Domain-Oriented Time Series Inference Agents for Reasoning and Automated Analysis

Anonymous authors
Paper under double-blind review

Abstract

Time series analysis is crucial in real-world applications, yet traditional methods focus on isolated tasks only, and recent studies on time series reasoning remain limited to either single-step inference or are constrained to natural language answers. In this work, we introduce TS-Reasoner, a domain-specialized agent designed agent designed for multi-step time series inference. By integrating large language model (LLM) reasoning with domain-specific computational tools and error feedback loop, TS-Reasoner enables domain-informed, constraint-aware analytical workflows that combine symbolic reasoning with precise numerical analysis. We assess the system's capabilities along two axes: 1) fundamental time series understanding assessed by TimeSeriesExam and 2) complex, multi-step inference, evaluated by a newly proposed dataset designed to test both compositional reasoning and computational precision in time series analysis. Experiments show that our approach outperforms standalone general-purpose LLMs in both basic time series concept understanding as well as the multi-step time series inference task, highlighting the promise of domain-specialized agents for automating real-world time series reasoning and analysis.

1 Introduction

Time series analysis lies at the heart of many high-impact real-world applications, powering decisions in domains such as finance, healthcare, and energy systems (Cheng et al., 2021; Sharma et al., 2021; Zhang et al., 2021). Over the past decades, research in time series modeling has seen significant advancements, not only in canonical tasks such as forecasting, anomaly detection, and classification (Hamilton, 2020; Lim & Zohren, 2021; Ismail Fawaz et al., 2019; Zamanzadeh Darban et al., 2024) but also the development of foundation models (Ansari et al., 2024; Liu et al., 2024b; Garza & Mergenthaler-Canseco, 2023; Goswami et al., 2024). The emergence of large-scale pretrained foundation models has further expanded the capabilities of time series analysis by enabling zero-shot inference. These models have proven to be powerful tools, pushing the boundaries of what is possible in time series modeling. While such advancements are promising, the fixed input-output paradigm in current time series deep learning models restricts their practical applicability. However, time series analysis routinely exhibits procedural complexity in practice when situated in real-world contexts. For example, a human expert would take multiple steps including retrieving relevant covariates, applying forecasting tools, validating operational constraints like maximum allowable system load, and refining the forecast accordingly to predict electricity load in a power grid (WANG et al., 2016).

Addressing the complexities of practical inference needs a shift toward intelligent agents that can flexibly compose and execute customized workflows for diverse inference tasks. While time series foundation models excel in current benchmarks, they lack the flexibility to handle diverse, multi-stage inference tasks and cannot generalize to scenarios requiring instruction following, constraint reasoning, or domain knowledge incorporation. Conversely, large language models (LLMs) demonstrate emerging reasoning (Mondorf & Plank, 2024) and instruction-following abilities (Zeng et al., 2023), making them a promising foundation for intelligent time series analysis agent. However, LLMs struggle with detecting temporal patterns and numerical computations. These failures stem from two key limitations: (1) discretization of continuous data disrupts temporal input (Spathis & Kawsar, 2024), and (2) LLMs are optimized for classification-style next-token prediction (Li et al., 2024), not numerical estimation (Cvejoski et al., 2022; Selvam, 2025). Addressing these limitations via full retraining or architectural overhaul is costly and inefficient. Hence, we call for a simple yet effective domain specialization of LLM (Ling et al., 2023) through a program-aided solution that integrates time series analysis specific tools. We introduce a time series domain-specialized Agent, TS-Reasoner, for multi-step time series inference. TS-Reasoner augments language-based reasoning with precise numerical computation by decomposing high-level instructions into structured workflows composed of domain operators. Using expert demonstration of tool usage in-context learning (Brown et al., 2020), TS-Reasoner effectively tackles the novel multi-step time series inference task introduced. In addtion, TS-Reasoner's ability to incorporate error feedback allows it to refine intermediate reasoning steps and correct miscalculations during execution.

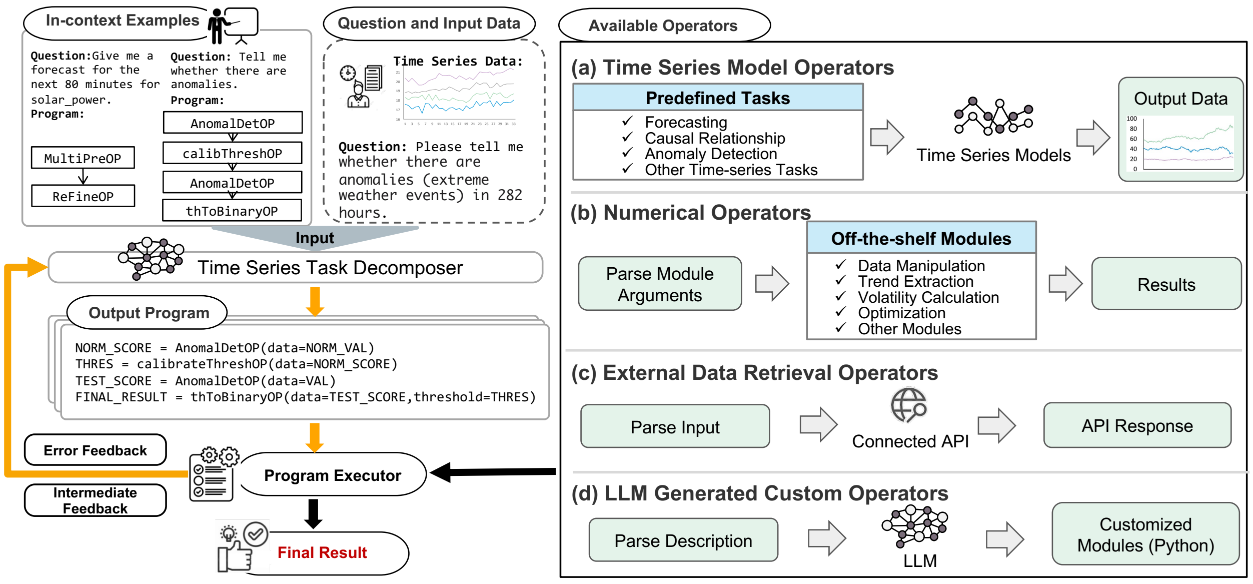
Even though TS-Reasoner is designed to tackle general multi-step time series inference tasks, robust evaluation requires grounding its capabilities in a suite of representative problems. We begin by assessing TS-Reasoner's basic time series understanding using the TimeSeriesExam benchmark (Cai et al., 2024). To further evaluate its ability to handle real-world complexity, we introduce a new dataset composed of carefully curated questions that reflect recurring challenges across practical domains. Specifically, our benchmark includes predictive tasks such as constraint-aware forecasting in smart grid operations (Han et al., 2021; WANG et al., 2016; Greif et al., 1999), as well as diagnostic tasks involving anomaly detection with threshold calibration (Yan et al., 2021) and causal discovery guided by domain knowledge (Aoki & Ester, 2020). These task types serve as proxies for broader categories of time series multi-step reasoning and collectively capture the procedural, analytical, and diagnostic complexity that TS-Reasoner is designed to address.

We summarize our contributions as follows: (1) We identify and formalize a critical gap in current time series modeling: real-world tasks often demand multi-step inference, constraint handling, domain knowledge integration, and dynamic workflow assembly which are capabilities missing from existing models and benchmarks. We define this as a new task paradigm which captures the procedural and compositional complexity of practical time series problems. (2) We propose TS-Reasoner, a time series domain specialized agent that leverages the LLM backbone to decompose complex multi-step tasks into structured workflows composed of well-defined operators to ensure numerical accuracy. (3) We construct a realistic benchmark grounded in scientific and engineering domains. Experiments across both basic and complex settings demonstrate TS-Reasoner's superior performance and reveal the limitations of current general-purpose LLM agents in time series domain.

2 Related Work

Traditional Time Series Tasks and Foundation Models Classical time series analysis encompasses key tasks such as forecasting Ekambaram et al. (2023), imputation Tashiro et al. (2021), classification Zhao et al. (2017), and anomaly detection Xu et al. (2021), each serving unique purposes across various domains. Traditionally, these tasks were addressed by dedicated models optimized for specific purposes, resulting in a fragmented approach. Inspired by the emergence of LLMs, there has been a shift toward general-purpose time series foundation models Woo et al. (2024); Liu et al. (2024b). LLMTime Gruver et al. (2024) encoded time series as strings, while TimeLLM Jin et al. (2023) and TimeMoE Shi et al. (2025) converted time series into language representations through alignment. TEMPO Cao et al. (2023) incorporated decomposition techniques and prompt design, enabling generalization to unseen data and multimodal scenarios. Additionally, Chronos (Ansari et al., 2024) employed scaling and quantization techniques for embedding time series. These models are pre-trained on diverse datasets to handle multiple preset tasks. However, they still operate under predefined task definitions, limiting their ability to perform complex and compositional reasoning. In addition, they lack the flexibility to adapt to intricate scenarios that require a deeper understanding of task instructions and the composition of different tasks or concepts.

LLM Reasoning & LLM-Based Agent Large Language Models (LLMs) have demonstrated emerging reasoning capabilities, particularly when aided by in-context learning (Huang & Chang, 2022; Qiao et al., 2022; Ahn et al., 2024). Techniques like Chain-of-Thought (CoT) prompting (Wei et al., 2022) and its extensions including Tree-of-Thoughts, Graph-of-Thoughts, and Self-Consistent CoT (Yao et al., 2023a;b;



```
NORM_SCORE = AnomalDetOP(data=NORM_VAL)
THRES = calibrateThreshOP(data=NORM_SCORE)
TEST_SCORE = AnomalDetOP(data=VAL)
FINAL_RESULT = thToBinaryOP(data=TEST_SCORE,threshold=THRES)
```

Figure 1: The architecture of TS-Reasoner. The LLM work as task decomposer, which learn from operator definitions and in-context examples to decompose task instances into sequence of operators. Then a program executor is responsible for solution plan execution and forms a feedback loop with task decomposer.

Wang et al., 2022; Qu et al., 2024) improve multi-step reasoning by encouraging structured intermediate steps. Moreover, program-based reasoning (Khot et al., 2022; Creswell & Shanahan, 2022; Gao et al., 2023) further extends LLM capabilities by integrating heterogeneous modules for various data modalities, such as VisProg (Gupta & Kembhavi, 2023) for visual reasoning which involves manipulating the image according to instructions.

Despite these advances, general-purpose LLMs often underperform in specialized domains. These challenges are commonly addressed through explicit tool augmentation, knowledge injection, or structured interfaces. SWE-agent (Yang et al., 2024) introduces an Agent-Computer Interface for autonomous software development. HoneyComb (Zhang et al., 2024) equips agents with a material science toolhub, while CRISPR-GPT (Huang et al., 2024) integrates biology-specific tools and domain knowledge for gene editing. Feedback mechanisms are also essential (Cai et al., 2025). Frameworks like AdaPlanner (Sun et al., 2023) allow LLM agents to adaptively refine their plans in response to environmental feedback, leading to improved decision-making in sequential tasks. Within time series domain, recent works (Jin et al., 2024) have also pointed to the direction of next generation time series agents based on LLMs. Drawing on these inspirations, TS-Reasoner combines domain tools with language-based reasoning and self-correction to support robust, multi-step time series inference.

3 TS-Reasoner

In this section, we formally introduce TS-Reasoner which addresses the limitations of standalone LLMs in handling numerical precision and computational complexity in time series analysis by decomposing complex time series inference tasks into structured operator execution pipelines. The core architecture of TS-Reasoner can be expressed as:

$$R^i = \text{TaskDecomposer}(x, C, \text{feedback}^{i-1}) \tag{1}$$

$$\text{Exec}(R^i) = \begin{cases} y, & \text{if successful and valid} \\ \text{feedback}^i, & \text{if failed or poor quality} \end{cases} \tag{2}$$

where x denotes the input task described in natural language, C represents the in-context examples containing operator definitions and demonstrated operator usages in response to sample questions, $R^i = [f_1, f_2, ..., f_n]$ is the reasoning trace consisting of specialized operators planned by the task decomposer at iteration i, y represents TS-Reasoner's final output obtained from successful execution of R. Figure 1 provides an overview of TS-Reasoner. The framework follows a modular design philosophy, separating high-level reasoning from computational execution. By delegating computationally intensive or mathematically precise operations to specialized operators.

Task Decomposer is embodied by a LLM and is responsible for transforming complex natural language instructions into structured operator execution plans. Given the definitions of operators in the toolbox and a set of example task-solution pairs $(x_1, R_1), (x_2, R_2), ..., (x_n, R_n)$, TS-Reasoner learns to map new tasks to appropriate decomposition sequences imitating human breaking down complex problems into manageable subtasks. The in-context learning paradigm does not require expensive retraining and allows LLM to recognize structural patterns in time series analysis workflows.

Specialized Operators TS-Reasoner relies on a suite of modular operators that wrap existing statistical tools, machine learning and deep learning models, and numerical utilities. We adopt state-of-the-art or widely used components and expose them through a unified interface. Each operator is given an informative name (e.g., ForecastOP, AnomalDetOP, calibrateThreshOP) and follows a consistent calling structure, enabling seamless orchestration within the execution pipeline. This standardized abstraction allows the task decomposer to compose complex workflows from heterogeneous tools in a model-agnostic and reusable manner. The available operators can generally be categorized into four groups, each addressing distinct aspects of time series analysis. **1) Time Series Model Operators** These operators form the analytical core of TS-Reasoner, applying machine learning and deep learning models to time series data. They handle canonical time series analysis tasks such as forecasting, imputation, and anomaly detection using models like ARIMA (Box & Jenkins, 1968), Chronos (Ansari et al., 2024), Lag-Llama (Rasul et al., 2023), and MOMENT (Goswami et al., 2024). Depending on the task, the amount of input data, and time series model choice, this family of operators can work in zero-shot mode or finetuning mode. **2) Numerical Operators** These provide standard statistical and mathematical functions used throughout the solution pipeline as well as simple shape manipulation operators to ensure seamless passing of variables. For example, operators within this family are equipped with functionalities like computing averages, performing decomposition, measuring volatility, or evaluating distribution properties.**3) External Data Retrieval Operators** These operators fetch relevant contextual data from external sources. The implementation of these operators mainly involves wrapping API calls to available services that supply time series based on queries. **4) LLM-Generated Custom Operators** To address the impracticality of predefining all necessary time series tools, we introduce a LLM-generated custom operator. This is a single operator that accepts a natural language prompt and returns executable code tailored to the specific analytical requirement, enabling a flexible and extensible pipeline that adapts to diverse task demands. The current toolkit covers 57 operators, a complete list of operators within each category is outlined in section C.

Feedback Mechanism for Self-Refinement Once the task decomposer generates a structured execution plan $R = [f_1, f_2, ..., f_n]$ that specifies the sequence and order of operators, the program executor interprets and executes each step accordingly. If an execution error is encountered, a feedback loop is initiated: the task decomposer is re-invoked to produce a revised end-to-end execution plan using the original query, in-context examples, the previously generated solution, and the error message from the current execution attempt (as shown in Equation 2). This error correction process is allowed to retry up to k times, where k is a predefined maximum number of retry attempts.

With the abundant suite of available time series foundation models, using time series model operators often necessitates model selection. In realistic settings, an intelligent agent may benefit from intermediate feedback either from automated evaluators or human/external signals to refine its model choices. Conditioned on such support from the evaluation benchmark, TS-Reasonerleverages feedback on prediction quality (e.g., MAPE) to experiment with different model choices and select the best-performing one. The program executor maintains a buffer memory of previously explored solution paths to prevent redundant evaluations. If the

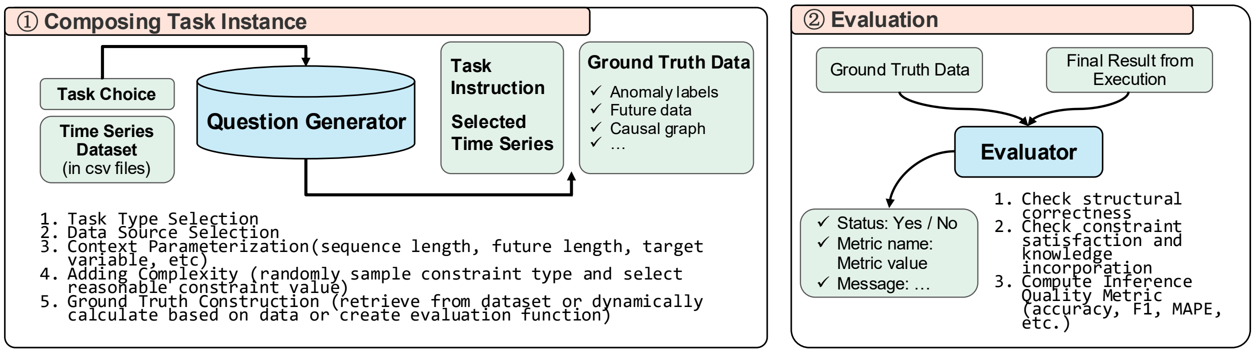

① Composing Task Instance
Task Choice
Time Series Dataset (in csv files)
Question Generator
Task Instruction
Selected Time Series
Ground Truth Data
✓ Anomaly labels
✓ Future data
✓ Causal graph
✓ …
1. Task Type Selection
2. Data Source Selection
3. Context Parameterization(sequence length, future length, target variable, etc)
4. Adding Complexity (randomly sample constraint type and select reasonable constraint value)
5. Ground Truth Construction (retrieve from dataset or dynamically calculate based on data or create evaluation function)
② Evaluation
Ground Truth Data
Final Result from Execution
Evaluator
✓ Status: Yes / No
✓ Metric name: Metric value
✓ Message: …
1. Check structural correctness
2. Check constraint satisfaction and knowledge incorporation
3. Compute Inference Quality Metric (accuracy, F1, MAPE, etc.)

Figure 2: The proposed pipeline for multi-step time series inference task instance generation and evaluation protocol.

task decomposer reverts to a previously attempted solution, the program execution proceeds directly to completion. This architecture ensures error handling, and adaptive refinement, enabling TS-Reasoner agent to dynamically optimize multi-step time series inference.

4 Multi-Step Time Series Inference Dataset

To enable the study of multi-step time series inference in realistic settings, we construct a benchmark dataset that reflects the complexity of real-world reasoning tasks. Unlike traditional time series datasets that focus narrowly on a single task, our benchmark includes diverse task types requiring constraint handling and domain knowledge integration in addition to time series analysis. All tasks are framed in natural language, paired with context-rich time series inputs and corresponding ground truth outputs.

We categorize the tasks into two broad classes: predictive tasks and diagnostic tasks. 1) Predictive tasks focus on time series forecasting under realistic operational constraints. These include requirements such as limiting the maximum allowable load in a grid, enforcing electricity load ramp rate restrictions, or bounding variability within an acceptable range (Greif et al., 1999; Oreshkin et al., 2020). Such constraints are common in engineering and energy systems, where violations may result in equipment damage, service disruption, or regulatory penalties (Nti et al., 2020). Our dataset simulates scenarios where future values must not only be predicted accurately but also satisfy operational constraints or leverage domain knowledge. To reflect practical challenges, we include forecasting scenarios with and without covariates data supplied in the question, and also scale to multi-grid settings where large volumes of data must be processed across multiple regions. 2) Diagnostic tasks emphasize pattern recognition, calibration, and structural inference. One subclass involves anomaly detection, where the model must identify abnormal patterns in a time series, often using reference samples or contextual priors such as known anomaly rates or anomaly-free samples to calibrate thresholds (Liu et al., 2016). These tasks simulate settings like extreme weather detection, where defining anomaly boundaries is not straightforward and depends on prior data distributions. Another subclass of diagnostic tasks centers on causal discovery. Here, the model is asked to infer causal relationships among variables, given partial domain knowledge such as the expected proportion of causal links (Aoki & Ester, 2020). These tasks require the integration of weak supervision with statistical reasoning (Huang et al., 2020) and offer a rich testbed.

Figure 2 shows the pipeline used to generate proposed pipeline. Each task instance is generated via a modular pipeline grounded in real-world multi-step time series analysis. We first survey existing literature to identify three task definitions that demand multi-step inference: constraint-aware forecasting, extreme weather detection with known prior and causal discovery with domain knowledge. These are annotated with natural language templates reflecting their problem definition and objectives. To instantiate a task, we source time series data from datasets from corresponding domains: ERA5[1] (climate), PSML (Zheng et al., 2021) (electricity), and synthetic generators (causal graphs). Each specific task instance is parameterized by

[1]https://climatelearn.readthedocs.io/en/latest/user-guide/tasks_and_datasets.html#era5-dataset

selecting time ranges, target variable, and context-specific constraints/knowledge. The result is a dynamic task specification with varying evaluation criteria across instances, distinguishing from traditional fixed-protocol benchmarks. For more details on specific task instance compilation, please see Proposed Dataset section in appendix.

Evaluation is conducted through a unified but flexible protocol. Each task instance is paired with a dedicated evaluator configured with access to ground truth data, contextual information, and task-specific constraints or domain knowledge. The evaluator assesses the final model output along three dimensions: structural correctness (e.g., output shape and format), constraint satisfaction (e.g., anomaly rates, causal priors, load limits), and inference quality using appropriate metrics such as accuracy, F1 score, or MAPE. This framework supports rigorous and interpretable assessment across diverse task types. Such dedicated evaluators also support generating intermediate feedback quality signals like MAPE by comparing predictions to ground truth, simulating expert or human feedback in realistic decision-making workflows.

4.1 Task Instance Templates

In this section, we provide an outline of templates used for each type of tasks. The exact template for each sub question type may vary from each other to best reflect the available information: with and without covariates data supplied, whether multiple time series is supplied to test larger-scale multi-step inference cases, what known priors are provided (anomaly rate or reference free sample or known ratio of causal links), and which constrains are selected (global electricity load variability limit, maximum alloable load, e.t.c). The current scale of the proposed benchmark consists of around 500 specific task instances. Figure 3 demonstrates sample questions generated from the task instance generation pipeline. [2].

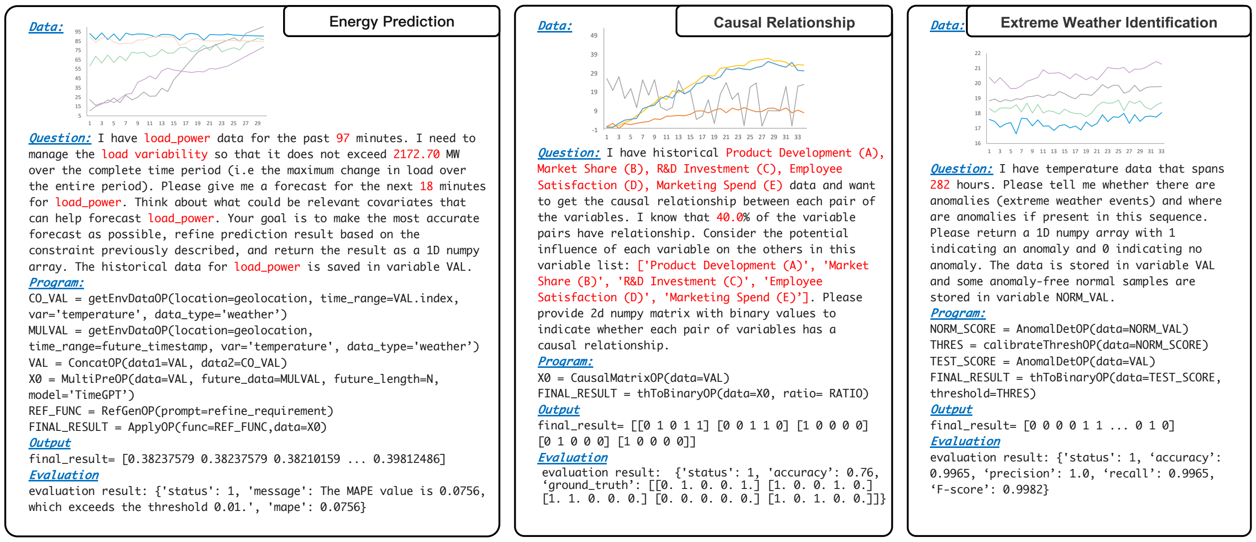
Predictive Tasks In predictive tasks, we primarily focus on load-related issues in the energy sector. Specifically, each test sample provides the model with a natural language question, relevant time series historical data, and operational constraint requirements. The questions are generated by the following templates:

Question Template (Electricity Load Prediction) *I have historical {influence variables} data and the corresponding target variable data for the past {historical length} minutes. [1. I need to ensure that the maximum allowable system load does not exceed {load value} MW. 2. I require that the system load is maintained above a minimum of {load value} MW. 3. I must monitor the load ramp rate to ensure it does not exceed {constraint value} MW for each time step. 4. I need to manage the load variability so that it does not exceed {constraint value} MW over the given period.] Think about how {influence variables} influence {target variable}. Please give me a forecast for the next {future length} minutes for target variable. Your goal is to make the most accurate forecast as possible, refine prediction result based on the constraint previously described. {output requirement: E.g. please return a 1D numpy array}.{data variable name specification: E.g. The historical data is stored in variable VAL.}* .

Diagnostic Tasks In diagnostic tasks, we primarily used extreme weather detection scenarios in climate science and synthetic causal discovery. Specifically, each task instance includes natural language description of the question, time series, and domain knowledge description. The extreme weather detection questions are generated by the following template:

Question Template (Extreme Weather Detection) *I have 2m temperature data that spans {sequence length} hours. Please tell me whether there are anomalies (extreme weather events) and where are anomalies if present in this sequence. [1. I also have some anomaly-free 2m temperature data from the same region. 2. I know that {anomaly ratio} percent of the times have anomalies.] {output requirement}.{data variable name specification}* .

[2]We also share a few sample questions here https://anonymous.4open.science/r/sample_dataset/



```
Energy Prediction
Data:
Question: I have load_power data for the past 97 minutes. I need to
manage the load variability so that it does not exceed 2172.70 MW
over the complete time period (i.e the maximum change in load over
the entire period). Please give me a forecast for the next 18 minutes
for load_power. Think about what could be relevant covariates that
can help forecast load_power. Your goal is to make the most accurate
forecast as possible, refine prediction result based on the
constraint previously described, and return the result as a 1D numpy
array. The historical data for load_power is saved in variable VAL.
Program:
CO_VAL = getEnvDataOP(location=geolocation, time_range=VAL.index,
var='temperature', data_type='weather')
MULVAL = getEnvDataOP(location=geolocation,
time_range=future_timestamp, var='temperature', data_type='weather')
VAL = ConcatOP(data1=VAL, data2=CO_VAL)
X0 = MultiPreOP(data=VAL, future_data=MULVAL, future_length=N,
model='TimeGPT')
REF_FUNC = RefGenOP(prompt=refine_requirement)
FINAL_RESULT = ApplyOP(func=REF_FUNC,data=X0)
Output
final_result= [0.38237579 0.38237579 0.38210159 ... 0.39812486]
Evaluation
evaluation result: {'status': 1, 'message': The MAPE value is 0.0756,
which exceeds the threshold 0.01.', 'mape': 0.0756}
```

```
Causal Relationship
Data:
Question: I have historical Product Development (A),
Market Share (B), R&D Investment (C), Employee
Satisfaction (D), Marketing Spend (E) data and want
to get the causal relationship between each pair of
the variables. I know that 40.0% of the variable
pairs have relationship. Consider the potential
influence of each variable on the others in this
variable list: ['Product Development (A)', 'Market
Share (B)', 'R&D Investment (C)', 'Employee
Satisfaction (D)', 'Marketing Spend (E)']. Please
provide 2d numpy matrix with binary values to
indicate whether each pair of variables has a
causal relationship.
Program:
X0 = CausalMatrixOP(data=VAL)
FINAL_RESULT = thToBinaryOP(data=X0, ratio= RATIO)
Output
final_result= [[0 1 0 1 1] [0 0 1 1 0] [1 0 0 0 0]
[0 1 0 0 0] [1 0 0 0 0]]
Evaluation
 evaluation result:  {'status': 1, 'accuracy': 0.76,
 'ground_truth': [[0. 1. 0. 0. 1.] [1. 0. 0. 1. 0.]
 [1. 1. 0. 0. 0.] [0. 0. 0. 0. 0.] [1. 0. 1. 0. 0.]]}
```

```
Extreme Weather Identification
Data:
Question: I have temperature data that spans
282 hours. Please tell me whether there are
anomalies (extreme weather events) and where
are anomalies if present in this sequence.
Please return a 1D numpy array with 1
indicating an anomaly and 0 indicating no
anomaly. The data is stored in variable VAL
and some anomaly-free normal samples are
stored in variable NORM_VAL.
Program:
NORM_SCORE = AnomalDetOP(data=NORM_VAL)
THRES = calibrateThreshOP(data=NORM_SCORE)
TEST_SCORE = AnomalDetOP(data=VAL)
FINAL_RESULT = thToBinaryOP(data=TEST_SCORE,
threshold=THRES)
Output
final_result= [0 0 0 0 1 1 ... 0 1 0]
Evaluation
evaluation result: {'status': 1, 'accuracy':
0.9965, 'precision': 1.0, 'recall': 0.9965,
'F-score': 0.9982}
```

Figure 3: Examples of End-to-End Tasks on Time Series. These tasks require the model to perform multi-step reasoning on time series data.

For causal mining, we synthesize a set of data grounded in domain knowledge related to climate science, finance, and economics. Specifically, we generate a series of multivariate time series data based on established causal relationships and meteorological principles. Each test sample consists of a time series dataset of various variables, accompanied by a natural language instruction that asks the model to uncover the causal dependencies between the given time series, and expert knowledge of percentage of true relationships. The reasoning model must infer dependencies based on the given data and instructions. The evaluation framework then measures the model's performance by comparing its inferred causal relationships with the ground truth. The questions are generated by the following template:

Question Template (Causal Discovery) *I have historical {variable names} data and want to get the causal relationship between each pair of the variables. I know that {ratio}% of the variable pairs have relationship. Consider the potential influence of each variable on the others in this variable list: variable names. {output requirement}.{data variable name specification}*

4.2 Dataset Statistics

Table 1 summarizes the dataset compiled for the multi-step time series inference task. The energy data with covariates contained electricity load from 6 major electricity grids in the United States (MISO, ERCOT,CAISO,NYISO,PJM, SPP) across 66 zones for 3 complete years (2018 -2020) with a minute level frequency [3]. The energy data without covariates contain both geolocation and load data for sub-regions within electricity grids at an hourly frequency. The data is retrieved from official websites of ERCOT, MISO, NYISO[4][5][6]. The climate data consisted of the ERA5 reanalysis dataset which is the most popular climate dataset. The extreme weather detection dataset is obtained from running scripts in climatelearn scripts[7].

[3]https://github.com/tamu-engineering-research/Open-source-power-dataset
[4]https://www.nyiso.com/load-data
[5]https://www.ercot.com/gridinfo/load/load_hist
[6]https://www.misoenergy.org/markets-and-operations/real-time--market-data/market-reports
[7]https://climatelearn.readthedocs.io/en/latest/user-guide/tasks_and_datasets.html#era5-dataset

Dataset	Number of CSVs	Avg Total Timestamps	Number of Variables
Climate Data	624	526	2048
Energy Data	22	8760	1-3
Energy Data w/ Covariates	66	872601	11
Causal Data	8	529	3–6

Table 1: Dataset Statistics of the constructed dataset. The exact number of time series are not calculated because it depends on randomly sampled sequence length when generating task instances.

5 Experiments

In this section, we conduct a series of comparative experiments to evaluate the performance of various LLM agents on both fundamental and complex time series understanding tasks. The evaluation includes multiple-choice questions assessing basic time series concepts, as well as more advanced inference tasks supported by the proposed dataset. Our baseline models include proprietary LLMs: GPT-4o (Hurst et al., 2024), DeepSeek model (Liu et al., 2024a), reasoning model GPT-o1 (Jaech et al., 2024).Additionally, we assess the ReAct agent based on GPT-4o (Yao et al., 2022), which synergizes reasoning and acting by generating traces for each in an interleaving manner. TS-Reasoner uses GPT-4o (Hurst et al., 2024) as the task decomposer. For all experiments, we perform a single run using the same hyperparameters for the most deterministic LLM output decoding: temperature=0.0 and top_p=1.0 for the most deterministic output. All experiments are ran using a single NVIDIA A40 GPU with 48G memory.

5.1 Basic Time Series Understanding

To evaluate the fundamental capabilities of these models, we utilize the publicly available TimeSeriesExam benchmark (Cai et al., 2024) consisted of multiple choice questions. This benchmark assesses understanding across five key time series concepts: Anomaly Detection, Pattern Recognition, Noise Understanding, Causality Analysis, and Similarity Analysis. We use strict accuracy as evaluation metrics where only the final answer within the response of LLM is checked against the ground truth answer instead of the complete response [8].

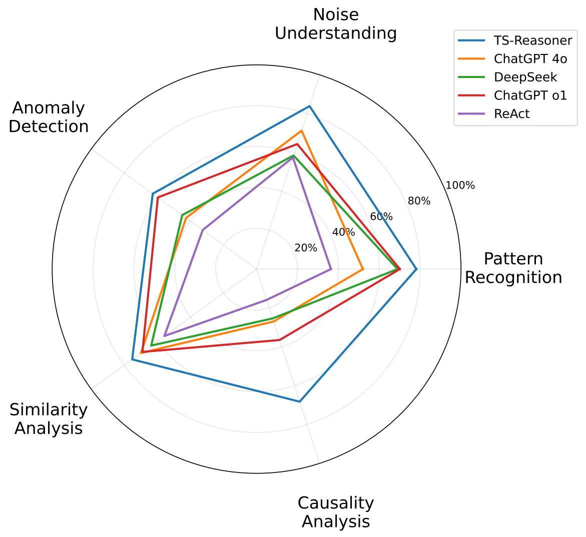


Figure 4: Strict Accuracy of TS-Reasoner and general purpose LLMs on the TimeSeriesExam.

As illustrated in Figure 4, TS-Reasoner consistently outperforms all baseline models across all dimensions. Notably, general-purpose LLMs exhibit particularly weak performance in causality analysis, where TS-Reasoner achieves an 87% improvement over the best-performing baseline. This significant advantage is largely attributed to TS-Reasoner 's integration with program-aided tools for statistical analysis, enabling it to determine autocorrelation, lag relationships, and causal dependencies through numerical computations. The results underscore the benefits of task decomposition and external program utilization in enhancing time series understanding rather than plain text reasoning.

[8]It was agreed by the original authors upon communication on this alternative metric that it better reflects the capability of LLMs.

5.2 Multi-Step Time Series Inference

Given the novel new paradigm of this task, there is a lack of suitable baselines models. For general purpose LLMs, we adopt a code-based approach due to two primary challenges: loss of precision of time series data in textual and image form and the limitation of context length encountered for tasks involving large amounts of time series data. To address these issues, we use the CodeAct agent implemented in agentscope (Gao et al., 2024; Wang et al., 2024). The CodeAct agent is equipped with Jupyter python interpreter and error feedback mechanism. Baseline models are prompted to autonomously generate code pipelines to solve the given tasks. Each agent has a maximum of k interaction turns to reach a solution.

Table 2 presents the performance of TS-Reasoner and baseline models on predictive tasks. Mean Absolute Percentage Error (MAPE) is chosen as the primary inference quality metric to ensure scale-invariant evaluation. TS-Reasoner outperforms baseline models in success rates while simultaneously achieving superior inference quality as evidenced by lower MAPE values. MAPE values exceeding 1 are considered unreasonable inference and counted towards failures. Among successful task completions,TS-Reasoner demonstrates the most performance gain on prediction tasks without explicitly provided covariates tasks. This improvement is largely attributed to its ability to retrieve relevant covariates deemed necessary by the task decomposer.

Table 2: Performance on Multi-Step Predictive Tasks. TSR denotes TS-Reasoner, TSR w/ PQ denotes TS-Reasoner prompted with paraphrased questions. LLM models are equipped with a CodeAct agent to receive execution feedback. The average MAPE values are calculated among all succeeded questions and the corresponding standard deviation is calculated.

Constraint Type	**Max Load**		**Min Load**		**Load Ramp Rate**		**Load Variability**		**Average**	
	Success Rate	MAPE (Std) ↓	Success Rate	MAPE (Std)↓	Success Rate	MAPE (Std)↓	Success Rate	MAPE (Std)↓	Success Rate	MAPE ↓
Prediction w/ Covariates										
TSR	1	0.0621 (0.10)	1	0.0564 (0.06)	1	0.0719 (0.20)	0.9444	0.0577 (0.06)	**0.9861**	0.0620
TSR w/ PQ	0.9474	0.0467 (0.08)	1	0.0564 (0.06)	0.9444	0.0659 (0.17)	0.9444	0.0577 (0.06)	0.9591	**0.0567**
4o	1	0.0685 (0.10)	1	0.1289 (0.19)	0.7778	0.0847 (0.19)	0.5	0.0443 (0.05)	0.8194	0.0816
o1	1	0.1289 (0.19)	0.9	0.0748 (0.07)	1	0.0861 (0.16)	0.6667	0.0556 (0.06)	0.8917	0.0864
Deepseek	1	0.1444 (0.22)	1	0.1387 (0.16)	0.8333	0.1180 (0.24)	0.6667	0.0835 (0.07)	0.8750	0.1212
ReAct	0.9474	0.0689 (0.10)	0.9	0.0845 (0.09)	0.8333	0.1021 (0.23)	0.9444	0.0630 (0.07)	0.9063	0.0796
Prediction w/o Covariates										
TSR	1	0.0799 (0.11)	1	0.1366 (0.20)	0.85	0.1191 (0.17)	0.85	0.0767 (0.04)	**0.9250**	0.1031
TSR w/ PQ	1	0.0806 (0.11)	1	0.1366 (0.20)	0.85	0.0763 (0.03)	0.85	0.0812 (0.05)	**0.9250**	**0.0937**
4o	0.8	0.1607 (0.17)	0.9	0.2007 (0.20)	0.5	0.3430 (0.28)	0.75	0.1838 (0.12)	0.7375	0.2220
o1	0.85	0.1518 (0.06)	0.8	0.1439 (0.06)	0.65	0.1433 (0.05)	0.5	0.1672 (0.07)	0.7000	0.1268
Deepseek	1	0.1612 (0.14)	0.9	0.1978 (0.19)	0.85	0.2042 (0.09)	0.6	0.1655 (0.07)	0.8375	0.1822
ReAct	0.55	0.1838 (0.12)	0.65	0.2543 (0.17)	0.4	0.4262 (0.37)	0.75	0.2861 (0.22)	0.5875	0.2876
Prediction across Multiple Grids										
TSR	0.9	0.1491 (0.18)	1	0.1614 (0.18)	0.9	0.1285 (0.13)	1	0.1752 (0.25)	**0.9500**	0.1535
TSR w/ PQ	0.85	0.1468 (0.18)	0.95	0.1647 (0.18)	0.95	0.1301 (0.12)	0.9	0.1453 (0.19)	0.9125	**0.1467**
4o	0.8	0.2318 (0.30)	0.85	0.3071 (0.35)	0.6	0.3077 (0.36)	0.55	0.5538 (0.44)	0.7000	0.3501
o1	0.85	0.3095 (0.37)	0.85	0.2604 (0.26)	0.95	0.1877 (0.23)	0.7	0.3222 (0.34)	0.8375	0.2708
Deepseek	0.85	0.1889 (0.25)	1	0.1792 (0.23)	0.75	0.3830 (0.40)	0.55	0.9093 (0.29)	0.7875	0.3416
ReAct	0.75	0.2229 (0.22)	0.65	0.2110 (0.23)	0.85	0.2591 (0.29)	0.8	0.1790 (0.23)	0.7625	0.2175

Figure 5 shows the performance of TS-Reasoner and baseline models for anomaly detection with threshold calibration and causal discovry with domain knowledge tasks. The upper-right corner represents the ideal problem solver that achieves both a high success rate and high inference quality among successful cases. We observe that TS-Reasoner dominates all baseline models in this aspect, highlighting its superior ability to integrate domain knowledge, and perform structured reasoning over complex time series analysis tasks.

Error Analysis In this section, we examine the failure modes of TS-Reasoner and baseline models. By systematically categorizing errors into execution failures, constraint violations, and insufficient/trivial predictions, we can isolate the fundamental weaknesses in current approaches. Execution failures include runtime errors or invalid outputs, while constraint violations refer to outputs that ignore task-specific requirements such as anomaly rates or variability limits. Inadequate results are defined as those yielding poor utility, such as MAPE > 1, forecasting accuracy $= 0$, or F1 score $= 0$. Figure 6 presents error distributions across different models on the electricity load prediction task without covariates. TS-Reasoner demonstrates superior robustness with a 92.5% success rate in both variants, with failures limited to constraint violations

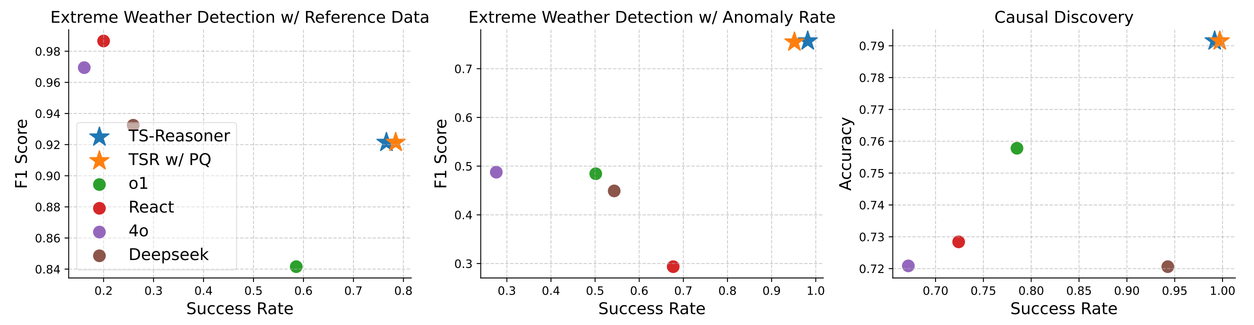


Figure 5: Performance on Multi-Step Diagnostic Tasks. A small jittering noise of 0.01 is added to success rate to distinguish overlapping points. TSR w/ PQ denotes TS-Reasoner when prompted with paraphrased questions.

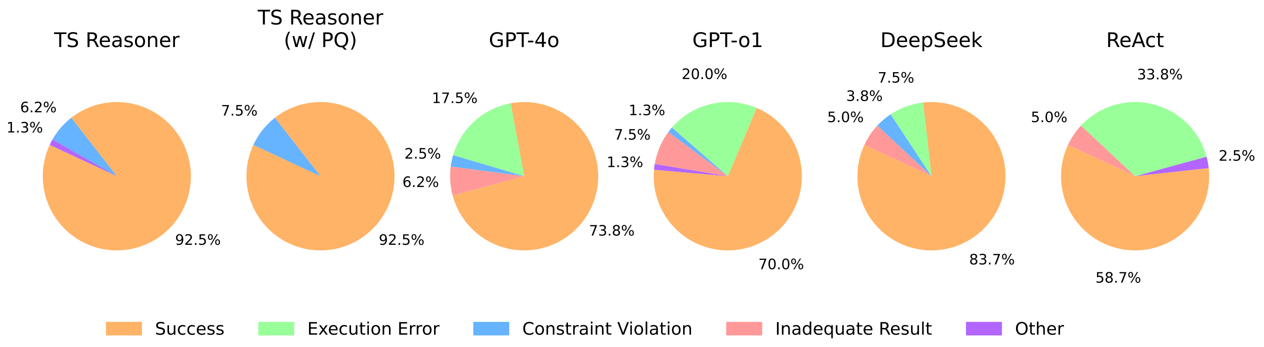


Figure 6: Error distribution of different approaches on electricity prediction task without covariates.

(6.2-7.5%). This performance significantly outpaces baseline models, which exhibit more diverse and severe error patterns.

General-purpose LLMs struggle primarily with execution errors, indicating difficulties generating valid computational pipelines for time series analysis. The ReAct agent, despite its explicit reasoning approach, achieves only a 58.7% success rate with the highest proportion of execution errors (33.8%) due to the complete lack of error feedback. The absence of execution errors in TS-Reasoner highlights the effectiveness of its modular architecture with well-tested computational components. This analysis underscores the necessity of domain-specialized agents like TS-Reasoner for complex time series tasks in real-world applications, as evidenced by its 19-34% higher success rates compared to baseline approaches. On the other hand, ReAct agent with explicit thinking can significantly reduce the constraint violation rate but still suffers from execution failures and generating trivial predictions.

Ablation Study Figure 7 presents an ablation study on the Electricity Prediction with Covariates task, where we progressively remove key components of TS-Reasoner to assess their individual contributions. The full agent achieves a 92.5% success rate with a MAPE of 0.1031. Removing intermediate feedback slightly reduces the success rate to 91.3%, with inadequate results beginning to emerge. When both intermediate and error feedback are removed, the success rate drops further to 90.0%, and execution errors begin to appear. The most significant degradation occurs when special operators are removed, reverting to a general-purpose LLM executor equipped with CodeAct. In this condition, the success rate falls sharply to 73.8%, and the MAPE more than doubles to 0.2220. These results demonstrate that feedback mechanisms play a critical role in securing a high success rate, while specialized time series operators are essential for maintaining strong inference quality.

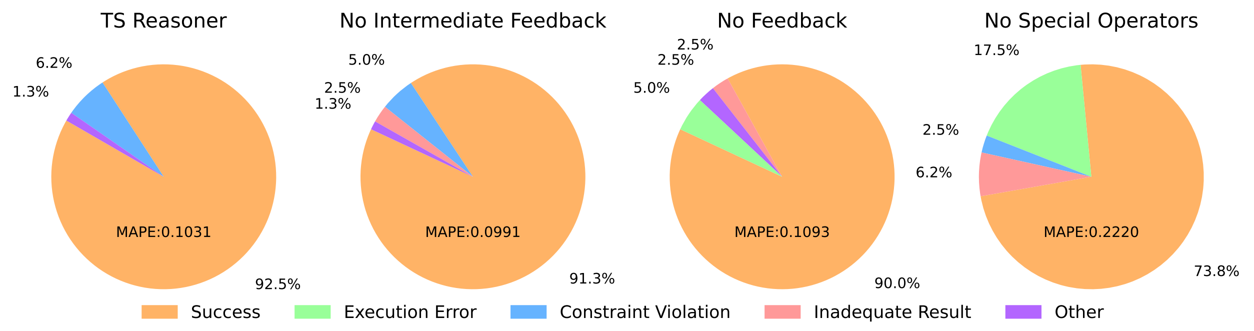


Figure 7: Ablation Study on Electricity Prediction w/ Covariates task. We removes each component in order, we first remove intermediate feedback only, then we remove both intermediate and error feedback (denoted as No Feedback), and lastly we remove the special operators and falling back to the general purpose LLM equipped with CodeAct.

Table 3: Causal Knowledge task: generalization to more ambiguous qualitative domain knowledge with a single added in-context example. Stock Price Future Volatility task: generalization to unseen domain and instruction.

Task	**Causal Known Prior**		**Stock Future Volatility**	
	SR	Acc ↑	SR	MAPE ↓
TSR	**1**	**0.739 (0.13)**	**0.98**	**0.620 (0.23)**
4o*	0.68	0.674 (0.16)	0.28	0.837 (0.19)
o1*	0.7	0.674 (0.14)	0.42	0.841 (0.18)
DS*	0.76	0.682 (0.15)	0.2	0.745 (0.30)
ReAct	0.66	0.659 (0.15)	0.16	0.970 (0.06)

6 Analysis

Generalizing to Qualitative Constraints and New Domains We further consider the practical utility of TS-Reasoner to handle novel scenarios without extensive retraining or specialized adaptation. We evaluate TS-Reasoner's adaptability to novel constraint types and previously unseen domains through two generalization tasks, with results presented in Table 3. First, we introduce a causal discovery task incorporating qualitative constraints that require reasoning about directional relationships rather than precise numerical quantity. With just a single additional in-context example added to demonstrate the handling of such qualitative constraints, TS-Reasoner achieves strong performance over baselines, demonstrating effective transfer learning capabilities. Secondly, we test adaptation to stock future volatility prediction task featuring unseen domain and instruction from the in-context samples. TS-Reasoner requires no explicit in-context examples for this task; instead, the task decomposer successfully generalizes by using its knowledge of toolkit functionalities to autonomously construct valid solution paths. These results demonstrate TS-Reasoner's generalization capabilities across both new constraint and application domains.

6.1 Reliability of Base Model

To assess the reliability of TS-Reasoner with respect to the choice of base language model, we conducted experiments substituting ChatGPT-4o (Hurst et al., 2024) with a smaller model, LLaMA 3.1 70B (Dubey et al., 2024), across the same set of tasks (Table 2). Despite the reduced capacity of LLaMA 3.1 70B, TS-Reasoner maintained comparable task success rates and inference quality across all evaluated settings. This consistency demonstrates that TS-Reasoner's performance is not solely reliant on the raw reasoning capacity of the underlying language model; instead, its structured execution pipeline and modular operator design enable it to generalize effectively even when the base decomposer model is weaker. The results highlight the reliability and adaptability of TS-Reasoner to different backbone models, ensuring robustness across deployments with varying LLM resources.

Table 4: Performance Comparison between TS-Reasoner (GPT-4o) and TS-Reasoner (LLaMA 3.1 70B) across various tasks.

Task Decomposer	**GPT-4o**			**LLaMA 3.1 70B**		
Task	**Success Rate**	**MAPE**	**Std**	**Success Rate**	**MAPE**	**Std**
Electricity w/ Covariates						
Max Load	1	0.0799	0.1128	1	0.0990	0.1055
Min Load	1	0.1366	0.1951	1	0.1623	0.2169
Load Ramp Rate	0.85	0.1191	0.1663	0.85	0.0928	0.0363
Load Variability	0.85	0.0767	0.0422	0.95	0.1534	0.2028
Electricity w/ Covariates						
Max Load	1	0.0621	0.1037	1	0.0677	0.1020
Min Load	1	0.0564	0.0605	1	0.0615	0.0539
Load Ramp Rate	1	0.0719	0.1991	1	0.0813	0.2032
Load Variability	0.9444	0.0577	0.0588	0.8889	0.0692	0.0928
Prediction Across Multiple Grids						
Max Load	0.9	0.1491	0.1750	0.85	0.1503	0.1800
Min Load	1	0.1614	0.1773	1	0.1614	0.1773
Load Ramp Rate	0.9	0.1285	0.1269	0.9	0.1324	0.1257
Load Variability	1	0.1752	0.2455	0.9	0.1938	0.2520
Task	**Success Rate**	**F1**	**Std**	**Success Rate**	**F1**	**Std**
Extreme Weather Detection w/ Reference Data	0.78	0.9214	0.1082	0.8	0.9232	0.1074
Extreme Weather Detection w/ Anomaly Rate	1	0.7569	0.0556	1	0.7569	0.0556
Task	**Success Rate**	**Accuracy**	**Std**	**Success Rate**	**Accuracy**	**Std**
Causal Discovery	1	0.7915	0.1246	1	0.7909	0.1239

7 Discussion and Limitation

Compared to traditional time series models, TS-Reasoner extends inference capabilities by supporting compositional reasoning and multi-step execution. However, TS-Reasoner only represents an initial step toward multi-step time series inference and cannot fully understands temporal signals. Unlike multimodal foundation models, which encode time series data, TS-Reasoner does not process raw time series data directly. Instead, it relies on task decomposition and structured execution. Further research effort is still needed to address challenges of semantic space alignment between time series and natural language. One limitation of TS-Reasoneris that generalizing to new tasks that requires tools beyond current toolbox may require additional toolbox engineering such as visualization tools. In addition, the success of TS-Reasoner relies in part on manually annotated in-context examples that demonstrate proper task decomposition and tool usage. A future direction of this work is to adaptively refine the in-context set by learning from execution errors and feedback signals, enabling the model to iteratively improve its reasoning workflows over time.

8 Conclusion

In this study, we introduced TS-Reasoner, a time series domain-specialized time series agent for multi-step inference tasks. TS-Reasoner integrates compositional reasoning with structured execution of expert tools. Through comprehensive benchmarking on TimeSeriesExam and a diverse set of real-world-inspired tasks that demonstrate multi-step complexity in nature, we evaluated TS-Reasoner's performance against leading general-purpose LLM-based agents. Our experiments demonstrate that TS-Reasoner consistently outperforms baseline models in both success rate and inference quality. The results reveal limitations of current general-purpose LLMs in multi-step time series analysis tasks, including frequent execution errors and suboptimal outputs. These findings suggest that LLMs alone are insufficient for such tasks, and that a hybrid approach combining LLMs with domain-specific operators is essential. By bridging LLM reasoning with time series analytical workflows, TS-Reasoner emerges as a simple yet effective solution, paving the way for more intelligent time series agents.

References

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*, 2024.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

Raquel Aoki and Martin Ester. Parkca: Causal inference with partially known causes. In *BIOCOMPUTING 2021: Proceedings of the Pacific Symposium*, pp. 196–207. World Scientific, 2020.

George EP Box and Gwilym M Jenkins. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 17(2):91–109, 1968.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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.

Shihao Cai, Jizhi Zhang, Keqin Bao, Chongming Gao, Qifan Wang, Fuli Feng, and Xiangnan He. Agentic feedback loop modeling improves recommendation and user simulation. In *Proceedings of the 48th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2025.

Yifu Cai, Arjun Choudhry, Mononito Goswami, and Artur Dubrawski. Timeseriesexam: A time series understanding exam. *arXiv preprint arXiv:2410.14752*, 2024.

Defu Cao, Furong Jia, Sercan O Arik, Tomas Pfister, Yixiang Zheng, Wen Ye, and Yan Liu. Tempo: Prompt-based generative pre-trained transformer for time series forecasting. *arXiv preprint arXiv:2310.04948*, 2023.

Yifang Cheng, Zachary Ross, Egill Hauksson, and Yehuda Ben-Zion. A refined comprehensive earthquake focal mechanism catalog for southern california derived with deep learning algorithms. In *AGU Fall Meeting Abstracts*, volume 2021, pp. S32A–05, 2021.

Antonia Creswell and Murray Shanahan. Faithful reasoning using large language models. *arXiv preprint arXiv:2208.14271*, 2022.

Kostadin Cvejoski, Ramsés J Sánchez, and César Ojeda. The future is different: Large pre-trained language models fail in prediction tasks. *arXiv preprint arXiv:2211.00384*, 2022.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pp. arXiv–2407, 2024.

Vijay Ekambaram, Arindam Jati, Nam Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pp. 459–469, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701030. doi: 10.1145/3580305.3599533. URL `https://doi.org/10.1145/3580305.3599533`.

Dawei Gao, Zitao Li, Xuchen Pan, Weirui Kuang, Zhijian Ma, Bingchen Qian, Fei Wei, Wenhao Zhang, Yuexiang Xie, Daoyuan Chen, Liuyi Yao, Hongyi Peng, Ze Yu Zhang, Lin Zhu, Chen Cheng, Hongzhu Shi, Yaliang Li, Bolin Ding, and Jingren Zhou. Agentscope: A flexible yet robust multi-agent platform. *CoRR*, abs/2402.14034, 2024.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pp. 10764–10799. PMLR, 2023.

Azul Garza and Max Mergenthaler-Canseco. Timegpt-1. *arXiv preprint arXiv:2310.03589*, 2023.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. *arXiv preprint arXiv:2402.03885*, 2024.

Claudia Greif, Raymond B Johnson, Chao an Li, Alva J Svoboda, and K Andrijeski Uemura. Short-term scheduling of electric power systems under minimum load conditions. *IEEE transactions on power systems*, 14(1):280–286, 1999.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G Wilson. Large language models are zero-shot time series forecasters. *Advances in Neural Information Processing Systems*, 36, 2024.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14953–14962, 2023.

James D Hamilton. *Time series analysis*. Princeton university press, 2020.

Kyo Beom Han, Jaesung Jung, and Byung O Kang. Real-time load variability control using energy storage system for demand-side management in south korea. *Energies*, 14(19):6292, 2021.

Biwei Huang, Kun Zhang, Jiji Zhang, Joseph Ramsey, Ruben Sanchez-Romero, Clark Glymour, and Bernhard Schölkopf. Causal discovery from heterogeneous/nonstationary data. *The Journal of Machine Learning Research*, 21(1):3482–3534, 2020.

Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*, 2022.

Kaixuan Huang, Yuanhao Qu, Henry Cousins, William A Johnson, Di Yin, Mihir Shah, Denny Zhou, Russ Altman, Mengdi Wang, and Le Cong. Crispr-gpt: An llm agent for automated design of gene-editing experiments. *arXiv preprint arXiv:2404.18021*, 2024.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*, 2023.

Ming Jin, Yifan Zhang, Wei Chen, Kexin Zhang, Yuxuan Liang, Bin Yang, Jindong Wang, Shirui Pan, and Qingsong Wen. Position: What can large language models tell us about time series analysis. In *41st International Conference on Machine Learning*. MLResearchPress, 2024.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*, 2022.

Yingcong Li, Yixiao Huang, Muhammed E Ildiz, Ankit Singh Rawat, and Samet Oymak. Mechanics of next token prediction with self-attention. In *International Conference on Artificial Intelligence and Statistics*, pp. 685–693. PMLR, 2024.

Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

Chen Ling, Xujiang Zhao, Jiaying Lu, Chengyuan Deng, Can Zheng, Junxiang Wang, Tanmoy Chowdhury, Yun Li, Hejie Cui, Xuchao Zhang, et al. Domain specialization as the key to make large language models disruptive: A comprehensive survey. *arXiv preprint arXiv:2305.18703*, 2023.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

Xu Liu, Juncheng Liu, Gerald Woo, Taha Aksu, Yuxuan Liang, Roger Zimmermann, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Moirai-moe: Empowering time series foundation models with sparse mixture of experts. *arXiv preprint arXiv:2410.10469*, 2024b.

Yunjie Liu, Evan Racah, Joaquin Correa, Amir Khosrowshahi, David Lavers, Kenneth Kunkel, Michael Wehner, William Collins, et al. Application of deep convolutional neural networks for detecting extreme weather in climate datasets. *arXiv preprint arXiv:1605.01156*, 2016.

Philipp Mondorf and Barbara Plank. Beyond accuracy: Evaluating the reasoning behavior of large language models–a survey. *arXiv preprint arXiv:2404.01869*, 2024.

Isaac Kofi Nti, Moses Teimeh, Owusu Nyarko-Boateng, and Adebayo Felix Adekoya. Electricity load forecasting: a systematic review. *Journal of Electrical Systems and Information Technology*, 7:1–19, 2020.

Boris N Oreshkin, Grzegorz Dudek, and Paweł Pełka. N-beats neural network for mid-term electricity load forecasting. *arXiv preprint arXiv:2009.11961*, 2020.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. Reasoning with language model prompting: A survey. *arXiv preprint arXiv:2212.09597*, 2022.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. *arXiv preprint arXiv:2407.18219*, 2024.

Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Arian Khorasani, George Adamopoulos, Rishika Bhagwatkar, Marin Biloš, Hena Ghonia, Nadhir Vincent Hassen, Anderson Schneider, et al. Lag-llama: Towards foundation models for time series forecasting. *arXiv preprint arXiv:2310.08278*, 2023.

Karthick Panner Selvam. Why large language models fail at precision regression, 2025. URL `https://karthick.ai/blog/2025/LLM-Regression/`.

Karishma Sharma, Yizhou Zhang, Emilio Ferrara, and Yan Liu. Identifying coordinated accounts on social media through hidden influence and group behaviours. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1441–1451, 2021.

Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-moe: Billion-scale time series foundation models with mixture of experts. In *The Twenty-First International Conference on Learning Representations*, 2025.

Dimitris Spathis and Fahim Kawsar. The first step is the hardest: Pitfalls of representing and tokenizing temporal data for large language models. *Journal of the American Medical Informatics Association*, 31 (9):2151–2158, 2024.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models. *Advances in neural information processing systems*, 36:58202–58245, 2023.

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems*, 34: 24804–24816, 2021.

Ke-qiu WANG, Si-guang SUN, Hong-yi WANG, Chang-xu JIANG, and Zhao-xia JING. 220kv city power grid maximum loadability determination with static security-constraints. *Power, Energy Engineering and Management (PEEM2016)*, pp. 1, 2016.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Forty-first International Conference on Machine Learning*, 2024.

Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Anomaly transformer: Time series anomaly detection with association discrepancy. *arXiv preprint arXiv:2110.02642*, 2021.

Xudong Yan, Huaidong Zhang, Xuemiao Xu, Xiaowei Hu, and Pheng-Ann Heng. Learning semantic context from normal samples for unsupervised anomaly detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 3110–3118, 2021.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=mXpq6ut8J3`.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023a.

Yao Yao, Zuchao Li, and Hai Zhao. Beyond chain-of-thought, effective graph-of-thought reasoning in language models. *arXiv preprint arXiv:2305.16582*, 2023b.

Zahra Zamanzadeh Darban, Geoffrey I Webb, Shirui Pan, Charu Aggarwal, and Mahsa Salehi. Deep learning for time series anomaly detection: A survey. *ACM Computing Surveys*, 57(1):1–42, 2024.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. Evaluating large language models at evaluating instruction following. *arXiv preprint arXiv:2310.07641*, 2023.

Huan Zhang, Yu Song, Ziyu Hou, Santiago Miret, and Bang Liu. Honeycomb: A flexible llm-based agent system for materials science. *arXiv preprint arXiv:2409.00135*, 2024.

Yizhou Zhang, Karishma Sharma, and Yan Liu. Vigdet: Knowledge informed neural temporal point process for coordination detection on social media. *Advances in Neural Information Processing Systems*, 34:3218–3231, 2021.

Bendong Zhao, Huanzhang Lu, Shangfeng Chen, Junliang Liu, and Dongya Wu. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162–169, 2017.

Xiangtian Zheng, Nan Xu, Loc Trinh, Dongqi Wu, Tong Huang, S Sivaranjani, Yan Liu, and Le Xie. Psml: a multi-scale time-series dataset for machine learning in decarbonized energy grids. *arXiv preprint arXiv:2110.06324*, 2021.

A Appendix

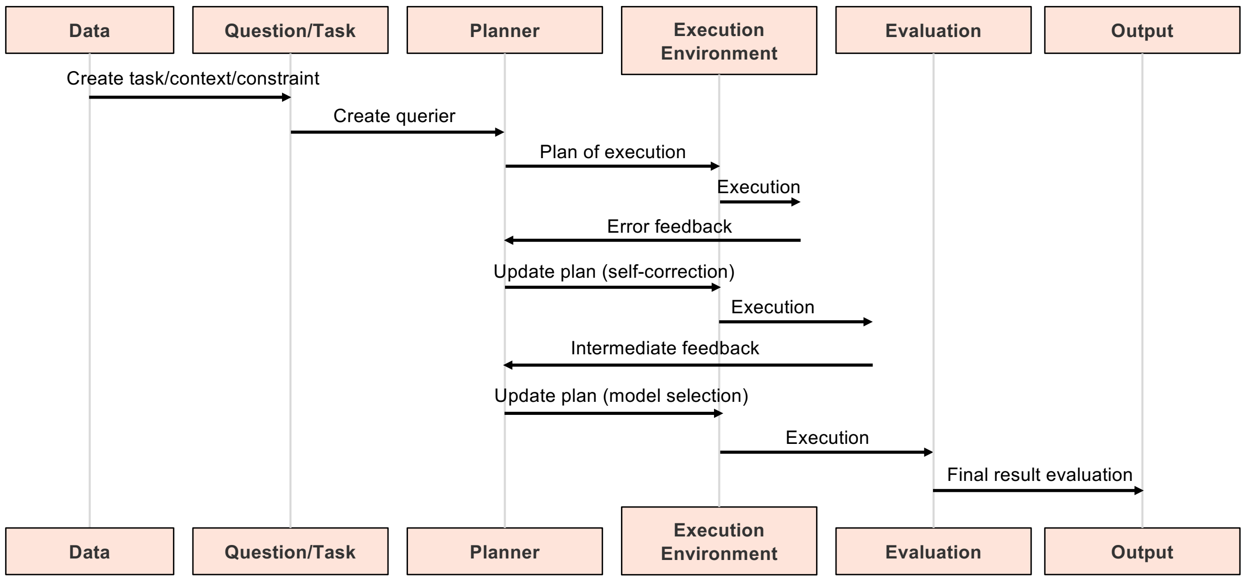
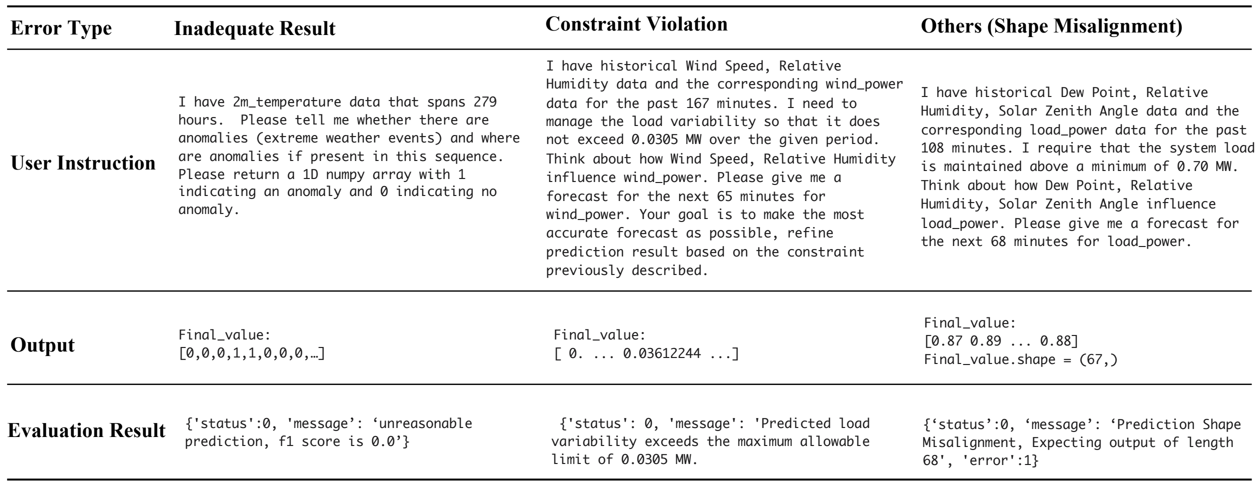


Figure 8: Illustration of an example TS-Reasoner workflow

B Prompt for TS-Reasoner

```
"""
def UniPreOP(data, future_length, model):
    Channel independent zero shot forecasting tool. For time series with
       covariates, this tool can be used by simply providing the main time series
       data.

    Input:

    - data: np.array([T,C]),T=history data len, C=variable number, the main time
       series

    - future_length: int, the number of steps to forecast

    - model: str, the model to use for prediction. available models:  "TEMPO", "
       ARIMA", "Chronos", "Moirai", "TabPFN", "Lagllama"

    Output:

    - np.array([N,C]), N=future\_length, C=variable number, the predicted future
       values

(more definitions of operators)

Example:

Question:
I have 2m_temperature data that spans 219 hours. Please tell me whether there are
   anomalies (extreme weather events) and where are anomalies if present in this
   sequence. Please return a 1D numpy array with 1 indicating an anomaly and 0
```



Error Type	Inadequate Result	Constraint Violation	Others (Shape Misalignment)
User Instruction	I have 2m_temperature data that spans 279 hours. Please tell me whether there are anomalies (extreme weather events) and where are anomalies if present in this sequence. Please return a 1D numpy array with 1 indicating an anomaly and 0 indicating no anomaly.	I have historical Wind Speed, Relative Humidity data and the corresponding wind_power data for the past 167 minutes. I need to manage the load variability so that it does not exceed 0.0305 MW over the given period. Think about how Wind Speed, Relative Humidity influence wind_power. Please give me a forecast for the next 65 minutes for wind_power. Your goal is to make the most accurate forecast as possible, refine prediction result based on the constraint previously described.	I have historical Dew Point, Relative Humidity, Solar Zenith Angle data and the corresponding load_power data for the past 108 minutes. I require that the system load is maintained above a minimum of 0.70 MW. Think about how Dew Point, Relative Humidity, Solar Zenith Angle influence load_power. Please give me a forecast for the next 68 minutes for load_power.
Output	Final_value: [0,0,0,1,1,0,0,0,…]	Final_value: [0. ... 0.03612244 ...]	Final_value: [0.87 0.89 ... 0.88] Final_value.shape = (67,)
Evaluation Result	{'status':0, 'message': ‘unreasonable prediction, f1 score is 0.0’}	{'status': 0, 'message': 'Predicted load variability exceeds the maximum allowable limit of 0.0305 MW.	{‘status’:0, ‘message’: ‘Prediction Shape Misalignment, Expecting output of length 68', 'error':1}

Figure 9: Example Result Errors

```
    indicating no anomaly. The data is stored in variable VAL and some anomaly-free
     normal samples are stored in variable NORM_VAL.
Program:
NORM_SCORE = AnomalDetOP(data=NORM_VAL)
THRES = calibrateThreshOP(data=NORM_SCORE)
TEST_SCORE = AnomalDetOP(data=VAL)
FINAL_RESULT = convertBinaryOP(data=TEST_SCORE, threshold=THRES)

(a few more examples)

Question:

I have 2m temperature data that spans 202 hours. Please tell me whether there are
    anomalies (extreme weather events) and where are anomalies if present in this
    sequence. Please return a 1D numpy array with 1 indicating an anomaly and 0
    indicating no anomaly. The data is stored in variable VAL and some anomaly-free
     normal samples are stored in variable NORM_VAL. Follow previous examples and
    answer my last question in the same format as previous examples within markdown
     format in ‘‘‘python‘‘‘.

Given the question and the toolbox definition provided above, please give steps of
     operations from the oprations availble listed above that can answer the
    question. Return only toolbox operations and enclose your response in ‘‘‘python
    ‘‘‘ markdown. Do not include any other information in your response. For *PreOP
     modules, you can recieve intermediate evaluation feedback to help you choose
    the best model. You should revert to a previously best performing model if all
    other models you tried fail to improve the performance (a lower MAPE value
    indicates better performance). You will be able to regenrate your response upon
     recieving feedback. Your goal is to achieve a status of 1 in the evaluation
    result when possible. For example, if model A gives lower MAPE value compared
    to model B, you should choose model A.You should replace {PLACEHOLDER} with the
     corresponding arguments you deem appropriate for the operation functions. DO
    NOT use any other irrelevant operations or custom python code, follow my
    examples where solutions should be in such format.
"""
```

Error Type	**Execution Error**	**Execution Error**
User Instruction	`I have historical Advertising Spend (A), Sales (B), Economic Factors (C), Customer Sentiment (D) data and want to get the causal relationship between each pair of the variables. I know that 41.66666666666667% of the variable pairs have relationship. Consider the potential influence of each variable on the others in this variable list: ['Advertising Spend (A)', 'Sales (B)', 'Economic Factors (C)', 'Customer Sentiment (D)']. Please provide 2d numpy matrix with binary values to indicate whether each pair of variables has a relationship.`	`I have load_power data for the past 135 minutes. I require that the system load is maintained above a minimum of 1490.6020757896545 MW. Please give me a forecast for the next 17 minutes for load_power. Think about what could be relevant covariates that can help forecast load_power. Your goal is to make the most accurate forecast as possible, refine prediction result based on the constraint previously described, and return the result as a 1D numpy array.`
Execution	`An error occurred: name 'n_variables' is not defined`	`An error occurred: can't multiply sequence by non-int of type 'float' final value.`
Evaluation Result	`{'status':0, 'message': "evaluator received input of NoneType"}`	`{'status':0, 'message':"An error occurred: 'Index' object has no attribute 'minute'"}`

Figure 10: Example Execution Errors.

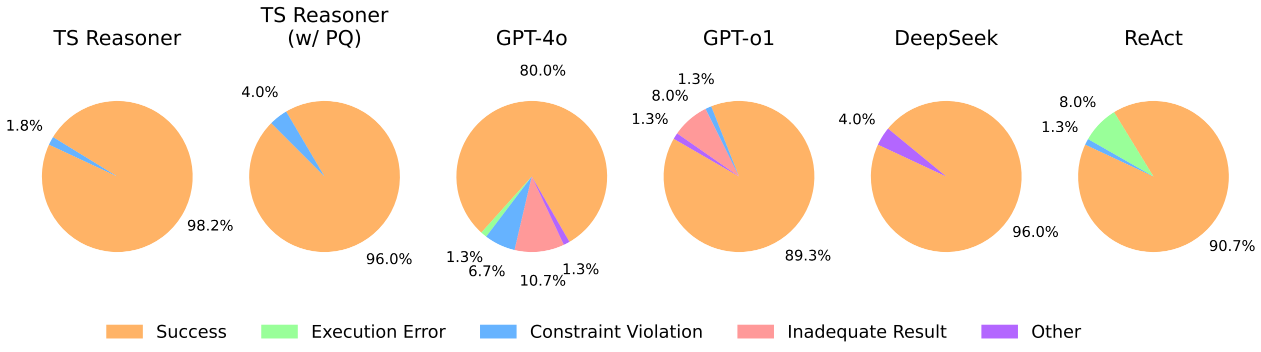


Figure 11: Error distribution of different approaches on electricity prediction task with covariates.

C Operator Description

Time Series Models operators
- UniPreOP: This operator is primarily used for univariate time series forecasting, leveraging advanced time series models to accurately predict future sequences. It currently supports Lag-Llama, Chronos, TimeGPT, TEMPO, Arima zero shot inference.

- MultiPreOP: This is a multivariate time series forecasting operator used to accurately predict the target variable based on multiple exogenous variables. It currently support TimeGPT zero shot inference and linear regression.

- AnomalDetOP: This is a time series anomaly detection operator used to generate anomaly scores based on reconstruction errors. It current support MOMENT foundation model zero shot inference.

- trainForecastOP: This is a operator that supports training time series forecasting models given input data. It currently supports iTransformer training. Model choice is based on current Time Series Library Leaderboard and more models can be included.

- trainADOP: This is a operator that supports training time series anomaly detection models given input data. It currently supports TimesNet training. Model choice is based on current Time Series Library Leaderboard and more models can be included.

Numerical operators
- CausalMatrixOP: This operator is used to calculate significance of granger causality relationship between each pair of variables in a time series dataset.

- ConcatOP: This operator concatenates the two given data horizontally.

- thToBinaryOP: This operator converts the input data into binary values based on threshold or percentile value.

- calibrateThreshOP: This operator approximates the input data with a normal distribution and returns the critical threshold value (3 standard deviations away from the mean) of this distribution.

- ApplyOP: This operator applies input function to the corresponding data. There are many more operators in this category that performs tasks as suggested by their names: checkStationaryOP (perform adf test), checkTrendStationaryOP (perform kpss test), getChptOP (perform bayesian changepoint detection), getTrendOP, compareDisOP (perform ks test), detectSpikesOP, getCyclePatternOP, getAmplitudeOP, getPeriodOP, testWhiteNoiseOP (perform box test), getNoiseCompOP, getAutoCorrOP, getMaxCorrLagOP, decomposeOP, getSlidingVarOP, getCyclePatternOP, detectFlippedOP, detectSpeedUpDownOP, detectCutoffOP, getTrendCoefOP, VolDetOP

Data retrieval operators
- getEnvDataOP: This operator can retrieve specified weather or air quality variables if available from openmeteo API. The input must specify a time range, geolocation, as well as resolution (daily or hourly).

-getElectricityDataOP: This operator retrieves electricity data given zone code from the official EIA website. The input must specify a grid zone name, time range, and variables to retrieve.

LLM generated custom operators
- RefGenOP: This operator is primarily used to generate a corresponding python function based on the requirement described in the prompt.

D Error Examples and Additional Error Analysis

This section illustrates an example execution workflow of TS-Reasoner (figure 8) and representative failure cases, including execution errors, shape mismatches, constraint violations, and inadequate predictions (figures 9 and 10). Examples are drawn from both TS-Reasoner and baseline models. Figures 11, 12, 13, 14, and 15 show the distribution of error types across various task settings. Execution errors in TS-Reasoner are minimal, reflecting the effectiveness of operator-based abstraction. However, introducing paraphrased

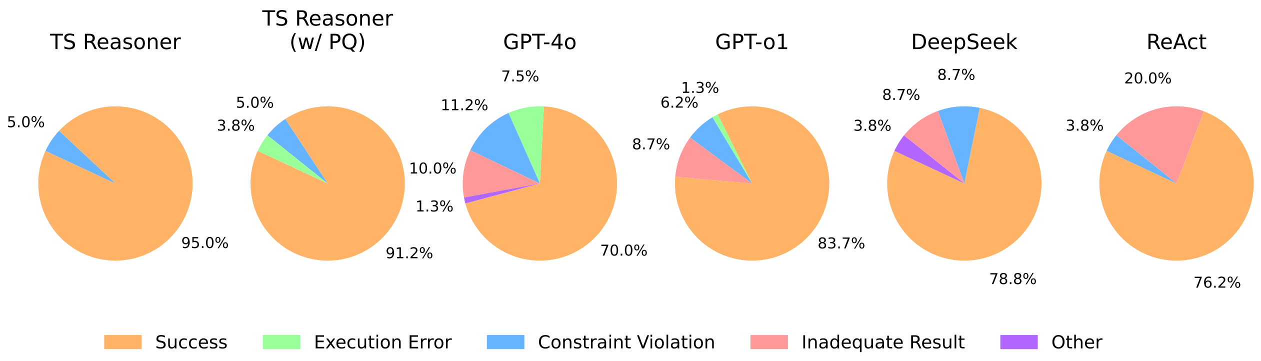


Figure 12: Error distribution of different approaches on electricity prediction task across multiple grids.

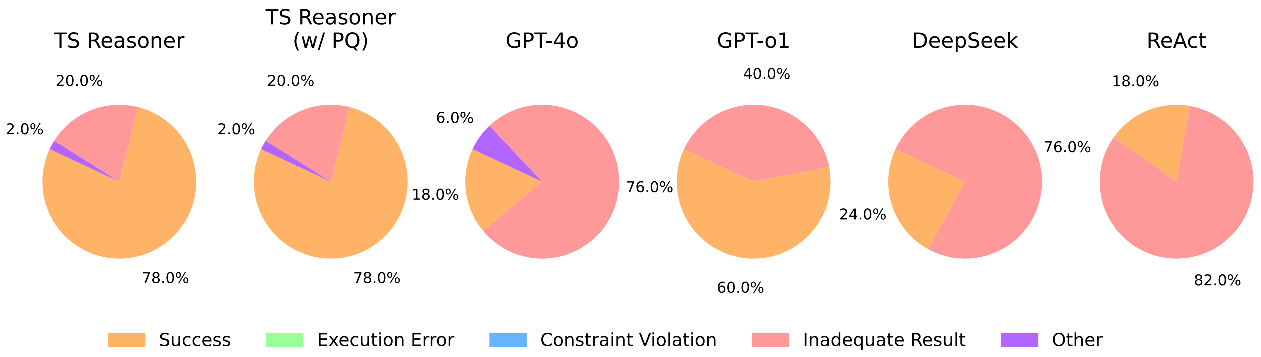


Figure 13: Error distribution of different approaches on extreme weather detection with anomaly free reference data.

questions occasionally triggers execution failures, suggesting sensitivity to linguistic variation. In extreme weather detection task (figure 13), inadequate result is the most dominant failure patterns, demonstrating an inability of baseline models to fully leverage anomaly free reference samples to calibrate anomaly threshold often producing a trivial all-zero labels.

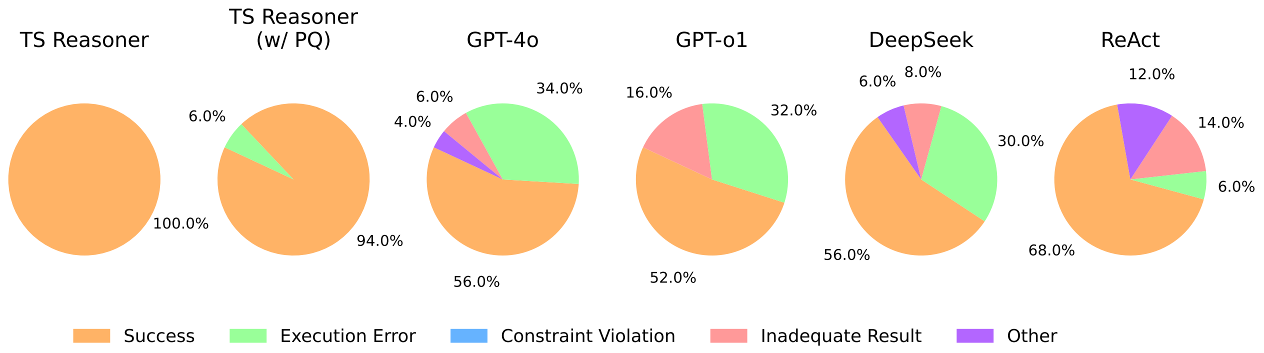


Figure 14: Error distribution of different approaches on extreme weather detection task with known anomaly rates.

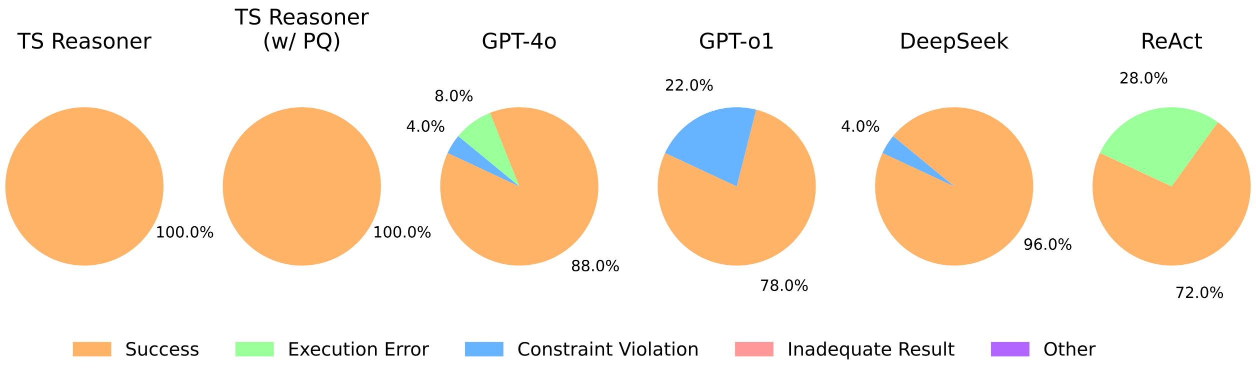


Figure 15: Error distribution of different approaches on causal discovery tasks.

Q. No.	Question	Ambiguity Level (Reasons)
1	Detecting anomalies in a 2-meter temperature time series dataset that spans 8.72 days (824 hours) and consists of 253 data points, I need to identify extreme weather events and determine their locations within the sequence. This requires labeling a binary numpy array where 1 indicates an anomaly and 0 indicates a normal value. The input data is stored in the variable "VAL" and the expected anomaly rate is to be stored in the variable "ANOMALY_RATE".	Low (Clear input and output, well-defined task, specific variable names)
2	Indicate the presence of extreme weather events within a 2m_temperature dataset spanning 260 hours. Using the provided VAL and NORM_VAL arrays, provide a one-dimensional numpy array where each entry corresponds to one hour, tagged with 1 for an identified anomaly and 0 for a typical observation.	Medium (Did not explain what data each variable contains)
3	Given a 2D array of temperature data with a duration of 241 hours, I'd like to determine and visualize the occurrence of anomalous events and their corresponding positions within the sequence. To achieve this, I'd appreciate it if you could provide an output array of the same length as the original data, where each element indicates whether the corresponding temperature reading is anomalous (1) or not (0), alongside some sample normal data for reference.	Medium (Did not define what is anomalous event but it should be referring to extreme weather events, no clear input variables)
4	Given a dataset of system load power over the past 123 minutes, we need to develop a 69-minute forecast that adheres to a maximum allowable system load of 694.4796 MW. The goal is to leverage relevant historical data and make an accurate load power prediction, considering constraints and return the forecast as a one-dimensional numpy array.	Medium (Unclear where data is stored in, did not define relevant historical data when it refers to historical load data)
5	Provide a 26-minute load power forecast for 17 distinct electric grid zones, constrained by a maximum allowable system load of 0.9872 MW per zone, based on historical data (1030 minutes) with 17 associated load_power values.	High (No input variable names, unclear output format)
6	What historical wind power data for 19 electric grid zones over 882 minutes can be utilized to create a 56-minute wind power forecast while ensuring each zone's system load remains above a minimum threshold of approximately 0.01 MW. The goal is to achieve the most accurate forecast possible, taking into account the specified constraint.	High (Unclear input variable names, the constraint number is approximated)
7	Given 70 minutes of historical load_power data in variable VAL, derive a 37-minute forecast for future load_power while ensuring that the predicted maximum system load does not exceed 1151.4102 MW. What relevant covariates should be considered for this forecast, and how can the accuracy of the prediction be refined given this constraint to produce a 1D numpy array result?	High (Multiple tasks mixed, formulated as questions rather than task instructions)

Table 5: Sample paraphrased questions classified based on ambiguity levels.