) of the signal, representing events such as a noise burst, a sensor calibration error, or a broader shift in measurement precision.
- **Hybrid Anomaly:** A combination of multiple anomaly types (e.g., a sudden spike plus a variance shift, or a level shift that leads into a slow drift). This class introduces realistic complexity, as multiple disruptions can interact in nontrivial ways.

Each anomaly is injected via a dedicated perturbation function into a base time series template (typically sine waves or random walks with noise). For each injected anomaly, we record:

- The type and nature of the anomaly;
- Its start position and duration;
- A textual explanation of the perturbation applied.

To support evaluation in one-shot learning settings, we additionally extract representative anomaly-free segments from the original time series. All sequences, including both normal and anomalous segments, are derived from the same baseline signal to maintain consistency in statistical properties and contextual relevance.

6.2.3 Dataset Characteristics and Statistics

For thorough evaluation of anomaly detection and explanation strategies, we constructed a synthetic dataset composed of 424 time series samples. Each sample is 500 time steps in length and includes between one and three anomalies, ensuring diverse scenarios ranging from sharp, isolated outliers to sustained drifts or combined disruptions.

As a realistic starting point, we derive the *base* time series from the first 500 non-anomalous time steps of the NASA SMAP dataset. Injected anomalies are thus superimposed onto an authentic baseline pattern, providing controlled yet realistic test cases. This design guarantees complete label control and clear, well-defined ground truth for each event.

Across these 424 samples, we inject and label *seven* distinct types of anomalies: sudden spikes, sudden drops, gradual drifts, outliers, level shifts, variance shifts, and hybrid anomalies. Each anomaly spans a contiguous window of about 12 to 13 time steps, consistent

with our aim to simulate both abrupt and short-lived deviations as well as more extended disruptions.

Table 6.1 provides a summary of anomaly distributions within the dataset, including the average length of each type. Overall, the dataset contains 265 labeled anomalies, offering a rich landscape for benchmarking detection and explanation methods under various prompting strategies (including zero-shot and one-shot). We refer to this synthetic benchmark as the **NasaSynth-Labeled Dataset**, reflecting its NASA-derived baseline signal and comprehensive suite of artificial anomalies.

Table 6.1: Distribution of Anomaly Types in the Synthetic Dataset

Anomaly Type	**Number of Instances**	**Average Length**
sudden_spike	53	12.49
sudden_drop	53	12.17
gradual_drift	53	12.62
outlier	53	12.77
level_shift	53	13.04
variance_shift	53	12.11
hybrid_anomaly	53	13.25

Each time series in the dataset is of fixed length (500 time steps), and includes between one and three anomalies. In total, this yields 414 time series containing a combined total of 265 labeled anomalies, covering a wide range of structures and challenges relevant to both detection and explanation tasks.

This dataset is specifically designed to support evaluation under different prompting strategies, including 1-shot setups, where clean segments from the same signal (free of anomalies) are extracted and provided as reference examples. The diversity of injected anomaly types and consistent formatting across samples make this dataset well-suited for benchmarking both anomaly detection accuracy and explanation quality with LLMs.

For clarity and future reference, we refer to this synthetic benchmark as the **NasaSynth-Labeled Dataset**—a signal-based synthetic dataset tailored for evaluating prompt-based anomaly detection and explanation with LLMs.

Chapter 7

Evaluation

In this chapter, we evaluate the effectiveness of our prompt-based framework for time series anomaly detection and explanation. Our evaluation is structured to address the following research questions:

- **RQ1:** Are large language models effective anomaly detectors for univariate time series?
- **RQ2:** How well do LLMs perform in generating accurate and interpretable explanations for detected anomalies?
- **RQ3:** How do different prompting strategies (zero-shot, one-shot, CoT) and explanation strategies (unified vs. dual-stage) impact detection and explanation performance?
- **RQ4:** What are the success and failure cases, and what insights can be drawn from them?

7.1 Datasets

To comprehensively assess our approach, we conduct experiments on both real-world and synthetic datasets. These datasets provide a diverse range of time series patterns, anomaly types, and domains, allowing for robust evaluation.

7.1.1 Synthetic Dataset

As described in Chapter 4, we constructed a synthetic dataset specifically designed to facilitate controlled evaluation of both detection and explanation capabilities. The key features of the synthetic dataset include:

- Explicitly labeled anomaly types: point anomalies, collective anomalies, level shifts, variance shifts, seasonal breaks, trend inversions, and hybrid anomalies.
- Corresponding normal pattern segments for each anomaly type, supporting one-shot learning evaluation.
- Ground-truth explanations for each anomaly, enabling a systematic assessment of explanation accuracy.

Dataset Statistics

This dataset serves as a primary benchmark for evaluating the quality of LLM-generated explanations, as well as the impact of different prompting strategies on detection accuracy.

7.1.2 Real-World Datasets

To assess the generalizability of our framework, we evaluate it on several well-established real-world datasets, widely used in the time series anomaly detection literature. These datasets span diverse domains, including satellite telemetry, production web traffic, and social media activity. Specifically, we include:

- **NASA MSL and SMAP:** Satellite telemetry signal datasets released by NASA, containing multivariate measurements from spacecraft components. We focus on the univariate subsets of these datasets, in line with prior works.
- **Numenta Anomaly Benchmark (NAB):** A comprehensive benchmark composed of real-world and synthetic time series data. We use five sub-datasets: *Art*, *AWS*, *AdEx*, *Traf*, and *Tweets*, covering a wide range of signal types and domains.

In total, these datasets offer a wide range of time series lengths, anomaly densities, and data dynamics, making them ideal for benchmarking the robustness and generalizability of anomaly detection methods.

Normal Segments for Prompting. For each real-world time series, we extracted a reference *normal* segment to support one-shot prompting. This was done by selecting a contiguous window of 200 time steps that do not overlap with any labeled anomaly. While this provides a consistent and easily replicable approach, it does not always yield the most representative example of normal behavior—particularly for signals with longer temporal cycles or more complex baseline dynamics.

A potential improvement in future work would involve reasoning more carefully about the properties of the signal (e.g., seasonality, periodicity, regime shifts), and selecting longer or more contextually rich reference segments when needed. Doing so could provide the LLM with a better understanding of the underlying "normal" patterns, improving both detection precision and explanation coherence.

Dataset Statistics

Table 7.1 summarizes the core characteristics of the real-world datasets used in our evaluation, including the number of sub-datasets, total signals, number of labeled anomalies, and average signal lengths (with standard deviation).

Table 7.1: Dataset Summary: 492 signals and 2349 anomalies.

Dataset	# Sub-datasets	# Signals	# Anomalies	Avg. Length
NASA	2	80	103	8686 ± 5376
NAB	5	45	94	6088 ± 3150
Total	11	492	2349	–

7.2 Evaluation Metrics

To evaluate the performance of our framework, we employ both detection-based and explanation-based metrics. These metrics are designed to assess the accuracy, robustness, and interpretability of the system across different anomaly types and prompting strategies.

Additional qualitative examples of detected anomalies and their alignment with ground-truth intervals are provided in Appendix A.4.

- **Detection Metrics:**
  - **Precision, Recall, and F1-score** computed at the anomaly point level, i.e., measuring the overlap between predicted anomaly indices and ground-truth anomaly positions.
- **Explanation Evaluation Metrics:**
  - **Explanation Classification Accuracy:** The percentage of explanations generated by the LLM that are correctly mapped to the ground-truth anomaly type, using our LLM-based evaluator.

For the synthetic dataset specifically, where each anomaly instance is annotated with a known type, we report detection and explanation metrics **per anomaly type**. This allows for a more fine-grained evaluation of the framework's ability to handle diverse anomaly structures and reasoning requirements.

7.3 Experimental Setup

We conduct our experiments using a range of open-source large language models (LLMs), including members of the LLaMA and Mistral families, as well as more recent models such as **DeepSeek** and IBM's **Granite** series.

The models are evaluated under various prompting strategies—**zero-shot**, **one-shot**, **few-shot**, and **Chain-of-Thought (CoT)** reasoning—detailed in Chapter 3. For explanation generation, we test both the unified (detection + explanation in a single pass) and dual-stage (separate detection and explanation) strategies. All models are queried using consistent hyperparameters to ensure fair comparison across setups.

Inference Configuration

To maintain determinism and reduce variability across repeated runs, we configure all LLMs with the same inference parameters, summarized in Table 7.2. These settings ensure that the model outputs the most likely completion, without introducing randomness or sampling-based variability—especially important in tasks involving explanations or reasoning steps.

Table 7.2: LLM Inference Parameters (used across all experiments)

Parameter	Value	Description
Temperature	0	No randomness; deterministic output
Top-p	1.0	Full distribution considered (no truncation)
# Samples	1	One output per prompt
One-shot window	200	Number of time series values used as input

A temperature of 0 forces the model to follow the highest-probability path during decoding, avoiding stochastic sampling and reducing noise in the output. Combined with a sample size of 1, this ensures that each prompt yields a single, deterministic, and most probable response. We also limit the one-shot context to 200 values per time series, in order to keep inputs concise and reduce unnecessary complexity in reasoning.

While it would be valuable to explore the effects of different temperature values, top-p thresholds, sample counts, and longer context windows on both detection and explanation

performance, such an investigation would require significantly more computational resources and is left as potential future work.

Model Overview

Table 7.3 summarizes the LLMs used in our study, along with their sizes and key inference parameters.

Table 7.3: Model and Hyperparameter Settings

Model	Size (B)	Temperature
LLaMA-3.2	7B	0
Mistral-v0.2	7B	0
DeepSeek-R1	8B	0
IBM Granite 3.2	8B	0

7.4 Comparison with Existing Anomaly Detection Methods

For comparative purposes, we evaluate our framework against a diverse set of baseline models commonly used in the time series anomaly detection literature. These baselines cover a wide spectrum of methodologies, from traditional statistical techniques to recent deep learning and transformer-based approaches. By including models with fundamentally different detection mechanisms, we aim to establish a strong and comprehensive comparison benchmark.

- **Statistical Methods:** We include classic unsupervised methods such as *ARIMA*, *Matrix Profiling (MP)*, and a simple *Moving Average (MAvg)*. These approaches rely on temporal structure and residual analysis, and are known for their simplicity and interpretability.
- **Deep Learning Models:** We evaluate reconstruction-based models such as *LSTM Autoencoders (LSTM-AE)*, *Variational Autoencoders (VAE)*, and *TadGAN*, which learn to reconstruct normal patterns and identify anomalies as deviations. We also include forecasting-based models like *LSTM with Dynamic Thresholding (LSTM-DT)*, as well as hybrid models like *AER*, which combines both reconstruction and prediction strategies.

- **Transformer-based Models:** We include *AnomalyTransformer (AT)*, a recent model specifically designed for unsupervised anomaly detection using attention mechanisms to capture temporal dependencies and detect deviations.
- **External Detection Services:** For completeness, we also compare against a production-level anomaly detection service: *Microsoft Azure Anomaly Detector*, a cloud-based system used in industrial applications.

These baseline models were selected to provide broad coverage over the current landscape of anomaly detection methodologies. Their inclusion allows us to rigorously assess the performance and advantages of prompt-based LLM approaches in both detection accuracy and explanation interpretability.

7.5 Results and Experiments

In this chapter, we present a series of experiments and ablation studies designed to systematically evaluate the effectiveness of our prompt-based LLM framework. Each section focuses on isolating specific factors that influence anomaly detection and explanation performance.

7.5.1 Model Selection and Behavior

Effect of LLM Model Choice

To assess the influence of language model architecture on detection quality and consistency, we tested several open-source LLMs using the same prompts, sampling parameters, and input format. Specifically, we evaluated Mistral, DeepSeek, Llama, and IBM Granite.

Despite using the same deterministic settings (temperature=0, top_p=1, single sample, fixed input structure), the behavior of these models varied significantly. These differences were not just in detection quality but also in the structure and interpretability of their outputs. Because our approach relies on parsing structured results (e.g., a final list of anomaly indices), this consistency is critical.

Inconsistencies in Output Behavior.

Some models, such as IBM Granite, failed to follow the task altogether, and produced irrelevant completions unrelated to time series. DeepSeek attempted to reason, but produced contradictory or incoherent interpretations. Below are examples taken from our initial evaluations:

DeepSeek:

The model responded with:

```
"So, for example, if the second part has an anomaly starting at 41
and ending at 0, but that's not possible, so perhaps the 41 and 0
are the start and end of the anomaly. But in reality, the data is
a time series, so the anomaly is a consecutive sequence. Wait, but
in the second part, the last few numbers are 258, 259, 260, ..., 168"
```

IBM Granite:

The model responded with:

```
"A: Python 2, 102 99 97 bytes
lambda x:x[1:]+[x[0]+1]
...
The 2019{2020 school year is coming to an end and we are
looking forward..."
```

indicating a complete divergence from the expected anomaly detection task.

Mistral:

The model responded with:

```
"Answer: [135, 136]
Explanation: The sequence contains two consecutive values that are
significantly larger than the rest of the sequence. This could
indicate an anomaly, such as a measurement error or an outlier."
```

This response was well-formatted, task-relevant, and interpretable—making it ideal for both automatic parsing and qualitative analysis.

Model Selection. Based on these experiments, we chose to use **Mistral** for all subsequent evaluations. It consistently followed instructions, provided interpretable reasoning, and formatted outputs in a way that aligned with our prompting protocol.

Limitations. We note that all the models evaluated were relatively small in scale and executed locally. The erratic behavior from some LLMs may stem from limitations in training data, instruction-following capability, or context size. Despite this, Mistral proved sufficiently reliable and performant for our use case.

7.5.2 LLMs for Anomaly Detection Performance

Comparison to Baseline Methods and Impact of Our Approach

Table 7.4 presents a comparative overview of F1 scores for *several state-of-the-art and classical anomaly detection methods* evaluated on the NASA (MSL, SMAP) and NAB (Art, AWS, AdEx, Traf, Tweets) datasets. Established techniques such as AER, LSTM-based detectors (LSTM DT, LSTM AE), ARIMA, and VAE generally set competitive benchmarks, with AER often achieving high scores (e.g., 0.587 and 0.819 on NASA datasets) and LSTM/AE or VAE variants also demonstrating robust F1 across several NAB tasks.

By contrast, **prompt-based LLM approaches**—including the SigLLM Prompter and our own Our Prompter (both zero-shot and one-shot variants)—exhibit varying performance. Nonetheless, several positive trends emerge:

- **Performance Gap vs. Leading Methods.** Traditional time-series methods (AER, LSTM DT) and certain deep-learning–based models (LSTM AE, VAE) generally outperform pure LLM prompts across many datasets. For instance, AER and LSTM DT frequently exceed F1 scores of 0.6 or 0.7 on NAB subsets (AdEx, Tweets). This gap highlights the maturity of specialized anomaly detectors, which are optimized for time-series structure and trained on these datasets.

- **Zero-Shot vs. One-Shot Behaviors.** For our approach, adding a single reference example (Our Prompter 1-shot) often boosts performance compared to the zero-shot setting. For instance, on NASA MSL, performance improves from 0.185 (zero-shot) to 0.363 (one-shot), illustrating how even one brief snippet of "normal" data can help the model calibrate its notion of typical behavior. However, some subsets (e.g., AdEx) remain challenging, with F1 staying at 0.0 despite the one-shot prompt.

- **LLMs as Detectors vs. Explainers.** Looking at the SigLLM Detector, we observe F1 scores of 0.429 on MSL and 0.431 on SMAP—results that exceed or rival many classical methods on some NAB categories. In comparison, our one-shot prompts close some of the gap on NASA MSL (0.363) but still lag behind on other subsets such as AdEx (0.0 vs. 0.727 for SigLLM Detector). This underscores that LLMs can indeed be leveraged effectively via forecasting *and* prompting, though specialized detectors maintain an advantage in certain domains.

- **Dataset-Specific Variability.** Although the one-shot prompt strengthens performance on some datasets, other subsets (e.g., AWS, Traf, AdEx) still show low F1 scores. This indicates that certain anomaly types—especially subtle or domain-specific patterns—are challenging for prompt-based LLM methods.

- **Considerations for Real-World Use.** While established time-series models currently retain higher average F1 scores, the simplicity and flexibility of a prompt-based solution remain compelling where labeled data is scarce or where *interpretable* justifications are highly valued. Crafting targeted prompts with minimal examples offers a streamlined path to a functioning detector and can facilitate verbal explanations for detected anomalies. However, for applications that demand consistently high accuracy across all anomaly categories, classical or specialized ML approaches remain preferable until LLM-based detectors demonstrate more stable reliability across diverse datasets.

Table 7.4: Benchmark Summary Results depicting F1 Score.

Pipeline	**NASA**		**NAB**					
	MSL	**SMAP**	**Art**	**AWS**	**AdEx**	**Traf**	**Tweets**	$\mu \pm \sigma$
AER	0.587	0.819	0.714	0.741	0.690	0.703	0.638	0.753 ± 0.109
LSTM DT	0.471	0.726	0.400	0.468	0.786	0.585	0.603	0.649 ± 0.161
ARIMA	0.525	0.411	0.308	0.382	0.727	0.467	0.514	0.582 ± 0.176
MP	0.474	0.423	0.571	0.440	0.692	0.305	0.343	0.570 ± 0.193
TadGAN	0.560	0.605	0.500	0.623	0.818	0.452	0.554	0.569 ± 0.142
LSTM AE	0.545	0.662	0.667	0.741	0.500	0.500	0.475	0.569 ± 0.158
VAE	0.494	0.613	0.667	0.689	0.583	0.483	0.533	0.557 ± 0.143
AT	0.400	0.266	0.414	0.430	0.500	0.371	0.287	0.467 ± 0.138
MAvg	0.171	0.092	0.222	0.408	0.880	0.157	0.776	0.458 ± 0.266
MS Azure	0.051	0.019	0.056	0.112	0.163	0.117	0.176	0.243 ± 0.225
SigLLM Prompter Mistral	0.160	0.154	0.370	0.268	0.000	0.135	0.257	0.223 ± 0.104
SigLLM Prompter GPT	0.049	0.110	0.154	0.194	0.133	0.133	0.197	0.133 ± 0.076
SigLLM Detector	0.429	0.431	0.400	0.362	0.727	0.480	0.762	0.525 ± 0.167
Our Prompter 0-shot	0.185	0.239	0.285	0.095	0.000	0.186	0.270	0.210 ± 0.064
Our Prompter 1-shot	0.363	0.260	0.200	0.264	0.000	0.117	0.328	0.326 ± 0.188

In summary, these findings emphasize that **prompt-based anomaly detection approaches can narrow the gap with specialized baselines in certain settings**, particularly when using a one-shot strategy. While classical methods (AER, LSTM DT, etc.) still dominate on average.

Effect of Shot Settings

Prompting strategy plays a fundamental role in how LLMs interpret and process time series data. In this section, we compare the performance of two primary prompting configurations: *zero-shot* and *one-shot*, highlighting how even minimal examples of "normal" behavior can shape both anomaly detection accuracy and, later on, the quality of generated explanations.

Zero-Shot vs. One-Shot Prompting. In the **zero-shot** scenario, the LLM receives only the target time series segment and must identify anomalies based purely on its internal

knowledge. This can lead to elevated false-positive rates if the model's notion of "normal" diverges from domain realities—particularly in contexts with subtle or gradual changes (e.g., industrial sensors that drift over time).

By contrast, the **one-shot** setup provides a short segment of clean, representative data before the main time series. Even this single exemplar can serve as an *anchor* for the model's notion of typical behavior. As seen in Table 7.4, our one-shot prompt significantly boosts the F1 score on NASA MSL (from 0.185 to 0.363), although it offers limited gains for other subsets like AdEx. This underscores that a single example can clarify baseline expectations for some anomaly types while failing to capture more complex or domain-specific patterns elsewhere.

Selecting the Most Effective One-Shot Example. One crucial factor is *which* normal sample is used as the reference. A short snippet may not reflect the full variability of "normal" behavior, especially if a signal's seasonal or periodic patterns span a longer window. For instance, if a sensor reading completes a full cycle only every 100 timesteps, but the provided one-shot snippet covers just 10 timesteps, the model may fail to recognize the broader "normal" fluctuation. In our work, we used a single, consistently sized snippet for simplicity. However, future studies should explore *optimal strategies* for selecting or constructing a longer and more diverse reference segment. This might dramatically improve detection outcomes in signals with extended periods or complex normal dynamics and is an area that was not fully addressed in our current approach.

Practical Advantages of One-Shot. From an application standpoint, many operational environments have at least one baseline period of stable data. Incorporating such a segment in the prompt *mimics* real-world usage, providing a rapid route to more reliable detection without extensive model retraining. If domain experts can supply an especially instructive example of "normal" behavior, it can meaningfully reduce false alarms while retaining the flexibility and interpretability benefits of large language models.

7.5.3 Explainability and Evaluation

Beyond detecting anomalies, a central objective of this thesis is to evaluate how effectively Large Language Models (LLMs) can *explain* anomalous events in a human-interpretable way. We compare three pipelines (Section 3.3) and multiple prompting strategies, assessing both **detection** (where relevant) and **explanation** performance. All results presented in this section are based on the **NasaSynth-Labeled Dataset** introduced in Section 6.2.3,

and the evaluation method presented in Section 3.3.2. Tables 7.5 and A.1 summarize our quantitative findings.

Overall Findings

Pipeline 3 as a Baseline for Explanation Only. Unlike Pipelines 1 and 2, which must both *detect* and *explain* anomalies, **Pipeline 3** assumes that anomaly locations are already known (i.e., the model is given the ground-truth anomalous segments). The LLM in Pipeline 3 focuses exclusively on generating a coherent, human-readable *justification* of why the provided points are anomalous. By eliminating the detection step, Pipeline 3 serves as a valuable *baseline* for how well an LLM can explain anomalies when the model is relieved of the need to identify them first.

To gauge explanation quality, we measure how accurately Pipeline 3 aligns with the *correct anomaly type*—an *Explanation Accuracy* or "Alignment Score." As shown in Table A.1, even when the model is told which values are anomalous (bypassing detection), the LLM achieves only modest alignment with the exact anomaly category in many cases. This indicates that while LLMs are capable of generating plausible rationales, significant refinement in prompt design or domain-specific cues is needed to consistently produce high-fidelity explanations.

Comparing Prompting Strategies for Anomaly Explanation

We explored zero-shot, one-shot, and Chain-of-Thought (CoT) prompts to see if they improved explanation quality. Despite these variations, our metrics do not clearly favor any single approach: *all* configurations yield relatively low explanation accuracy. While providing an example or encouraging step-by-step reasoning might refine the *style* of the explanation, this did not substantially increase alignment with true anomaly types in our tests.

Comparison of Pipeline Architectures

Pipeline 1 (Single-Step Detection and Explanation). In this configuration, a *single* LLM identifies anomalies and explains them in the same prompt. As seen in Table 7.5, detection quality (F1) remains low overall, but reaches its highest value (0.192) when given one-shot prompts without CoT. Explanation Accuracy, while also limited, appears slightly higher in certain zero-shot scenarios (0.170) but does not exceed 0.115 in one-shot conditions.

Pipeline 2 (Dual LLM: Detection then Explanation). Here, one LLM flags anomalies, and a second LLM explains the flagged points. The data in Table 7.5 show a small

boost in detection F1 (up to 0.211) under a one-shot, CoT prompt, compared to Pipeline 1's maximum of 0.192. However, Explanation Accuracy remains incomplete for certain runs, and does not appear definitively better when it *is* reported (0.110). This two-step approach can offer more modularity (e.g., distinct prompt styles for detection vs. explanation), but it risks *cascading errors*: If the first stage misses or misclassifies anomalies, any subsequent explanation will be misaligned.

Pipeline 3 (Explanation Only, Ground-Truth Anomalies). In Pipeline 3, the model is given the *true* anomaly locations and focuses solely on generating a coherent rationale for why those points are anomalous. Without the burden of detection, one might expect a higher Explanation Accuracy. Indeed, Table 7.5 indicates a 0.302 alignment score (i.e., the model matches the labeled anomaly type about 30% of the time). While this outperforms the explanation scores observed in Pipelines 1 and 2, it also underscores how challenging it remains for an LLM to provide correct, domain-specific justifications, *even when the anomalies are already known.*

Table 7.5: Detection and Explanation Performance under Different Prompting Strategies.

	Prompting Setting		**Detection**			**Explanation**
Pipeline	1-shot	CoT	F1 Score	Precision	Recall	Accuracy
1	no	no	0.090	0.118	0.072	0.170
	no	yes	0.051	0.079	0.037	0.095
	yes	no	0.192	0.369	0.130	0.115
	yes	yes	0.189	0.349	0.130	0.108
2	no	no	0.090	0.118	0.072	0.090
	no	yes	0.101	0.118	0.107	0.105
	yes	no	0.192	0.350	0.132	0.110
	yes	yes	0.211	0.451	0.138	0.115
3	X	X	X	X	X	0.302

Overall, Pipeline 1 offers a straightforward, single-step solution that may simplify deployment, while Pipeline 2 introduces modularity at the risk of error propagation, and Pipeline 3 serves as a useful upper bound for purely explanatory tasks. Depending on whether simplicity, flexibility, or pure explanation is the priority, each pipeline has its own appeal.

Performance by Anomaly Type

Table A.1 details detection (F1, precision, recall) and explanation accuracy per anomaly category:

- **Sharp, Localized Anomalies (sudden_spike, sudden_drop, outlier).** Pipelines 1 and 2 handle these anomalies relatively well, indicating that abrupt deviations are easier for LLMs to flag and describe.
- **Gradual Anomalies (gradual_drift, variance_shift, trend_inversion).** More subtle changes over longer timescales prove challenging for both detection and explanation, as the boundaries between normal and abnormal behavior are less clearly defined.
- **Composite Anomalies (hybrid_anomaly).** Performance tends to fall between sharp and gradual anomalies. While the model may note multiple factors contributing to the anomaly, producing a precise rationale that integrates them all remains difficult.

Discussion

Our experiments indicate that **prompt design** and **pipeline architecture** exert only modest influences on LLM-based anomaly detection and explanation—at least under the specific evaluation criteria we employ. While prompt variants (e.g., one-shot; Chain-of-Thought) can change the *style* of responses, they do not robustly elevate detection metrics or explanation alignment scores. This suggests that our methodology for guiding and measuring explanations may itself be a limiting factor, rather than the inherent capacity of LLMs.

Challenges in Explanation Evaluation. Determining the "quality" of a textual explanation is inherently difficult. Our "Explanation Accuracy" metric checks whether a rationale matches a prescribed anomaly label, but it may overlook deeper insights the LLM provides. Perhaps a broader, more fine-grained assessment—combining human feedback or domain-specific checks—would yield a more faithful portrait of each explanation's true value.

Representative outputs generated by the language models for various prompting strategies are provided in Appendix A.2.

Chapter 8

Conclusion and Future Directions

8.1 Conclusion

In this thesis, we explored the use of Large Language Models (LLMs) for time series anomaly detection using purely prompt-based methods, without any fine-tuning or architecture modifications. We demonstrated that LLMs (Mistral in our case) can serve as effective anomaly detectors when time series data is formatted as structured prompts. Moreover, we showed that LLMs could not only be capable of identifying anomalous points, but also of generating coherent, human-interpretable explanations for their decisions.

Our evaluation was conducted across both real-world and synthetic datasets. The synthetic dataset, carefully designed with labeled anomaly types, allowed us to assess explanation quality. Our experiments revealed that prompting strategies (such as one-shot learning) and model choice significantly affect detection performance and output consistency.

While LLMs are not yet competitive with state-of-the-art deep learning models in terms of raw detection accuracy across all scenarios, their strengths lie in interpretability and flexibility, attributes that are often underappreciated in conventional approaches. Our findings suggest that prompt-based anomaly detection with LLMs is a promising avenue for applications where explanation and transparency are just as important as accuracy.

8.2 Future Directions

While this work provides a first step toward understanding how LLMs can be used for time series anomaly detection and explanation, several directions remain open for further investigation:

- **Model Exploration:** Our experiments focused on a small set of open-source models.

Future work could evaluate a broader range of LLMs, including newer, larger, or more instruction-tuned models (e.g., GPT-4, Claude, Gemini) to assess how pretraining data and model architecture influence time series reasoning.

- **Robustness and Parameter Sweeps:** We employed fixed inference settings for consistency. Systematic experimentation with parameters such as temperature, top-p, and number of output samples could uncover the trade-offs between performance, randomness, and explanation diversity.

- **Few-Shot Learning:** While we prioritized one-shot prompting for practicality, exploring few-shot strategies may further improve both precision and explanation clarity by providing richer context and pattern diversity. Additionally, selecting the best single example for one-shot can be nontrivial; for many signals, it may be crucial to present a longer or more representative "normal" segment so the model properly learns typical fluctuations.

- **Multivariate Time Series:** This thesis focused solely on univariate time series. Extending the approach to multivariate settings would enable us to test whether and/or how well LLMs can reason over multiple signals jointly, an essential skill for real-world monitoring systems.

- **Better Anomaly Labeling:** Ground-truth explanations in real-world datasets are scarce. Improved annotation of anomaly types and causes would allow for more rigorous evaluation of explanation accuracy and potentially support the training of explanation-aware models.

- **Fine-Tuning for Format Consistency:** One persistent challenge was inconsistent output formatting. Fine-tuning LLMs on structured anomaly detection outputs could enforce better adherence to output templates, reducing parsing errors and improving reliability.

- **Long-Term Evaluation of Better Models:** As LLM technology evolves rapidly, repeating these experiments with future models could yield stronger results. A recurring benchmarking pipeline—especially one that includes explanation evaluation—would ensure that future progress can be systematically tracked.

- **Mixed Methods or Multi-Modal Inputs:** Pairing LLMs with domain-specific detectors (e.g., ARIMA or robust deep learning models) might reduce false positives and yield more refined context for explanation.

- **Advanced Evaluation Metrics:** Refining LLM-based assessors, or combining them with partial human oversight, could provide richer, more dependable metrics for explanation alignment.

In summary, while prompt-based LLMs hold promise for easily deployable, interpretable anomaly detection, they demand careful prompt engineering, are prone to unreliable outputs in certain contexts, and exhibit mixed performance on more subtle anomalies. By addressing these directions, future work can deepen our understanding of how LLMs interact with temporal data and help bridge the gap between explainability and performance in anomaly detection systems.

Appendix A

Prompt Templates and Representative Outputs

A.1 Prompt Templates for Anomaly Detection

In this work, we explore the ability of large language models (LLMs) to detect and explain anomalies in time series data using natural language prompts. All prompts follow a consistent format instructing the model to return a clear answer and a brief explanation. Prompts were tested in zero-shot and one-shot settings, with optional reasoning strategies such as Chain of Thought (CoT).

A.1.1 Zero-Shot Prompt for Anomaly Detection

```
Below is a time series sequence (scaled to integers):
[12, 13, 14, ..., 90, 91, 15, 13]

Identify any anomalous subsequence and explain briefly why it is anomalous.

Format your final answer EXACTLY like this:
Answer: [start_value_of_anomaly, end_value_of_anomaly]
Explanation: [short explanation]

If no anomalies, return:
Answer: no anomalies
Explanation: no anomalies
```

```
Please think step-by-step before giving your final answer.
```

This prompt assumes no prior exposure to normal patterns and is designed to elicit anomaly detection purely from statistical irregularities.

A.1.2 One-Shot Prompt Using Normal Reference Pattern

```
Below are the first 200 elements of the time series, showing the normal behavior:
[10, 11, 10, 12, 11, 10, 12, 11, 10, 12, 11, ...]

Below is another part from that time series sequence,
analyze the following time series
(same distribution) for an anomalous sequence (if any significant),
and explain what kind of anomaly it is
(and why you think it could be one):

[10, 11, 12, 13, 14, 90, 91, 92]

Provide your final answer in this format:
Answer: [start_value_of_anomaly, end_value_of_anomaly]
Explanation: [brief explanation of the anomaly]

If no anomalies, return:
Answer: no anomalies
Explanation: no anomalies

Please think step-by-step before giving your final answer.
```

In this one-shot setting, the model is provided with a long reference pattern representing the normal behavior, followed by a test sequence to analyze.

A.1.3 Prompt for Explanation of Known Anomalies

```
Below is a time series sequence (scaled to integers):
[12, 13, 14, ..., 90, 91, 15, 13]

These values in the sequence are considered anomalous: [13, 14]
```

```
Explain what kind of anomaly it is and why.

Format your final answer EXACTLY like this:
Explanation: [short explanation]

Please think step-by-step before giving your final answer.
```

This prompt is used for explanation-only tasks, assuming the anomaly is already detected. The model is expected to justify the anomaly in human-understandable terms.

A.2 Representative Model Outputs

Example 1 – Correct Detection with CoT Reasoning

```
Step 1: Most values in the sequence are around 12{15.
Step 2: Values 90 and 91 show a sudden spike not seen elsewhere.
Step 3: This deviation suggests an anomaly.

Answer: [90, 91]
Explanation: Sudden spike inconsistent with prior values.
```

Example 2 – No Anomalies Detected

```
Step 1: All values fluctuate slightly between 10 and 13.
Step 2: There are no significant jumps or drops in value.

Answer: no anomalies
Explanation: Sequence is consistent with normal behavior.
```

These outputs demonstrate that LLMs can follow structured reasoning and provide clear justifications for their anomaly decisions when prompted appropriately.

A.3 Detailed Results by Anomaly Type

Table A.1 shows the full breakdown of detection and explanation scores by anomaly type and pipeline. It supports the discussion in Section 7.5.3.

Table A.1: Detection and Explanation Scores by Anomaly Type and Pipeline. ("–" indicates that the pipeline does not produce a detection score for Pipeline 3.)

Anomaly type	**Pipelines**	**Detection**			**Explanation**
		F1 Score	Precision	Recall	Accuracy
sudden_spike	1	**0.174**	**0.316**	**0.120**	0.320
	2	**0.174**	**0.316**	0.120	0.260
	3	–	–	–	**0.480**
sudden_drop	1	0.095	0.231	0.060	0.200
	2	**0.127**	**0.308**	**0.080**	0.040
	3	–	–	–	0.200
gradual_drift	1	**0.061**	**0.125**	**0.040**	**0.020**
	2	0.060	0.040	**0.040**	0.000
	3	–	–	–	0.000
outlier	1	**0.156**	**0.357**	**0.100**	0.200
	2	0.154	0.333	**0.100**	0.300
	3	–	–	–	**0.840**
level_shift	1	0.462	0.643	0.360	0.000
	2	**0.468**	**0.667**	**0.360**	0.120
	3	–	–	–	**0.140**
variance_shift	1	0.000	0.000	0.000	0.000
	2	0.000	0.000	0.000	0.000
	3	–	–	–	**0.180**
hybrid_anomaly	1	**0.436**	**0.607**	**0.340**	0.120
	2	0.425	0.567	**0.340**	0.160
	3	–	–	–	**0.420**
All Anomalies	1	0.189	0.349	0.130	0.108
	2	**0.211**	**0.451**	**0.138**	0.160
	3	–	–	–	**0.302**

A.4 Additional Visualizations of Anomaly Detection Results

This appendix provides additional visual examples of time series analyzed using the proposed LLM-based anomaly detection pipeline. In each figure:

- The **blue curve** represents the input time series.
- The **red segments** correspond to the ground-truth anomaly windows.

- The **green shaded regions** indicate the anomalies detected by the model.

Each caption includes the dataset, the instance ID, the prompting configuration (e.g., 0-shot or 1-shot), and the detection performance metrics (F1, precision, recall).

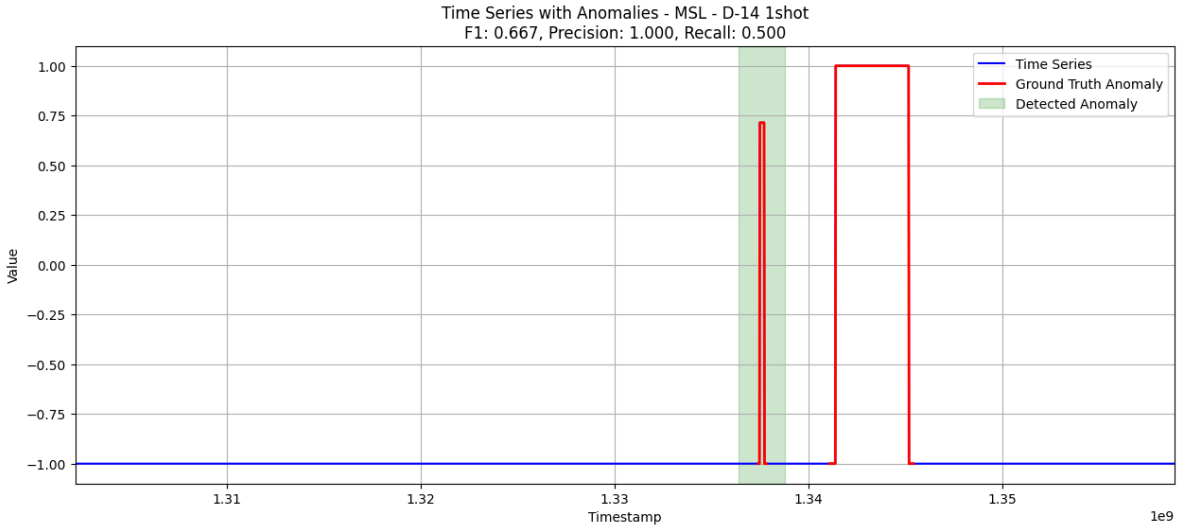

Figure A.1: MSL Dataset – D-14 (1-shot). The model detects one of the two anomaly intervals.

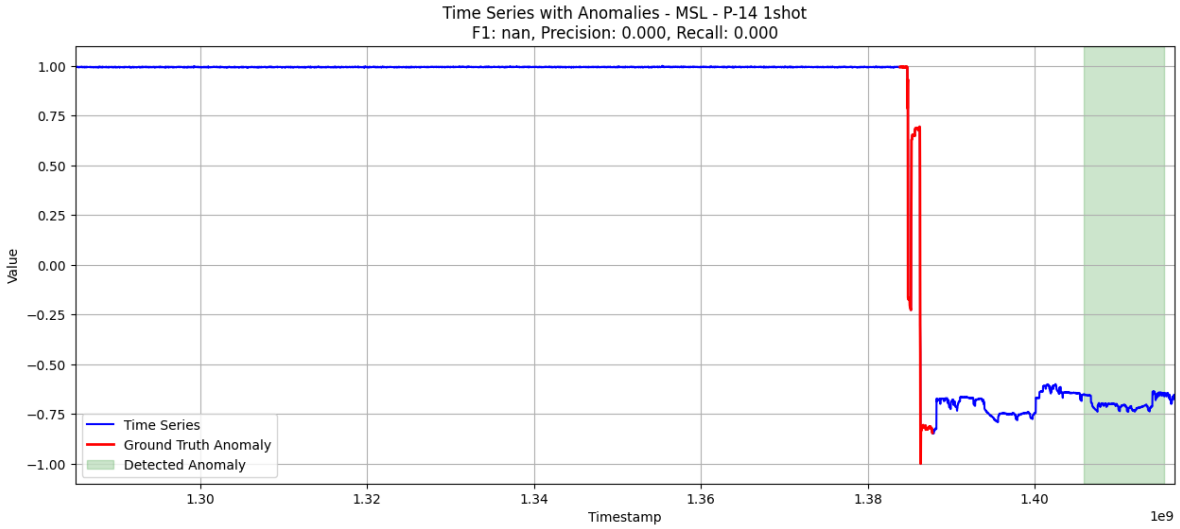

Figure A.2: MSL Dataset – P-14 (1-shot). Model fails to detect the anomaly.

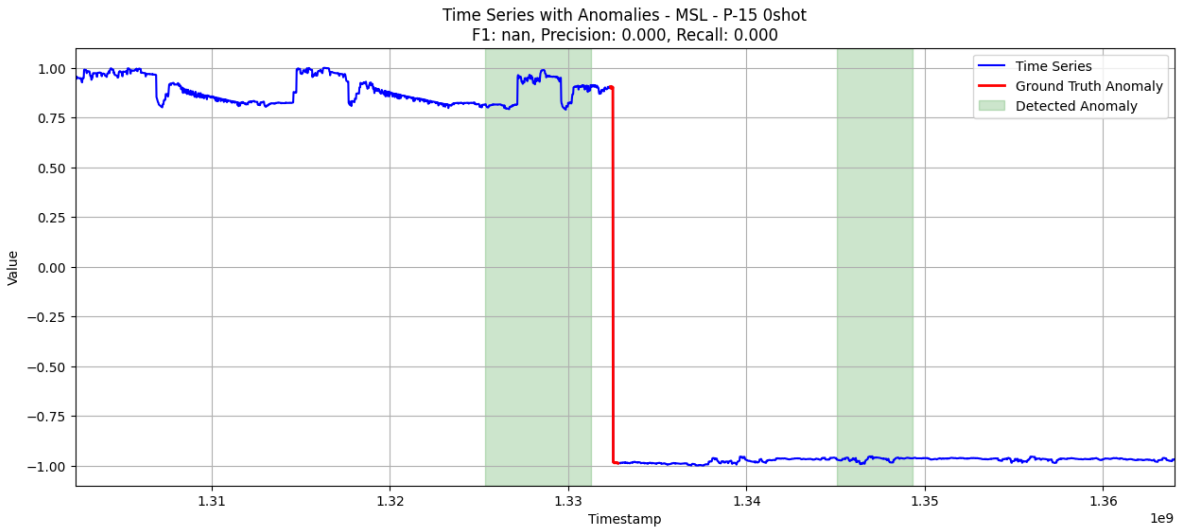

Figure A.3: MSL Dataset – P-15 (0-shot). Model detects spurious regions far from the true anomaly.

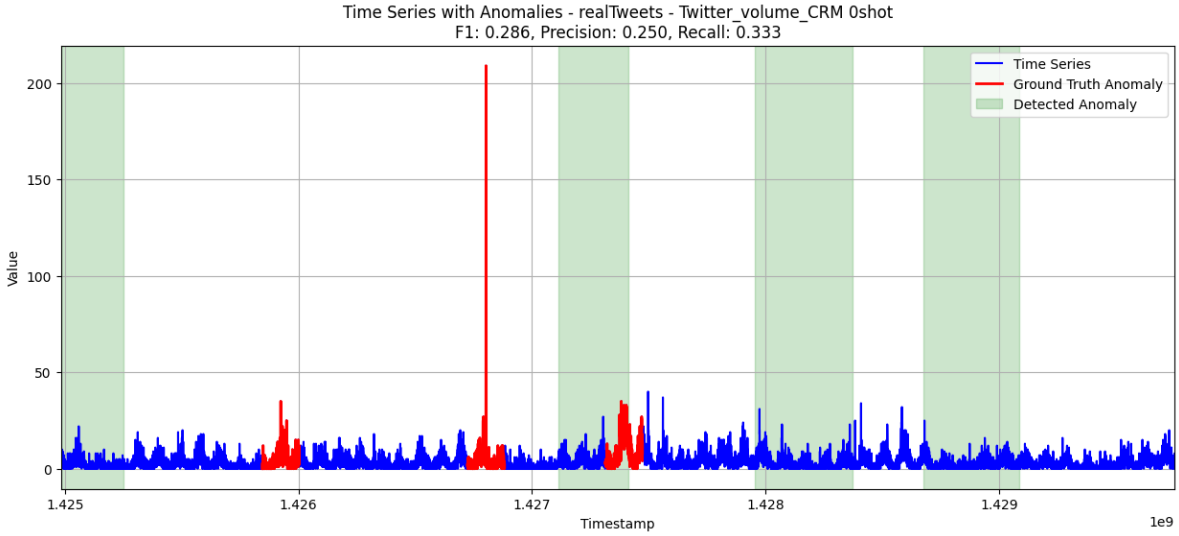

Figure A.4: realTweets – Twitter_volume_CRM (0-shot). Sparse but partially correct detections of multiple anomalies.

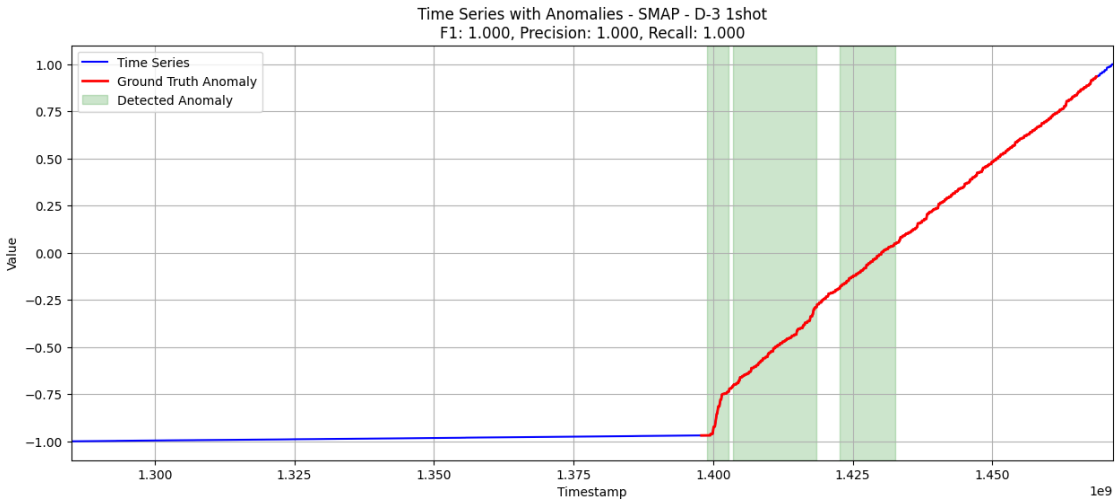

Figure A.5: Example from SMAP – D-3 (1-shot). Successful detection of sustained anomaly.

Bibliography

Sarah Alnegheimish, Matthew Roughan, Sarah M Erfani, James Bailey, and Christopher Leckie. The case for a learned anomaly detector for industrial time series. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022.

Sarah Alnegheimish, Linh Nguyen, Laure Berti-Équille, and Kalyan Veeramachaneni. Large language models can be zero-shot anomaly detectors for time series. In *Proceedings of the First International Conference on Data-Centric Machine Learning Research*. PMLR, 2023.

Aman Ansari, Xuchuan Zhang, Jinsong Qian, and et al. Chronos: Language modeling for time series with pretrained transformers. *arXiv preprint arXiv:2402.09322*, 2024.

George E. P. Box and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, 1970.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33: 1877–1901, 2020.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2023.

Anh Dau, Eamonn Keogh, et al. The ucr time series classification archive. *arXiv preprint arXiv:1810.07758*, 2018.

Steven Dooley, Isabel Valera, and et al. Forecastpfn: Probabilistic forecasting with parallel function networks. *arXiv preprint arXiv:2305.17535*, 2023.

Aaron Geiger and Ran El-Yaniv. Tadgan: Time series anomaly detection using generative adversarial networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1119–1129, 2020.

Nicholas Gruver, Di He, David Salinas, Valentin Flunkert, Syama Rangapuram, Jan Gasthaus, and Tim Januschowski. Are large language models zero-shot time series forecasters? *arXiv preprint arXiv:2310.06810*, 2023.

Kyle Hundman, Vangelis Constantinou, Christian Laporte, Ian Colwell, and Travis Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 387–395, 2018.

Zhewei Ji, Nayeon Lee, Jason Fries, Yejin Yu, Kyunghyun Cho, et al. Survey of hallucination in natural language generation. *ACM Computing Surveys (CSUR)*, 2023.

Patrick Koh, Robert Berwick, Daniel Ziegler, Yoav Shoham, and Oren Levy. Self-instruct: Aligning gpt-3 with human preferences through reinforcement learning from human feedback. *arXiv preprint arXiv:2302.06585*, 2023.

Nelson F Liu, Matt Gardner, Yonatan Belinkov, and et al. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.

Douglas C. Montgomery. *Introduction to Statistical Quality Control*. John Wiley & Sons, 8th edition, 2019.

Dawei Park, Yoshihiro Hoshi, and Charles Kemp. Multimodal anomaly detection for time series using generative adversarial networks. *arXiv preprint arXiv:1809.04758*, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.

Kashif Rasul, Justin Bayer, and et al. Lag-llama: Language modeling with time series at scale. *arXiv preprint arXiv:2309.04254*, 2023.

Chitwan Saharia, William Chan, Anton Glaese, Daniel Ziegler, Ping Xu, Quoc V Le, et al. Imagen: Text-to-image diffusion models. *arXiv preprint arXiv:2205.11487*, 2022.

Victor Sanh, Albert Webson, Colin Raffel, Angela Lo, Jonathan Drori, Clara Vania, John Bohannon, Swaroop Mishra, Beatriz Liu, Nisanth Goyal, et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2022.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Feng Xue, Biao Zhang, Yao Zhao, and et al. Promptcast: Forecasting with prompting large language models. *arXiv preprint arXiv:2302.01197*, 2023.

Kevin Yang, Maggie Xu, Nikhil Kumar, Tatsunori Hashimoto, Peter Bailis, and Christopher Re. Time series forecasting with llms: Understanding and enhancing their preferences. *arXiv preprint arXiv:2401.04796*, 2024.

Denny Zhou, Nathan Zhang, Yuhuai Dai, Denny Zhou, and et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

Alexandra Zytek, Sara Pido, Sarah Alnegheimish, Laure Berti-Équille, and Kalyan Veeramachaneni. Explingo: Explaining ai predictions using large language models. In *Proceedings of the IEEE International Conference on Big Data (IEEE BigData)*. IEEE, 2024.