# BOOTSTRAPPED EXPLORATION WITH CAUSAL REASONING: A TRAINING PARADIGM FOR ADAPTIVE FORECASTING AGENT

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Time series forecasting is critical in domains such as finance, energy, and healthcare, yet real-world datasets often exhibit non-stationarity, noise, missing values, and distribution shifts, posing severe challenges for generalization. In practice, industry solutions typically rely on customized forecasting frameworks that combine imputation, decomposition, and specialized models. However, such frameworks incur high labor costs. Moreover, we observe that many frameworks suffer from the impacts of distribution shifts, which degrade their respective performance. Thus, it is critical to establish a new paradigm that retains high transferability across diverse datasets while accumulating reusable strategy knowledge. This is fundamental for large-scale and dynamic environments. While large language model-based agents have recently demonstrated strong reasoning and tool-use capabilities, no forecasting agent can automatically adapt to diverse time-series datasets. This gap arises from two core obstacles: the scarcity of labeled supervision and the inherent complexity of mapping dataset-specific meta-features to effective forecasting strategies. To address these challenges, we propose **BECRA**, a novel agent training paradigm that learns forecasting intelligence through contrast-aware exploration and causal lesson extraction, without any human-annotated supervision. BECRA distills symbolic strategy lessons that enable in-context planning on unseen datasets, achieving zero training adaptation. Source code is attached here [1].

## 1 INTRODUCTION

Time-series forecasting underpins a wide spectrum of real-world applications, including finance, energy, and healthcare (Torres et al., 2021). Improving predictive accuracy is crucial for decision-making, yet real world datasets often exhibit significant challenges. For example, data scarcity frequently occurs when the observation period is short, leaving models with insufficient samples to capture meaningful patterns; non-stationarity is manifested through structural breaks or regime shifts that obscure underlying regularities; and noise and anomalies, introduced by sensor failures, market manipulations, or missing values, can easily mislead forecasting models (Masini et al., 2023).

While recent deep learning models for forecasting are powerful, their effectiveness remains constrained by dataset specific meta-features (Lim & Zohren, 2021). Performance is highly conditional on the alignment between a model's design and the dataset specific meta-features (Benidis et al., 2022). For instance, Transformers (Vaswani et al., 2017) may underperform on strongly periodic data compared to Autoformer (Wu et al., 2021), while PatchTST (Nie et al., 2022) excels in univariate forecasting but may struggle in multivariate scenarios with strong intervariable dependencies. These examples highlight a fundamental limitation: each architecture is specialized for certain conditions, and no single model can generalize perfectly across all datasets (Zeng et al., 2023).

In practice, industrial applications often rely on customized frameworks that align methods like imputation and decomposition with dataset specific meta-features (Fatima & Rahimi, 2024). While this intuition is reasonable, the approach suffers from systemic limitations in large scale, dynamic environments. First, they are costly and inefficient, requiring extensive manual trial-and-error tuning

[1] https://figshare.com/s/a1527cca4d5ccf8d21e0

by domain experts (Trirat et al., 2024). Second, they lack transferability; these brittle, fine-tuned solutions do not generalize and must be rebuilt when data conditions shift (Gong et al.). Third, they fail to adapt to data evolution; as static frameworks, their performance degrades rapidly with distribution shifts, creating an impractical burden of constant re-tuning (Fan et al., 2024). Finally, they prevent knowledge accumulation, as the manual tuning process fails to distill reusable strategic insights, forcing organizations into a costly, repetitive exploration cycle for each new dataset.

These deficiencies point to a fundamental bottleneck: current practice struggles to support the demand for adaptive forecasting in complex and dynamic data environments, where systems fail to accumulate and reuse interpretable strategy knowledge and cannot achieve rapid transfer to new datasets. Recently, the rise of large language models (LLMs) has demonstrated that agents can autonomously solve complex tasks in open environments through reasoning and tool use (Guo et al., 2024). This motivates a natural question: can we build a forecasting agent that automatically learns how to select effective forecasting strategies based on dataset meta-features, thereby overcoming the inherent limitations of existing frameworks (Talagala et al., 2023)? However, training such a forecasting agent faces two central technical challenges. First, unlike NLP or vision domains, time series forecasting lacks large scale annotated corpora that map datasets to optimal pipelines, leaving agents without supervised signals for strategy learning (Das et al., 2024). Second, the relationship between dataset specific meta-features and effective forecasting strategies is highly complex and context-dependent, making it difficult to capture with handcrafted heuristics (Talagala et al., 2023). These challenges explain why no forecasting agent has yet emerged that can generalize across diverse datasets and dynamically evolving environments (Kong et al., 2025).

To address these challenges, we propose **BECRA**, a novel agent training paradigm designed to endow agents with adaptive forecasting intelligence. Unlike traditional frameworks that rely on expert heuristics and manual tuning, BECRA builds intelligence around a cycle of exploration–contrast–induction–application: it conducts structured exploration over the combinatorial space of toolchain configurations, then applies a contrastive induction operator to establish performance differences into causal strategy lessons, and encodes these lessons into an interpretable symbolic representation. This knowledge representation is not only reusable across different tasks and setting (such as zero-shot learning adaptation), but also acts as a constraint in subsequent optimization. BECRA can invoke and compose lessons through in-context planning without additional parameter updates, presenting unique capability for unprecedented sustainability in dynamic industry environment for time-series forecasting.

Our contributions can be summarized as follows:

- We identify the limitations of existing time series forecasting models and frameworks, highlighting the need for adaptive agents that can reason over diverse dataset meta-features.
- We propose BECRA, a novel agent training paradigm that constructs forecasting intelligence through contrast-aware exploration and causal lesson extraction, entirely without human-annotated supervision.
- We design a library of dataset meta-features and modular forecasting toolchains, enabling systematic exploration and interpretable causal reasoning.

## 2 Related work

**Time-Series Forecasting Models:** Classical models such as ARIMA (Box & Jenkins, 1976) struggle with non-linear dependencies; although deep learning improved forecasting, modern practice is dominated by diverse Transformer variants (e.g., Informer (Zhou et al., 2021b), Autoformer, PatchTST) whose performance is sensitive to dataset meta-features, so no single architecture is universally optimal. Recent time-series foundation models (e.g., Chronos (Ansari et al., 2024), Moirai (Woo et al., 2024)) target broader applicability, yet their gains can be unreliable under substantial distribution shifts, and they still remain single-architecture predictors rather than adaptive systems for heterogeneous real-world data. **LLM-Based Agents and Training Paradigms:** LLMs have enabled agents that reason and use tools (e.g., ReAct (Yao et al., 2023), AutoGPT (Yang et al., 2023), Voyager (Wang et al., 2023)), suggesting a path beyond static modeling toward adaptive decision making; however, agentic forecasting is underexplored due to the lack of large-scale supervision mapping datasets to optimal pipelines, and because effective strategies depend strongly on dataset-specific meta-features, limiting direct transfer of existing agent training paradigms. **Automated Machine Learning:** AutoML methods (e.g., Auto-sklearn (Feurer et al., 2022)) can achieve strong one-off performance but rely on expensive per-dataset search and retraining.

**LLM-Agent for Time-Series Forecasting:** Recent research has explored integrating LLM-based agents with time-series forecasting. One line of work incorporates external news to capture fluctuations (Wang et al., 2024b), while another leverages LLMs to analyze datasets for selecting auxiliary data that improves training-set quality (Yeh et al., 2025). TimeCopilot further addresses the fragmentation of the TSFM ecosystem via agent-based integration by unifying APIs of multiple TSFMs and recommending models according to dataset characteristics (Garza & Rosillo, 2025), while TimeSeries Scientist lets the LLM analyze dataset meta-features and then recommend an entire forecasting pipeline (Zhao et al., 2025). In contrast, BECRA is proposed as an agent training paradigm whose goal is to strengthen the agent's ability to select suitable forecasting frameworks for a given dataset. By repeatedly contrasting good and bad performances, BECRA distills and validates transferable causal lessons, thereby enhancing the agent's intelligence.

## 3 BECRA Framework

The BECRA framework, depicted in Fig. 1, implements a four-stage cycle to endow the agent with adaptive intelligence. It begins by exploring the strategy space to build a corpus of empirical evidence, from which it induces abstract causal lessons, verifies their robustness, and ultimately uses this knowledge for zero-shot planning on unseen datasets.

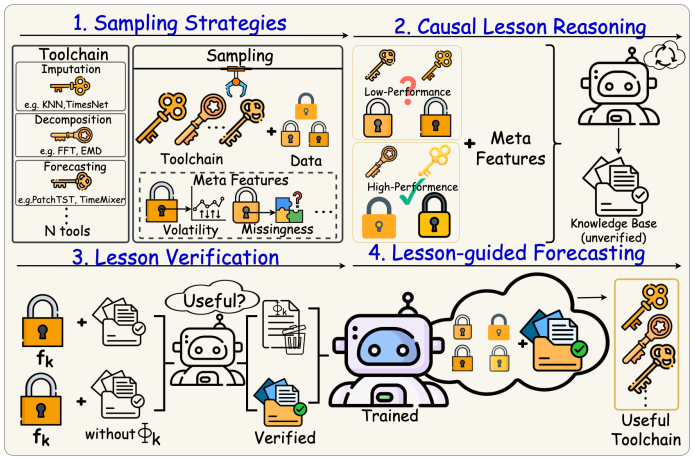

Figure 1: A high-level overview of the BECRA framework, illustrating the four-stage cycle.

### 3.1 Exploratory Construction of Forecasting Strategies

To efficiently explore the combinatorial space of time series processing tools and forecasting models, we propose a sampling strategy based on the Upper Confidence Bound (UCB) principle from bandit theory (Slivkins et al., 2019). Unlike conventional approaches that focus solely on discovering top-performing pipelines, our method, termed **Contrast-aware UCB Sampling**, explicitly preserves both *high-performing* and *low-performing* combinations for subsequent contrastive analysis and causality-aware lesson extraction.

Let $\mathcal{A}$ denote the set of candidate toolchains, where each $a \in \mathcal{A}$ represents a predefined sequence of components (e.g., imputation method, decomposition technique, forecasting model). After each evaluation on a dataset, we update the empirical performance of each toolchain $a$ using the following UCB objective: $a^* = \arg\max_{a \in \mathcal{A}} \mu(a) + \lambda \cdot \sigma(a)$, where: $\mu(a)$ is the average historical performance (e.g., inverted MSE) of toolchain $a$, $\sigma(a)$ is the standard deviation of its performance across datasets, reflecting uncertainty, $\lambda$ is the exploration-exploitation trade-off coefficient. This formulation promotes exploration of pipelines with either strong past performance or high variance, allowing for the discovery of robust and generalizable toolchains across diverse time-series characteristics.

We incorporate contrast-awareness by explicitly recording not only high-reward samples but also low-reward ones. Specifically, we actively retain pipeline-dataset pairs where the toolchain performs poorly, especially under dataset meta-features similar to those of successful cases. This facilitates later-stage causal contrastive reasoning to extract fine-grained lessons on why some pipelines fail while others succeed under similar conditions. The pseudocode of this part is in Appendix A.6.1.

### 3.2 Extracting Strategy Lessons via Contrastive Causal Reasoning

To generalize forecasting strategies across diverse time series datasets, it is essential to extract lessons that capture the causal conditions under which certain toolchains succeed or fail. Building on the contrast-aware sampling in the previous stage, we now seek to explain these behavioral differences by contrasting high-performing and low-performing outcomes under similar conditions. Rather than training a predictive model, we leverage an LLM-based agent to perform causal reasoning over sampled strategy-dataset pairs and to distill abstract strategy-level lessons.

Formally, let $\mathcal{D} = \{(x_i, a_i, y_i)\}_{i=1}^{N}$ denote the sampled set, where $x_i$ represents the meta-features of a dataset, $a_i$ denotes a forecasting strategy (i.e., a fixed toolchain), and $y_i$ is the observed performance. We partition this set into two subsets based on a predefined performance threshold $\tau$: $\mathcal{D}^+ = \{(x_i, a_i) \mid y_i \geq \tau\}, \quad \mathcal{D}^- = \{(x_j, a_j) \mid y_j < \tau\}$. For each strategy $a$, we form contrastive tuples as: $\mathcal{C}_a = \{(x_i, x_j) \mid (x_i, a) \in \mathcal{D}^+, (x_j, a) \in \mathcal{D}^-\}$. These tuples define contrastive behavioral shifts under identical strategies $a$ across different datasets. We then invoke the agent's reasoning module to induce a set of explicitly expressible, interpretable, and referable causal lessons $\mathcal{L} = \{\phi_1, \phi_2, \ldots, \phi_K\}$, where each $\phi_k$ is a structured explanatory statement. For example: $\phi_k$**:** *"Strategy A succeeds when the time series exhibits high volatility with frequent spikes (feature $f_k$). Because this implies the signal contains strong non-stationary jumps, which aligns with Strategy A's design that prioritizes local trend adaptation."*

Unlike conventional rule extraction via supervised learning (Rudin, 2019), we emphasize the agent's capability for language-based induction and semantic abstraction, aiming to derive interpretable and reusable causal rules (lessons). This enables generalization beyond training distributions and supports deliberate, explanation-grounded decision-making. The pseudocode of this part is in Appendix A.6.2.

### 3.3 Lesson Verification via Probabilistic Generalization

Although the lessons distilled in the previous stage offer interpretable insights, they remain heuristic. To transform them into reliable planning priors, we introduce a verification stage to rigorously test their generalization on a validation set. We treat each lesson $\phi_k$ as a symbolic hypothesis (e.g., "Strategy $A$ succeeds when feature pattern $f_k$ holds"). To evaluate it, we measure its causal effect, $\Delta_{\phi_k}$, which is the performance gain achieved by providing the lesson to the agent as a prior when its conditions $f_k$ are met: $\Delta_{\phi_k} = P(y = 1 \mid f_k, \phi_k) - P(y = 1 \mid f_k, \neg\phi_k)$, where $y = 1$ signifies a successful outcome. A high positive $\Delta_{\phi_k}$ provides strong evidence for the lesson's practical value. This effect is computed through controlled paired comparisons, where the agent performs two planning episodes under the same prompt, toolchains, and dataset conditions, with the only manipulating factor being the presence or absence of $\phi_k$.

To avoid false negatives and prevent the agent from being misled by over-conservative priors, this validation is bidirectional. We not only test lessons that predict success but also verify negative causal lessons (i.e., those that predict failure). For a "failure" lesson, we check if performance improves when the lesson is contradicted or removed, thus mitigating Type II errors. In this case, paired comparison is defined as the contrast between following negative lesson and explicitly contradicting it, thereby ensuring that harmful recommendations can be reliably detected. Let $\phi^-$ represent a negative lesson (e.g., "avoid strategy $S$"), and let Contradict($\phi^-$) denote the corresponding counterfactual intervention (e.g., "force the use of strategy $S$"). The causal effect is calculated as follows: $\Delta_{\phi^-} = P(y = 1 \mid f_k, \phi^-) - P(y = 1 \mid f_k, \text{Contradict}(\phi^-))$. The significantly positive $\Delta_{\phi^-}$ indicates that following $\phi^-$ leads to better performance than violating it, thereby mathematically validating the correctness of the the negative lesson. Ultimately, a lesson $\phi_k$ (or $\phi^-$) is retained in the knowledge base only if its estimated causal effect $\Delta_{\phi_k}$ (or $\Delta_{\phi^-}$) exceeds a predefined confidence threshold. Otherwise, it is marked as low-confidence and excluded from downstream planning. We define the success threshold $\tau$ in a relative, percentile-based manner: $y = 1$ if a strategy's performance ranks in the top $\alpha\%$ on the dataset (we use $\alpha = 30$), and $y = 0$ otherwise. This filtering process ensures that only empirically-grounded and generalizable lessons guide the agent's final decisions, enabling safe generalization and reducing the risk of overfitting. The pseudocode of our Lesson Verification is shown in Appendix A.6.3.

### 3.4 Forecasting with Lesson-Guided Planning

With the agent equipped with a library of symbolic lessons, we now shift from exploration to deployment. Instead of relying on fine-tuned models, we conduct forecasting through in-context learning (ICL) (Dong et al., 2022): each lesson acts as a symbolic prior, enabling the agent to adaptively plan strategies for new datasets via prompt-based inference. Given a new dataset characterized by meta-features $x_{\text{new}}$, the agent first queries the knowledge base (lesson memory) $\mathcal{L} = \{\phi_1, \dots, \phi_K\}$ to retrieve relevant lessons $\mathcal{L}_x \subset \mathcal{L}$ whose feature patterns match $x_{\text{new}}$. These lessons are then injected as structured prompts, guiding the agent's planning over forecasting toolchains without additional training. The final decision is thus informed by accumulated causal knowledge, rather than memorized statistical mappings.

We deliberately chose in-context learning over fine-tuning for three primary reasons: First, fine-tuning is ill-suited for the typical data landscape of time-series forecasting, where datasets are often too small or lack the diversity required for robust supervised optimization. Consequently, fine-tuning tends to capture shallow, dataset-specific correlations rather than generalizable causal principles, leading to fragile models (Yin et al., 2024). Our ICL approach, in contrast, uses symbolic induction to distill reusable knowledge, enabling strong transfer across diverse domains. Second, fine-tuning is computationally expensive and inflexible in dynamic environments (Mosbach et al., 2023). The need for costly retraining with every data distribution shift makes it impractical for real-world deployment. Our framework, by contrast, achieves zero training adaptation via lesson memory, enabling instantaneous strategy planning on new datasets. Finally, our ICL-based paradigm unlocks the full potential of state-of-the-art, closed-source LLMs. While these powerful models cannot be fine-tuned, their superior reasoning can be effectively guided by our distilled, symbolic lessons. This allows us to leverage the most powerful reasoning engines available without the need for retraining, ensuring efficient and scalable adaptation. The pseudocode of this part is in Appendix A.6.4.

### 3.5 Meta-Feature and Toolchain Design in BECRA

In BECRA, each dataset is characterized by a set of scalar meta-features—compact quantitative descriptors of its structure, variability, and complexity—which provide a principled interface for the agent's decision-making. Following community standards in time-series meta-learning (Lemke & Gabrys, 2010; Wei et al., 2025; Talagala et al., 2023; Zhao et al., 2025), we organize these features into functional groups that align with key modeling needs (e.g., Missingness for imputation, Periodicity for seasonal modeling, Stationarity for differencing), together with other characteristics such as volatility and trend structure. Importantly, meta-features in BECRA are *not* intended to reconstruct the raw time series; rather, their key requirement is *discriminative power*—supporting a stable mapping from dataset characteristics to effective toolchain components. As a result, there is a natural trade-off: an overly sparse meta-feature set may under-specify tool applicability boundaries and limit transferable lessons, while over-engineered or highly domain-specific descriptors may introduce noise and bias that reduce interpretability and cross-domain transfer. In practice, BECRA mitigates potential unreliability via the lesson verification stage (Section 3.3), which filters out low-confidence lessons that do not exhibit stable performance associations under the given meta-feature conditions. A comprehensive description of all meta-features is available in Appendix A.4.

BECRA operates on a modular six-stage forecasting pipeline (imputation, anomaly handling, transformation, decomposition, normalization, and forecasting), and its core function is to dynamically select only the necessary stages and the most suitable tools for each dataset. The tool library is curated with two guiding principles: *(i) diversity of inductive biases* and *(ii) modular coverage*. Diversity is essential because BECRA learns causal lessons through contrastive comparisons: a homogeneous library (e.g., only closely related neural forecasters) would collapse informative contrasts and reduce meaningful success/failure evidence for lesson induction. Modular coverage ensures the library spans the full pipeline so that different data pathologies described by meta-features have corresponding candidate tools. Concretely, the library includes representative choices across stages, such as KNN (Fix, 1985)/TimesNet for imputation, Isolation Forest for anomaly handling (Liu et al., 2008), Box–Cox for transformation (Sakia, 1992), FFT/EMD for decomposition (Yi et al., 2025), and forecasting models such as PatchTST. This curated set provides the heterogeneous primitives required for learning generalizable causal lessons and for robust zero-shot transfer across datasets. The complete library is detailed in Appendix A.5.

## 4 Experiments

### 4.1 Experimental Setup

Our experiments are conducted on a diverse set of widely-used benchmarks, including ETTh1, ETTh2, ETTm1, ETTm2 (Zhou et al., 2021a), Electricity (Lee et al., 2018), Weather (Godahewa et al., 2021), and the large-scale, highly heterogeneous M4 dataset (Makridakis et al., 2020). We evaluate BECRA against three categories of baselines: untrained agents powered by LLM (e.g., ChatGPT-4 (Achiam et al., 2023), Gemini (Team et al., 2023), and LLaMA (Touvron et al., 2023)), SOTA single-architecture models (TimeMixer (Wang et al., 2024a), TimesNet (Wu et al., 2022), PatchTST, iTransformer (Liu et al., 2023)), LLM-based models (e.g., TimeLLM (Jin et al., 2023) & GPT4TS (Zhou et al., 2023)) and pre-trained time-series foundation models (Chronos, Moirai). To ensure a fair comparison, we distinguish between the evaluation protocols. Our proposed BECRA agent, powered by ChatGPT-4, is evaluated in a zero-shot training manner: it leverages strategic lessons learned exclusively from other datasets to construct a forecasting framework for the target dataset, without training on it thereby ensuring the evaluation rigorously tests the agent's ability to generalize to a truly unseen dataset, rather than its ability to recall a known optimal solution. Similarly, the untrained LLM agent baselines are evaluated on their strategic capabilities; they are given the target dataset's meta-features and tasked with recommending a pipeline for prediction. In contrast, all other baselines (single-architecture and foundation models) follow the standard protocol of being trained or fine-tuned on the target dataset's training split. This design allows us to fairly assess both BECRA's strategic intelligence against other agents and its final forecasting accuracy against state-of-the-art forecasters. The prompt templates used are provided in Appendix A.7.

All experiments are conducted on a cluster with dual NVIDIA 5090 GPUs. For consistency, the sequence length is fixed at 96 for all experiments. We evaluate long-term forecasting on prediction lengths 96, 192, 336, 720 and short-term forecasting on the M4 dataset with input lengths [12, 96] and prediction lengths [6, 48], respectively. We use standard metrics like MSE, MAE, SMAPE, MASE, and OWA to measure performance. Detailed experimental information is in Appendix A.10.

### 4.2 Long-Term Forecasting Results

We conducted a comprehensive evaluation of BECRA's long-term forecasting performance across six benchmark datasets, with the averaged results presented in Table 1. Our evaluation on BECRA employs a strict leave-one-out protocol: for each target dataset, the agent's knowledge base is constructed from causal lessons extracted exclusively from the other datasets. This ensures the agent has zero prior knowledge of the target's specific strategy to performance mapping, testing its ability to generalize rather than recall. The results in Table 1 validate that BECRA can leverage its distilled symbolic lessons via in-context planning to select and compose a high-performing forecasting toolchain for entirely unseen datasets, achieving true **zero training adaptation**. Furthermore, the results confirm BECRA's **cross-domain transferability**. For instance, despite fundamental differences between datasets like Electricity and Weather, the agent consistently generated optimal forecasting toolchains for a target domain using lessons sourced from entirely different ones. This

| Models | **BECRA** | | ChatGPT-4 | | Gemini | | LLaMA | | TimeMixer | | TimesNet | | PatchTST | | iTransformer | | TimeLLM | | GPT4TS | | Chronos | | Moirai | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metric | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| ETTh1 | **0.441** | **0.436** | 0.472 | 0.479 | 0.452 | 0.446 | 0.566 | 0.520 | 0.467 | 0.450 | 0.479 | 0.477 | 0.451 | 0.441 | 0.464 | 0.455 | 0.460 | 0.490 | 0.447 | 0.436 | 0.588 | 0.466 | 0.48 | 0.439 |
| ETTh2 | **0.367** | **0.375** | 0.384 | 0.409 | 0.486 | 0.466 | 0.598 | 0.544 | 0.432 | 0.432 | 0.446 | 0.451 | 0.413 | 0.420 | 0.428 | 0.434 | 0.389 | 0.408 | 0.381 | 0.408 | 0.455 | 0.427 | 0.367 | 0.377 |
| ETTm1 | **0.380** | **0.389** | 0.768 | 0.597 | 0.875 | 0.640 | 1.288 | 0.703 | 0.3841 | 0.397 | 0.408 | 0.417 | 0.383 | 0.395 | 0.407 | 0.412 | 0.395 | 0.390 | 0.389 | 0.397 | 0.555 | 0.465 | 0.422 | 0.391 |
| ETTm2 | **0.274** | **0.321** | 1.058 | 0.720 | 1.267 | 0.803 | 1.245 | 0.795 | 0.278 | 0.325 | 0.295 | 0.332 | 0.285 | 0.327 | 0.291 | 0.334 | 0.281 | 0.321 | 0.285 | 0.331 | 0.295 | 0.338 | 0.329 | 0.343 |
| Electricity | **0.171** | **0.264** | 0.211 | 0.293 | 0.198 | 0.293 | 0.202 | 0.300 | 0.185 | 0.275 | 0.194 | 0.296 | 0.190 | 0.275 | 0.180 | 0.270 | 0.175 | 0.265 | 0.205 | 0.290 | 0.204 | 0.273 | 0.186 | 0.270 |
| Weather | **0.240** | **0.270** | 0.258 | 0.281 | 0.278 | 0.302 | 0.262 | 0.283 | 0.244 | 0.275 | 0.251 | 0.288 | 0.258 | 0.280 | 0.335 | 0.363 | 0.25 | 0.274 | 0.274 | 0.29 | 0.279 | 0.306 | 0.264 | 0.273 |

Table 1: Long-term forecasting results. All the results are averaged from 4 different prediction lengths, that is $\{96, 192, 336, 720\}$ while sequence length is fixed at 96.

demonstrates that by extracting abstract, domain-agnostic causal relationships between meta-features and strategies, BECRA shows the ability to transfer intelligence across domains.

Evaluation against three key baseline categories demonstrates the multifaceted advantages of the BECRA paradigm. First, its performance surpasses untrained LLM-powered agents (e.g., ChatGPT-4, Gemini, LLaMA), highlighting the necessity of the BECRA training paradigm. The results indicate that, while LLMs exhibit powerful general-purpose reasoning, such capabilities are not innately suited for time-series forecasting. A specialized, unsupervised process of distilling causal lessons, such as BECRA's, is essential to transform them into domain-expert planners. Second, BECRA's adaptive agent design consistently outperforms specialized single-architecture models (e.g., TimeMixer, TimesNet, PatchTST, iTransformer). Unlike these models, which are bound by a fixed inductive bias, BECRA learns a higher-order strategy of aligning the appropriate bias with dataset meta-features. Finally, we compare BECRA against emerging LLM-based (e.g., TimeLLM & GPT4TS) and pre-trained time-series foundation models (e.g., Chronos and Moirai). As shown in Table 1, BECRA demonstrates consistently superior performance. Its ability to outperform large-scale models, while relying on a relatively lightweight toolchain, suggests that strategic intelligence, which is the knowledge of how to decide, is more important than raw model capacity or the breath of available tools. This finding is further reinforced by our targeted experimental design. We deliberately excluded time-series foundation models from the BECRA's tool library. This was motivated by two primary considerations: (1) to prevent the agent from learning a 'shortcut' of simply relying on a powerful model's capacity instead of engaging in the fine-grained analysis of meta-features and algorithmic principles, which would compromise our research objectives; (2) to mitigate the risk of data leakage from foundation models that may have been exposed to evaluation datasets. Uncertainty estimation results are provided in Appendix A.9.

### 4.3 SHORT-TERM FORECASTING RESULTS

| Models | | BECRA | TimeLLM | GPT4TS | PatchTST | FEDformer | Autoformer | TimesNet | ChatGPT-4 | Gemini | LLaMA |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Yearly | SMAPE | **13.275** | 13.419 | 13.531 | 13.477 | 13.728 | 13.974 | 13.387 | 14.920 | 13.418 | 13.436 |
| | MASE | **2.984** | 3.005 | 3.015 | 3.019 | 3.048 | 3.134 | 2.996 | 3.364 | 3.045 | 3.043 |
| | OWA | **0.782** | 0.789 | 0.793 | 0.792 | 0.803 | 0.822 | 0.786 | 0.880 | 0.793 | 0.794 |
| Quarterly | SMAPE | **9.997** | 10.100 | 10.177 | 10.380 | 10.792 | 11.338 | 10.100 | 11.122 | 10.202 | 10.124 |
| | MASE | **1.166** | 1.178 | 1.194 | 1.233 | 1.283 | 1.365 | 1.182 | 1.360 | 1.194 | 1.169 |
| | OWA | **0.879** | 0.889 | 0.898 | 0.921 | 0.958 | 1.012 | 0.890 | 1.001 | 0.899 | 0.886 |
| Monthly | SMAPE | **12.633** | 12.980 | 12.894 | 12.959 | 14.260 | 13.958 | 12.679 | 15.626 | 12.791 | 12.677 |
| | MASE | **0.922** | 0.963 | 0.956 | 0.970 | 1.102 | 1.103 | 0.933 | 1.274 | 0.969 | 0.937 |
| | OWA | **0.871** | 0.903 | 0.897 | 0.905 | 1.012 | 1.002 | 0.878 | 1.141 | 0.899 | 0.880 |
| Others | SMAPE | **4.696** | 4.795 | 4.940 | 4.952 | 4.954 | 5.485 | 4.891 | 7.186 | 5.061 | 4.925 |
| | MASE | **3.189** | 3.178 | 3.228 | 3.347 | 3.264 | 3.865 | 3.302 | 4.677 | 3.216 | 3.391 |
| | OWA | **0.997** | 1.006 | 1.029 | 1.049 | 1.036 | 1.187 | 1.035 | 1.494 | 1.040 | 1.053 |

Table 2: Short-term forecasting results on M4 dataset. The input length and prediction length are set to $[12, 96]$ and $[6, 48]$, respectively.

We evaluated BECRA on the large-scale and highly heterogeneous M4 dataset for short-term forecasting, with the results presented in Table 2. The experiment shows that BECRA achieves best performance across all frequency categories (Yearly, Quarterly, Monthly, etc.), comprehensively outperforming the baseline models. This result provides evidence for the cross-task generalization capability of the strategic lessons distilled by BECRA. It demonstrates that the abstract causal knowledge learned from long-term forecasting tasks can be successfully transferred and adaptively applied to short-term problems across diverse frequencies and domains in a zero-shot manner.

### 4.4 ROBUSTNESS EVALUATION UNDER DATA MISSING SCENARIOS

To rigorously evaluate BECRA's robustness against the imperfect data common in real world settings, we conducted stress tests on the ETTh1 dataset. We introduced missing values using two patterns designed to simulate key data quality issues: (1) Random Missing, to represent sporadic events like transient sensor noise or network packet loss, and (2) Block Missing, to mimic more severe, systematic interruptions such as batch processing failures. We then escalated the challenge by progressively increasing the missing data ratio (15%, 25%, 35%) and extending the prediction horizon (from 96 to 720 steps) to verify BECRA's adaptive capabilities under severe conditions.

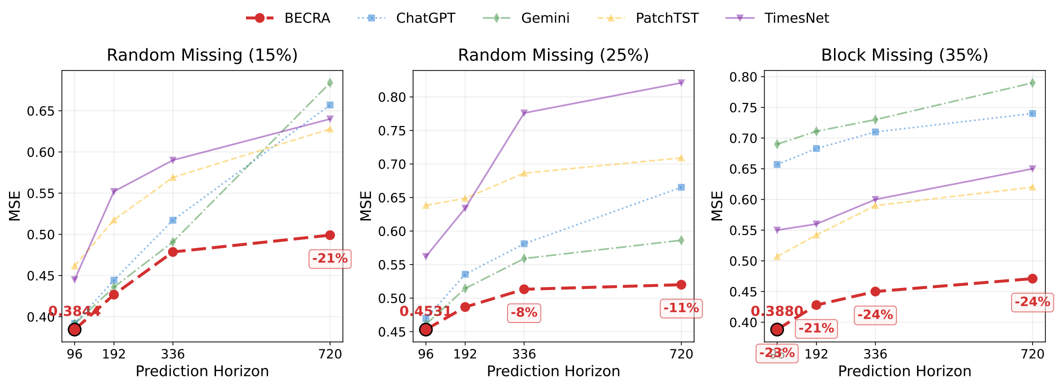

Figure 2: **BECRA Robustness Evaluation under Data Missing and Anomaly Scenarios on the ETTh1 dataset.** The plots show MSE as the prediction horizon increases, with results averaged over several severity levels. Left: Random Missing (averaged over 15%, 25%, 35% missing rates). Middle: Block Missing (averaged over 15%, 25%, 35% missing rates). Right: Anomaly Contamination (averaged over 2%, 5%, 10% rates).

The results of this stress test, shown in Fig. 2 (left and middle subplots), reveal two core conclusions. First, BECRA demonstrates better performance, with its MSE curve consistently lying below all baselines in both the Random and Block Missing scenarios. Second, and more importantly, it exhibits robust performance. This is visually evident from its substantially flatter performance curve, which shows that its accuracy degrades far more gracefully than the steeply rising error curves of the baselines as the prediction horizon extends. This confirms BECRA's stability under the dual pressures of long-range forecasting and incomplete data.

### 4.5 Robustness Evaluation under Anomaly Contamination Scenarios

To further evaluate BECRA's robustness, We conducted a data contamination stress test on BECRA to simulate this real-world problem by injecting random outliers into the ETTh1 dataset at varying ratios (2%, 5%, and 10%). This test aimed to verify if BECRA could not only diagnose the presence of anomalies but also dynamically adapt its strategy to mitigate their interference.

As depicted in the right subplot of Fig. 2, the averaged results of this test highlight two key takeaways. First, BECRA achieves a performance lead, consistently maintaining the lowest MSE curve on the contaminated data. Second, it exhibits stable robustness: its performance curve remains nearly horizontal, in sharp contrast to the clear upward trend of the baselines as the prediction horizon increases. This demonstrates BECRA's strong stability against the combined challenges of outliers and long-range forecasting.

## 5 Ablation Study

To deconstruct our paradigm and validate its core components, we conducted four targeted ablation studies. For a fair comparison with our main findings, these experiments used the identical datasets, toolchains, evaluation protocols, and prediction horizons as the long-term forecasting tasks.

First, to validate the necessity of our causal lessons, we ablate the entire lesson-guided planning module in **w/o Lessons**. This variant operates directly on the experience database constructed by our Contrast-aware UCB Sampling, which contains both high- and low-performing toolchain combinations. It therefore knows what strategies performed well or poorly on certain datasets, but lacks the abstracted symbolic lessons that explain why. As shown in Table 3, the performance of this variant collapses. This provides evidence that these symbolic lessons are the core of the agent's intelligence; the deep, causal understanding they provide is the foundational reason for our framework's ability to generalize effectively to new datasets.

| Dataset | BECRA | | Greedy | | w/o Lessons | | w/o Verification | | LLaMA (Finetuned) | | ChatGLM (Finetuned) | | Mistral (Finetuned) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| ETTh1 | **0.441** | **0.436** | 0.450 | 0.445 | 0.563 | 0.520 | 0.465 | 0.450 | 0.443 | 0.439 | <u>0.442</u> | 0.439 | 0.442 | <u>0.438</u> |
| ETTh2 | **0.367** | **0.375** | 0.412 | 0.420 | 0.587 | 0.534 | 0.432 | 0.431 | <u>0.368</u> | <u>0.397</u> | 0.395 | 0.401 | 0.369 | 0.399 |
| ETTm1 | **0.380** | **0.389** | 0.384 | 0.396 | 1.197 | 0.712 | 0.384 | 0.397 | 0.384 | 0.397 | 0.392 | 0.399 | <u>0.383</u> | <u>0.394</u> |
| ETTm2 | **0.274** | **0.321** | 0.284 | 0.326 | 1.034 | 0.722 | <u>0.277</u> | 0.326 | 0.281 | 0.327 | 0.285 | 0.324 | 0.276 | <u>0.322</u> |
| Electricity | **0.171** | **0.264** | 0.191 | 0.275 | 0.203 | 0.294 | <u>0.173</u> | <u>0.272</u> | 0.184 | 0.274 | 0.191 | 0.278 | 0.186 | 0.275 |
| Weather | **0.240** | **0.270** | 0.257 | 0.281 | 0.258 | 0.281 | <u>0.243</u> | <u>0.271</u> | 0.256 | 0.280 | 0.255 | 0.281 | 0.258 | 0.283 |

Table 3: Ablation study of BECRA components. **BECRA**: Full components. **Greedy**: Uses greedy sampling instead of contrast-aware UCB. **w/o Lessons**: Removes lesson-guided planning. **w/o Verification**: Skips the lesson verification stage. **Finetuned**: Agents using supervised fine-tuning instead of in-context learning. All the results are averaged from 4 different prediction lengths, that is $\{96, 192, 336, 720\}$ while sequence length is fixed at 96. The LLaMA-7B, ChatGLM-6B (GLM et al., 2024) and Mistral-7B (Jiang et al., 2023) baselines are based on relatively small scale models.

Second, having established the critical value of lessons, we investigate the optimal method for an agent to leverage them. This ablation validates our choice of using in-context learning (ICL), where lessons act as dynamic, zero-shot prompts, against the alternative of using them as training data for supervised fine-tuning (SFT). For the SFT variant, we fine-tuned three representative **LLMs** to map meta-features and lessons to the optimal toolchain. As shown in Table 3, our ICL-based agent outperforms the SFT counterparts. We attribute SFT's underperformance to two primary issues: 1) its tendency to overfit on limited training data by memorizing specific pairs instead of internalizing general principles, and 2) the risk of damaging the LLM's powerful pre-trained reasoning abilities. Our ICL approach avoids these pitfalls and offers higher computational efficiency (as it requires no costly retraining) and scalability (new lessons can be instantly integrated into the knowledge database). This makes ICL far better suited for the dynamic, continuously learning environments found in real-world applications.

Third, we test the importance of the Lesson Verification stage. In this ablation **w/o Verification**, the agent uses all raw lessons directly from the induction phase without any filtering. The results show a consistent, albeit slight, performance degradation. This drop is attributed to spurious or flawed lessons that act as noise and can mislead the agent's planning. The verification stage thus serves as a crucial quality-control filter, pruning these unreliable priors to ensure the agent's final knowledge base is robust, generalizable, and reliable.

Finally, we validate our Contrast-aware UCB Sampling strategy against a **Greedy** variant that collects only high-performing (positive) examples. The agent relying on lessons from this success-only data performs significantly worse. This is because a Greedy approach lacks the necessary contrast for deep causal reasoning. While it might learn shallow correlations (what works), it cannot understand the crucial context of why a strategy is effective. By analyzing both successes and failures, our contrast-aware method enables the distillation of deeper, causal lessons, confirming that contrast is a prerequisite for extracting profound and actionable knowledge.

## 6 Transferability and Efficiency

To evaluate the generalizability and model-agnostic nature of the BECRA training paradigm, we conducted a transferability analysis. Keeping the knowledge base (lesson memory) and prompt structure identical, we replaced the agent's original reasoning engine with a variety of state-of-the-art closed and open-source LLMs. These experiments used the identical datasets, toolchains, evaluation protocols, and prediction horizons as the long-term forecasting tasks.

The results in Table 4 show comparable performance across all tested LLMs. While the original agent performs best, other powerful models like Claude 3.5 show only a marginal and graceful degradation. This is an important finding: it demonstrates that the strategic guidance within the symbolic lessons distilled by the BECRA paradigm is portable and can be successfully understood by advanced LLMs. This successful decoupling of the "knowledge" from the "reasoner" is significant as it validates the generalizability of the BECRA itself. Ultimately, this confirms BECRA is a robust paradigm for creating portable forecasting intelligence, not just a model-specific solution.

In addition, BECRA can also significantly improve efficiency. Compared with representative AutoML baselines, BECRA has reduced trial costs by more than 80% through its transferable lesson memory and sampling parameters. The details of **computational cost analysis** is in Appendix A.8.

| **Dataset** | **BECRA** | | Claude 3.5 | | Gemini Pro | | Llama-3.1-70B | | Qwen-2.5-72B | |
|---|---|---|---|---|---|---|---|---|---|---|
| | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE | MSE | MAE |
| ETTh1 | **0.441** | **0.436** | 0.442 | 0.438 | 0.445 | 0.441 | 0.446 | 0.442 | 0.444 | 0.440 |
| ETTh2 | **0.367** | **0.375** | 0.369 | 0.397 | 0.372 | 0.400 | 0.375 | 0.402 | 0.370 | 0.397 |
| ETTm1 | **0.380** | **0.389** | 0.382 | 0.392 | 0.385 | 0.394 | 0.388 | 0.397 | 0.383 | 0.392 |
| ETTm2 | **0.274** | **0.321** | 0.276 | 0.323 | 0.279 | 0.325 | 0.282 | 0.328 | 0.277 | 0.323 |
| Electricity | **0.171** | **0.264** | 0.173 | 0.272 | 0.176 | 0.274 | 0.182 | 0.278 | 0.174 | 0.272 |
| Weather | 0.240 | 0.270 | **0.239** | **0.270** | 0.243 | 0.273 | 0.254 | 0.281 | 0.247 | 0.276 |

Table 4: Transferability analysis of the BECRA paradigm across different LLMs. All the results are averaged from 4 different prediction lengths, that is $\{96, 192, 336, 720\}$ while sequence length is fixed at 96. LLM baselines are accessed via API: Claude 3.5 (Anthropic, 2024), Gemini Pro (Team et al., 2023), Qwen-2.5-72B (Yang et al., 2025), and LLaMA-3.1-70B (Grattafiori et al., 2024).

## 7 Discussion: Scope and Future Work

**Potential lesson conflicts under segmented meta-features.** BECRA relies on causal lessons that map dataset meta-features to recommended toolchain components. In this work, meta-features are computed once at the dataset level, yielding a single global characterization; thus, when meta-features are internally consistent, the retrieved lessons typically do not conflict. A natural extension is to compute *segmented* meta-features for long or non-stationary series, where different temporal segments may exhibit different characteristics. In this setting, lessons activated by different segments may provide inconsistent guidance, making explicit conflict handling and aggregation of retrieved lessons an important direction for future work.

**Extending BECRA to orchestrate time-series foundation models (TSFMs).** In this work, for fairness and methodological clarity, we do not include TSFMs as selectable tools in BECRA's toolchain space. A natural future direction is to extend BECRA to orchestrate TSFMs as a dedicated model hub (in the spirit of systems like TimeCopilot (Garza & Rosillo, 2025)), and systematically evaluate different TSFMs under non-leaking protocols on benchmarks such as GIFT-eval (Aksu et al., 2024). This would enable deeper analysis of TSFM behaviors, including their relative strengths across domains, frequencies, and horizons, and how agentic pipeline reasoning can complement model capacity.

**Human-in-the-loop extension for high-stakes deployment.** While BECRA includes an automatic Lesson Verification stage to filter unreliable or spurious lessons, in high-stakes settings (e.g., safety-critical or high-cost decision making) it can be beneficial to introduce an optional human review step before a lesson is committed to the knowledge base. This provides an additional safeguard against rare failure modes such as hallucinated rationales or brittle causal associations, at the cost of reduced automation and increased latency. Exploring systematic criteria and workflows for such human-in-the-loop vetting (e.g., reviewing only borderline-confidence lessons) is a practical direction for improving reliability while managing the efficiency–accuracy trade-off.

## 8 Conclusion

We proposed BECRA, the first agent training paradigm for time series forecasting tasks, which combines contrast-aware UCB sampling with lesson-guided planning to construct reusable lesson memory. This paradigm addresses two challenges in the time series field: the lack of large benchmark datasets and the difficulty of aligning heterogeneous data characteristics with appropriate forecasting tools across domains. Experiments show that BECRA achieves strong forecasting performance, robustness and efficiency, while also realizing portability across large language models. In addition to time series, the exploration, contrast and induction of BECRA provides a universal paradigm for knowledge acquisition. This approach offers a new path in fields where datasets are limited and it is difficult to associate data with the right tools, such as materials science or drug discovery. Instead of expensive, repetitive searches for every new task, BECRA enables a shift toward reusable strategic reasoning, transform expert domain knowledge into practical and scalable frameworks.

## REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Gift-eval: A benchmark for general time series forecasting model evaluation, 2024. URL `https://arxiv.org/abs/2410.10393`.

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.

Anthropic. Introducing claude 3.5 sonnet. `https://www.anthropic.com/news/claude-3-5-sonnet`, 2024.

Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Yuyang Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, et al. Deep learning for time series forecasting: Tutorial and literature survey. *ACM Computing Surveys*, 55(6):1–36, 2022.

George Box and GM Jenkins. Analysis: Forecasting and control. *San francisco*, 1976.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*, 2024.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

Wei Fan, Shun Zheng, Pengyang Wang, Rui Xie, Jiang Bian, and Yanjie Fu. Addressing distribution shift in time series forecasting with instance normalization flows. *arXiv preprint arXiv:2401.16777*, 2024.

Syeda Sitara Wishal Fatima and Afshin Rahimi. A review of time-series forecasting algorithms for industrial manufacturing systems. *Machines*, 12(6):380, 2024.

Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. Auto-sklearn 2.0: Hands-free automl via meta-learning. *Journal of Machine Learning Research*, 23 (261):1–61, 2022.

Evelyn Fix. *Discriminatory analysis: nonparametric discrimination, consistency properties*, volume 1. USAF school of Aviation Medicine, 1985.

Azul Garza and Renée Rosillo. Timecopilot, 2025. URL `https://arxiv.org/abs/2509.00616`.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024.

Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I Webb, Rob J Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. *arXiv preprint arXiv:2105.06643*, 2021.

Peiliang Gong, Emadeldeen Eldele, Min Wu, Zhenghua Chen, Xiaoli Li, and Daoqiang Zhang. Towards adaptive time series foundation models against distribution shift.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL `https://arxiv.org/abs/2310.06825`.

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.

Xiangjie Kong, Zhenghao Chen, Weiyao Liu, Kaili Ning, Lechao Zhang, Syauqie Muhammad Marier, Yichen Liu, Yuhao Chen, and Feng Xia. Deep learning for time series forecasting: a survey. *International Journal of Machine Learning and Cybernetics*, pp. 1–34, 2025.

YW Lee, KG Tay, and YY Choy. Forecasting electricity consumption using time series model. *International Journal of Engineering & Technology*, 7(4.30):218–223, 2018.

Christiane Lemke and Bogdan Gabrys. Meta-learning for time series forecasting and forecast combination. *Neurocomputing*, 73(10-12):2006–2016, 2010.

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

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pp. 413–422. IEEE, 2008.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*, 2023.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2019.04.014. URL `https://www.sciencedirect.com/science/article/pii/S0169207019301128`. M4 Competition.

Ricardo P Masini, Marcelo C Medeiros, and Eduardo F Mendes. Machine learning advances for time series forecasting. *Journal of economic surveys*, 37(1):76–111, 2023.

Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, and Yanai Elazar. Few-shot fine-tuning vs. in-context learning: A fair comparison and evaluation. *arXiv preprint arXiv:2305.16938*, 2023.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

Remi M Sakia. The box-cox transformation technique: a review. *Journal of the Royal Statistical Society Series D: The Statistician*, 41(2):169–178, 1992.

Aleksandrs Slivkins et al. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2):1–286, 2019.

Thiyanga S Talagala, Rob J Hyndman, and George Athanasopoulos. Meta-learning how to forecast time series. *Journal of Forecasting*, 42(6):1476–1501, 2023.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

José F Torres, Dalil Hadjout, Abderrazak Sebaa, Francisco Martínez-Álvarez, and Alicia Troncoso. Deep learning for time series forecasting: a survey. *Big data*, 9(1):3–21, 2021.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Patara Trirat, Yooju Shin, Junhyeok Kang, Youngeun Nam, Jihye Na, Minyoung Bae, Joeun Kim, Byunghyun Kim, and Jae-Gil Lee. Universal time-series representation learning: A survey. *arXiv preprint arXiv:2401.03717*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and Jun Zhou. Timemixer: Decomposable multiscale mixing for time series forecasting. *arXiv preprint arXiv:2405.14616*, 2024a.

Xinlei Wang, Maike Feng, Jing Qiu, Jinjin Gu, and Junhua Zhao. From news to forecast: Integrating event analysis in llm-based time series forecasting with reflection, 2024b. URL https://arxiv.org/abs/2409.17515.

Wang Wei, Tiankai Yang, Hongjie Chen, Ryan A Rossi, Yue Zhao, Franck Dernoncourt, and Hoda Eldardiry. Efficient model selection for time series forecasting via llms. *arXiv preprint arXiv:2504.02119*, 2025.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. 2024.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2022.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Hui Yang, Sifu Yue, and Yunzhong He. Auto-gpt for online decision making: Benchmarks and additional opinions. *arXiv preprint arXiv:2306.02224*, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Chin-Chia Michael Yeh, Vivian Lai, Uday Singh Saini, Xiran Fan, Yujie Fan, Junpeng Wang, Xin Dai, and Yan Zheng. Empowering time series forecasting with llm-agents, 2025. URL https://arxiv.org/abs/2508.04231.

Kun Yi, Qi Zhang, Wei Fan, Longbing Cao, Shoujin Wang, Hui He, Guodong Long, Liang Hu, Qingsong Wen, and Hui Xiong. A survey on deep learning based time series analysis with frequency transformation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 6206–6215, 2025.

Qingyu Yin, Xuzheng He, Luoao Deng, Chak Tou Leong, Fan Wang, Yanzhao Yan, Xiaoyu Shen, and Qiang Zhang. Deeper insights without updates: The power of in-context learning over fine-tuning. *arXiv preprint arXiv:2410.04691*, 2024.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11121–11128, 2023.

Haokun Zhao, Xiang Zhang, Jiaqi Wei, Yiwei Xu, Yuting He, Siqi Sun, and Chenyu You. Time-seriesscientist: A general-purpose ai agent for time series analysis, 2025. URL `https://arxiv.org/abs/2510.01538`.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, volume 35, pp. 11106–11115. AAAI Press, 2021a.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021b.

T. Zhou, P. Niu, X. Wang, L. Sun, and R. Jin. One fits all: Power general time series analysis by pretrained LM. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

## A Appendix

### A.1 LLM Usage Claim

During the writing process, we solely used a large language model as an auxiliary grammar tool. Its main function was to help us polish and proofread the text, ensuring accuracy in grammar, punctuation, and spelling. All core ideas, content creation, and logical arguments were developed independently by human effort. The LLM did not participate in any creative work, nor did it provide any original content or concepts throughout this process.

### A.2 Ethcis Statement

In full compliance with the ICLR Code of Ethics, this study did not involve any human or animal experimentation. All datasets were sourced and used according to relevant guidelines to protect privacy. We have taken meticulous care to ensure the research process is free from bias and discriminatory outcomes. No personally identifiable information was used, nor were any experiments conducted that could pose privacy or security concerns. Our commitment to transparency and integrity is a cornerstone of this research.

### A.3 REPRODUCIBILITY STATEMENT

We have made every effort to ensure that the results presented in this paper are reproducible. Code and datasets have been made publicly available in an anonymous repository to facilitate replication and verification

### A.4 Meta-Feature Descriptions

This appendix provides a detailed description of the meta-features used by the BECRA agent to characterize time-series datasets. For each feature, we describe its significance for forecasting strategy selection and the specific metrics used for its quantification.

#### A.4.1 Basic Information

This group of features captures the fundamental scale and dimensionality of the dataset.

**Series Length** The total number of observations (time steps) in the series. This metric informs the agent about the volume of historical data available. Very short series may be insufficient for training complex models, making simpler models or transfer learning more appropriate.

**Number of Variates** The number of parallel time series variables in the dataset. A value of 1 indicates a univariate task, while a value greater than 1 indicates a multivariate task. This is a critical feature for deciding whether to use models that can capture cross-channel dependencies (e.g., TimesNet) or channel-independent models (e.g., PatchTST).

### A.4.2 MISSINGNESS

This feature quantifies the extent of missing data, which is a common problem in real-world time series.

**Missing Rate** This metric measures the proportion of missing ('NaN') values in the dataset. It is calculated as:

$$\text{Missing Rate} = \frac{\text{Count of NaN values}}{\text{Total number of observations}}$$

A high missing rate signals that an imputation strategy is essential. The agent uses this value to decide whether to apply imputation and to select an appropriate method (e.g., simple interpolation for low rates vs. a sophisticated model like TimesNet for high rates).

### A.4.3 VOLATILITY

Volatility features measure the degree of fluctuation or dispersion in the series.

**Variance ($\sigma^2$)** Variance measures the average squared deviation of each observation from the series mean ($\mu$), indicating the overall magnitude of fluctuations. It is calculated as:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

where $N$ is the series length and $x_i$ is the value at time $i$.

**Standard Deviation ($\sigma$)** As the square root of the variance, the standard deviation is expressed in the same units as the data, making it more interpretable.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

High variance or standard deviation suggests a highly fluctuating series, which may benefit from smoothing, decomposition, or models designed to handle volatility.

**Coefficient of Variation (CV)** The CV measures relative volatility by normalizing the standard deviation by the mean.

$$\text{CV} = \frac{\sigma}{\mu}$$

This is particularly useful for comparing the volatility of series with different average values. A high CV indicates significant volatility relative to the series' scale.

### A.4.4 TREND STRUCTURE

These features assess the presence and nature of trends in the data.

**Linear and Non-linear Trend Scores** We assess trend by fitting regression models to the time series, with time as the independent variable. The "score" is the coefficient of determination ($R^2$), which measures the proportion of the variance in the series that is predictable from the time variable. For a linear trend, the model is $y_t = \beta_0 + \beta_1 t + \epsilon_t$. For a non-linear trend, a polynomial model such as $y_t = \beta_0 + \beta_1 t + \beta_2 t^2 + \epsilon_t$ can be used. The $R^2$ is calculated as:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

where $y_i$ is the observed value, $\hat{y}_i$ is the value predicted by the regression model, and $\bar{y}$ is the mean of the series. A high $R^2$ value indicates a strong trend, guiding the agent to include trend modeling or detrending components in the toolchain.

### A.4.5 Periodicity

Periodicity features are crucial for identifying recurring cyclical patterns (seasonality).

**Autocorrelation Function (ACF)** The ACF measures the correlation of the time series with a lagged version of itself. The value at lag $k$ is calculated as:

$$\text{ACF}(k) = \frac{\sum_{t=k+1}^{N}(x_t - \mu)(x_{t-k} - \mu)}{\sum_{t=1}^{N}(x_t - \mu)^2}$$

Strong, significant peaks in the ACF plot at regular intervals are a clear sign of seasonality. The agent uses the magnitude of these peaks to gauge the strength of periodicity.

**Partial Autocorrelation Function (PACF)** The PACF measures the correlation between an observation at time $t$ and an observation at time $t - k$, after removing the linear effects of the intermediate lags $(t - 1, t - 2, \ldots, t - k + 1)$. It helps to identify the direct relationship between observations. Significant PACF values help in determining the order of autoregressive models.

**Fourier Spectral Peaks** By applying a Fast Fourier Transform (FFT) to the series, we can decompose it into a sum of sine and cosine waves of different frequencies. The resulting power spectrum shows the magnitude of each frequency. Prominent peaks in the spectrum correspond to the dominant periodicities in the data. The agent uses the location and height of these peaks to confirm seasonality and its period, guiding the choice of frequency-based models (e.g., FEDformer) or seasonal decomposition.

### A.4.6 Regime Shift

This feature identifies abrupt and fundamental changes in the statistical properties of the series.

**Change Points** A change point is a time index where the statistical properties of the series (e.g., mean, variance, trend) change significantly. We identify these points using statistical algorithms like the Pruned Exact Linear Time (PELT) method. The metric is the total count of significant change points detected. A high number of change points suggests structural breaks, which may warrant the use of segment-wise modeling or adaptive forecasting methods.

### A.4.7 Stationarity

Stationarity implies that the statistical properties of a series do not change over time. Many forecasting models assume stationarity.

**Augmented Dickey-Fuller (ADF) Test** The ADF test is a statistical test for a unit root in a time series sample. Its null hypothesis is that a unit root is present (the series is non-stationary). The test is based on the regression model:

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + \cdots + \delta_{p-1} \Delta y_{t-p+1} + \epsilon_t$$

The test statistic is used to compute a p-value for the coefficient $\gamma$. A low p-value (e.g., $< 0.05$) leads to the rejection of the null hypothesis, suggesting the series is stationary.

**KPSS Test** The Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test has a null hypothesis that the series is stationary around a deterministic trend. It is often used as a confirmatory test alongside ADF. A low p-value (e.g., $< 0.05$) indicates that the null hypothesis is rejected and the series is non-stationary. The agent uses the results of both tests to make a robust decision on whether differencing or other transformations are required.

### A.4.8 Distributional Shape

These features describe the shape of the data's probability distribution, providing insights into non-Gaussian properties.

**Skewness** Skewness measures the asymmetry of the distribution. A value of 0 indicates a perfectly symmetric distribution. The sample skewness is calculated as:

$$g_1 = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^3}{(\sigma^2)^{3/2}}$$

Significant skewness may violate the assumptions of some models and suggests that a data transformation (e.g., a log or Box-Cox transform) could be beneficial.

**Kurtosis** Kurtosis measures the "tailedness" of the distribution compared to a normal distribution. Excess kurtosis (kurtosis minus 3) is often used, where a value greater than 0 indicates heavy tails and a higher likelihood of outliers. The sample excess kurtosis is:

$$g_2 = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^4}{(\sigma^2)^2} - 3$$

High kurtosis can guide the agent to choose models or loss functions that are more robust to outliers.

### A.4.9 Noise Characteristics

These features quantify the level of random, unpredictable noise in the series.

**Signal-to-Noise Ratio (SNR)** The SNR compares the level of the desired signal to the level of background noise. It is often expressed in decibels (dB):

$$\text{SNR}_{\text{dB}} = 10\log_{10}\left(\frac{P_{\text{signal}}}{P_{\text{noise}}}\right) = 20\log_{10}\left(\frac{\sigma_{\text{signal}}}{\sigma_{\text{noise}}}\right)$$

where $P$ is power and $\sigma$ is the standard deviation. The signal and noise can be separated using filters or smoothing (where the smoothed series is the signal and the residual is the noise). A low SNR indicates a noisy series where denoising could be a valuable preprocessing step.

**Entropy** Shannon entropy measures the amount of uncertainty or randomness in the data. For a time series, we can bin the values to create a probability distribution and calculate entropy as:

$$H(X) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i)$$

where $p(x_i)$ is the probability of the series taking a value in the $i$-th bin. A high entropy value suggests the series is highly unpredictable and may be difficult to forecast accurately.

## A.5 Comprehensive Tool Library

This appendix provides a comprehensive description of the tools available to the BECRA agent for each stage of the time-series processing pipeline. The diversity of these tools enables the agent to compose a wide variety of strategies tailored to different dataset characteristics.

### 1. Imputation

- **Mean / Median / Mode**: Simple statistical imputation, fast but may distort variance.

- **Forward / Backward Fill**: Propagates the last or next valid observation, useful for data with temporal continuity.

- **Linear Interpolation**: Fills missing values using a linear function between known points.

- **K-Nearest Neighbors (KNN)**: Imputes missing values based on the average of the $k$ nearest neighbors.

- **MICE**: Multivariate Imputation by Chained Equations, an advanced statistical method for complex patterns.

- **SAITS**: Self-Attention based Imputation for Time Series, a state-of-the-art deep learning approach.

- **TimesNet**: Captures intra-period and inter-period variations with strong performance on imputation tasks.

### 2. Anomaly Handling

- **Z-Score / IQR**: Statistical methods based on standard deviation or interquartile range to detect outliers.

- **Isolation Forest**: An efficient tree-based unsupervised algorithm for anomaly detection.

- **Local Outlier Factor (LOF)**: A density-based algorithm effective at detecting local anomalies.

- **One-Class SVM**: A semi-supervised method that learns a decision boundary around normal data points.

- **Autoencoder**: A neural network that reconstructs normal data; high reconstruction error indicates anomalies.

### 3. Transformation

- **Log Transformation**: Stabilizes variance, often used for data with exponential growth.

- **Box-Cox Transformation**: A powerful transformation to correct for skewness and non-normality, requires positive data.

- **Yeo-Johnson Transformation**: An extension of Box-Cox that also supports zero and negative values.

- **Differencing**: First-order or seasonal differencing to remove trends or seasonality, making a series stationary.

### 4. Decomposition

- **Classical Decomposition**: Decomposes a series into trend, seasonal, and residual components using moving averages.

- **STL (LOESS)**: Seasonal-Trend decomposition using Loess, more robust and versatile than classical decomposition.

- **Fast Fourier Transform (FFT)**: Decomposes a signal into its constituent frequencies, ideal for strongly periodic series.

- **Wavelet Transform**: Provides a time-frequency representation, well-suited for non-stationary signals.

- **EMD / CEEMDAN**: Empirical Mode Decomposition and its variants, data-driven methods for nonlinear, non-stationary signals.

### 5. Normalization

- **StandardScaler**: Standardizes features to zero mean and unit variance, the default choice for many models.

- **MinMaxScaler**: Scales features to a specified range, typically $[0, 1]$.

- **MaxAbsScaler**: Scales each feature by its maximum absolute value, useful for sparse data.

- **RobustScaler**: Scales features using statistics robust to outliers (e.g., quantiles).

6. FORECASTING

- **ARIMA / SARIMA**: Classical statistical models for stationary series.

- **ETS (Exponential Smoothing)**: A family of forecasting methods based on weighted averages of past observations.

- **LightGBM / XGBoost**: Gradient boosting models, powerful when combined with feature engineering (e.g., lag features).

- **LSTM / GRU**: Recurrent Neural Networks for sequential data and long-term dependencies.

- **TCN (Temporal Convolutional Network)**: Uses convolutions for sequence modeling, often faster than RNNs.

- **Informer / Autoformer**: Efficient Transformer-based models for long-sequence forecasting.

- **PatchTST / iTransformer**: State-of-the-art Transformer models using channel-independent patching or inverted representations.

- **N-BEATS / N-HiTS**: MLP-based models with deep stacks of blocks, strong for univariate forecasting.

- **DLinear / TiDE**: Simple but powerful linear and MLP-based baselines.

- **TimesNet / TimeMixer**: Advanced state-of-the-art time-series models designed to capture diverse temporal patterns.

## A.6 PSEUDOCODE

### A.6.1 EXPLORATORY CONSTRUCTION OF FORECASTING STRATEGIES

The pseudocode of our Contrast-aware UCB Sampling for Exploratory Construction is shown as follows:

**Algorithm 1** Contrast-aware UCB Sampling

```
    Input: Candidate toolchains $\mathcal{A}$; dataset meta-features $x$; empirical stats $\mu(a), \sigma(a)$ for $a \in \mathcal{A}$;
    coefficients $\lambda > 0$, $\varepsilon \in [0, 0.1]$; early-noise rounds $T_{\text{noise}}$; batch sizes $K_{\text{pos}}, K_{\text{neg}}$; total rounds
    $T_{\text{rounds}}$.
 2: Output: Updated $\mu, \sigma$; Sampled set $\mathcal{D}$; logs.
 3: Initialize $\mathcal{D} \leftarrow \emptyset$, Logs $\leftarrow \emptyset$
 4: for $t = 1$ to $T_{\text{rounds}}$ do
 5:     // (1) UCB scoring
 6:     for each $a \in \mathcal{A}$ do
 7:         $S(a) \leftarrow \mu(a) + \lambda \cdot \sigma(a)$
 8:     end for
 9:     // (2) Early diversity noise
10:     if $t \le T_{\text{noise}}$ then
11:         for each $a \in \mathcal{A}$ do
12:             $S(a) \leftarrow S(a) + \text{Uniform}(-\varepsilon, +\varepsilon)$
13:         end for
14:     end if
15:     // (3) Contrast-aware selection
16:     $P \leftarrow$ TopK by $S(a)$
17:     $U \leftarrow$ TopK by $\sigma(a)$
18:     $L \leftarrow$ BottomK by $S(a)$
19:     // selects $K_{neg}$ negative candidates from union of low-score ($L$) and high-uncertainty ($U$) sets
20:     $N \leftarrow \text{DiverseMerge}(U, L, K_{\text{neg}})$
21:     $\mathcal{S}_{\text{batch}} \leftarrow P \cup N$
22:     // (4) Evaluate and update
23:     for each $a \in \mathcal{S}_{\text{batch}}$ do
24:         $y \leftarrow \text{Evaluate}(a, x)$
25:         UpdateMeanStd$(\mu(a), \sigma(a); y)$
26:         $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x, a, y)\}$
```

```
27:         Append log $(t, x, a, S(a), \mu(a), \sigma(a), y)$
28:     end for
29: end for
```

### A.6.2 Extracting Strategy Lessons via Contrastive Causal Reasoning.

The pseudocode of our Extracting Strategy Lessons via Contrastive Causal Reasoning is shown as follows:

**Algorithm 2** Extracting Strategy Lessons

```
    Input: Sampled set $\mathcal{D} = \{(x_i, a_i, y_i)\}_{i=1}^{N}$; threshold $\tau$; reasoning agent
 2: Output: Structure Lesson library $\mathcal{L} = \{\phi_1, \dots, \phi_K\}$
 3: // (1) Split $\mathcal{D}$ into positive and negative outcomes
 4: $\mathcal{D}^+ \leftarrow \{(x_i, a_i) \mid y_i \geq \tau\}$
 5: $\mathcal{D}^- \leftarrow \{(x_j, a_j) \mid y_j < \tau\}$
 6: // (2) Create contrastive tuples for each strategy
 7: for each strategy $a$ do
 8:     $\mathcal{C}_a \leftarrow \{(x_i, x_j) \mid (x_i, a) \in \mathcal{D}^+, (x_j, a) \in \mathcal{D}^-\}$
 9: end for
10: // (3) Invoke LLM agent to extract lessons
11: for each $\mathcal{C}_a$ do
12:     Provide $\mathcal{C}_a$ to agent
13:     Agent compares positive and negative meta-features under $a$
14:     Agent generates candidate explanatory statements $\{\phi\}$ describing the conditions for success/-
    failure
15:     $\mathcal{L} \leftarrow \mathcal{L} \cup \{\phi\}$
16: end for
```

### A.6.3 Lesson Verification via Probabilistic Generalization.

The following pseudocode of our Lesson Verification explicitly implements controlled paired comparisons, ensuring that the only manipulated factor is the presence or absence of the lesson $\phi_k$, thus eliminating confounding effects during verification.

**Algorithm 3** Lesson Verification

```
    Input: Lesson library $\mathcal{L} = \{\phi_1, \dots, \phi_K\}$; validation datasets with meta-features $\{x\}$; prede-
    fined success criterion $t$; confidence margin $\delta$
 2: Output: Verified lesson set $\mathcal{L}^* \subseteq \mathcal{L}$
 3: Initialize $\mathcal{L}^* \leftarrow \emptyset$
 4: for each lesson $\phi_k \in \mathcal{L}$ do
 5:     // Determine the validation datasets that satisfy the lesson condition $f_k$
 6:     $\mathcal{X}_k \leftarrow \{x \mid f_k(x) = \text{true}\}$
 7:     // Decide whether $\phi_k$ is a positive (guiding) or negative (warning) lesson
 8:     if $\phi_k$ is a positive (guiding) lesson then
 9:         // Controlled paired planning with lesson $\phi_k$
10:         $\mathcal{Y}_{\text{with}} \leftarrow \emptyset$
11:         for each $x \in \mathcal{X}_k$ do
12:             Agent plans strategy guided by $\mathcal{L}$ (containing $\phi_k$)
13:             Run forecasting with chosen strategy
14:             $y \leftarrow 1$ if result $\geq t$; else $0$
15:             Append $y$ to $\mathcal{Y}_{\text{with}}$
16:         end for
17:         $P(y = 1 \mid f_k, \phi_k) \leftarrow \text{mean}(\mathcal{Y}_{\text{with}})$
18:         // Controlled paired planning without lesson $\phi_k$ (all other conditions fixed)
19:         $\mathcal{Y}_{\text{without}} \leftarrow \emptyset$
20:         for each $x \in \mathcal{X}_k$ do
21:             Agent plans strategy guided by $\mathcal{L} \setminus \{\phi_k\}$
22:             Run forecasting with chosen strategy
23:             $y \leftarrow 1$ if result $\geq t$; else $0$
24:             Append $y$ to $\mathcal{Y}_{\text{without}}$
```

```
25:         end for
26:         $P(y = 1 \mid f_k, \neg\phi_k) \leftarrow \text{mean}(\mathcal{Y}_{\text{without}})$
27:         // Compute marginal causal effect under single-factor perturbation
28:         $\Delta_{\phi_k} \leftarrow P(y = 1 \mid f_k, \phi_k) - P(y = 1 \mid f_k, \neg\phi_k)$
29:     else
30:         // Negative lesson: compare follow vs. contradict $\phi_k$
31:         $\mathcal{Y}_{\text{follow}} \leftarrow \emptyset$
32:         for each $x \in \mathcal{X}_k$ do
33:             Agent plans strategy guided by $\mathcal{L}$ and constrained to satisfy $\phi_k$
34:             Run forecasting with chosen strategy
35:             $y \leftarrow 1$ if result $\geq t$; else 0
36:             Append $y$ to $\mathcal{Y}_{\text{follow}}$
37:         end for
38:         $P(y = 1 \mid f_k, \phi_k) \leftarrow \text{mean}(\mathcal{Y}_{\text{follow}})$
39:         $\mathcal{Y}_{\text{contra}} \leftarrow \emptyset$
40:         for each $x \in \mathcal{X}_k$ do
41:             Agent plans strategy under explicit intervention Contradict($\phi_k$)
42:             Run forecasting with chosen strategy
43:             $y \leftarrow 1$ if result $\geq t$; else 0
44:             Append $y$ to $\mathcal{Y}_{\text{contra}}$
45:         end for
46:         $P(y = 1 \mid f_k, \text{Contradict}(\phi_k)) \leftarrow \text{mean}(\mathcal{Y}_{\text{contra}})$
47:         // Compute marginal causal effect under contradiction-based intervention
48:         $\Delta_{\phi_k} \leftarrow P(y = 1 \mid f_k, \phi_k) - P(y = 1 \mid f_k, \text{Contradict}(\phi_k))$
49:     end if
50:     // Retain lesson if effect exceeds confidence margin
51:     if $\Delta_{\phi_k} > \delta$ then
52:         $\mathcal{L}^* \leftarrow \mathcal{L}^* \cup \{\phi_k\}$
53:     else
54:         Mark $\phi_k$ as low-confidence
55:     end if
56: end for
```

### A.6.4 Forecasting with Lesson-Guided Planning.

The pseudocode of our Lesson-Guided Planning Forecasting procedure is shown as follows:

**Algorithm 4** Forecasting with Lesson-Guided Planning

```
    Input: New dataset meta-features $x_{\text{new}}$; lesson library $\mathcal{L} = \{\phi_1, \ldots, \phi_K\}$; available toolchains $\mathcal{A}$
 2: Output: Selected forecasting strategy $a^*$; evaluation result
 3: // (1) Retrieve relevant lessons
 4: $\mathcal{L}_x \leftarrow \{\phi_k \in \mathcal{L} \mid f_k(x_{\text{new}}) = \text{true}\}$
 5: // (2) Build structured prompt
 6: Prompt $\leftarrow$ encode $(x_{\text{new}}, \mathcal{L}_x, \mathcal{A})$ into LLM input format
 7: // (3) Agent reasoning and strategy planning
 8: Agent $\leftarrow$ Prompt
 9: Agent outputs candidate strategy $a^*$
10: // (4) Forecasting execution
11: Run forecasting using $a^*$ on dataset with meta-features $x_{\text{new}}$
12: Collect predicted outputs and calculate evaluation result
```

## A.7 Prompt

**Extracting Strategy Lessons.** We use the following prompt for extracting Contrastive Causal Strategy Lessons:

```
You are an expert in causal analysis for time series forecasting. Given
    contrastive pairs under the same toolchain strategy a:
```

```
- Positives meta features: {Positive_meta}
- Negatives meta features: {Negative_meta}

Analyze the experimental results to extract causal lessons that explain
    why certain forecasting strategies succeed or fail under specific
    dataset characteristics.

Focus on identifying:
1. Causal relationships between dataset features and strategy performance
2. Necessary and sufficient conditions for strategy success
3. Causal mechanisms that explain the relationships
4. Confidence levels based on evidence strength

Your output should be Json format:
{
  "toolchain": "{Toolchain_name}",
  "important_features": "{Key_meta_features}",
  "description": "{Explanation}"
}

where "description" should be following condition (When), Result
    (Strategy succeeds / fails) and Causal explanation.

Here is an example:
Input: Positives meta features: {seasonal_peak = 24h, missing_rate =
    0.05, CV = 0.3}. Negatives meta features: {seasonal_peak = weak,
    missing_rate = 0.25, CV = 0.35}

Output: {
  "toolchain": "KNN-FFT-PatchTST",
  "important_features": "Strong seasonal peak, missing_rate < 0.1",
  "description": "This strategy succeeds when seasonal structure is
      clear and missingness is low, because FFT isolates periodicity and
      PatchTST captures residuals. When missing_rate > 0.2, imputation
      noise dominates and performance collapses."
}
```

**Lesson Verification.** We use the following prompt for LLM strategy selection during lesson verification.

```
You are an expert in time-series forecasting and tool planning.
Your task is to select the best forecasting strategy for the validation
    dataset based on causal lessons learned, so that we can verify
    whether these lessons can improve the forecasting performance.

You are given:
- Meta features of dataset: {Meta_features}
- Causal lessons: {Lessons_description}
- Available Strategies: {Tool_chains}

Select the most appropriate strategy for this dataset based on lessons.
    Consider:
1. Which lessons best match the dataset's characteristics?
2. Which strategies have the strongest verified causal effects?
3. How do the meta-features align with the lesson conditions?

Your output should be Json format:
{
  "strategy": "{Selected_toolchain}",
  "reason": "{Explanation}"
}

Here is an example:
Input:
```

```
- Meta features of dataset:
  {seasonal_peak=24h, missing_rate=0.06, volatility=medium,
      KPSS=non-stationary}
- Causal lessons:
  1. Lesson1: KNN-FFT-PatchTST succeeds when seasonal peak=24h and
      missing_rate<0.1.
  2. Lesson2: TimesNet-EMD-TimeMixer fails when volatility is low and
      trend is smooth.
- Available Strategies:
  1. KNN-FFT-PatchTST,
  2. TimesNet-EMD-TimeMixer,
  3. CEEMDAN-TimeMixer,
  4. PatchTST

Output:
{
  "strategy": "KNN-FFT-PatchTST",
  "reason": "The dataset exhibits strong seasonal peaks (24 hours) and a
      low missing rate (<0.1), which is consistent with the Lesson1."
}
```

**Lesson-Guided Planning.** We use the following prompt for LLM strategy selection in final Lesson-Guided Planning.

```
You are an expert in time series forecasting and tool planning.
Your task is to plan the best forecasting strategy for the new dataset
    by meta features and lessons learned. You should adaptively match
    the meta-features of the dataset with lessons learned. And select
    the most appropriate strategy based on these lessons learned.

You have the memory of causal lessons, each of which describes
the conditions for the success or failure of certain prediction
    strategies.

You are given:
- Meta features of the new dataset: {Meta_features}
- Causal lessons: {Lessons_description}
- Available strategies: {Tool_chains}

Where causal lessons describe the conditions for the success or failure
    of certain prediction strategies.

Select the most appropriate strategy for this dataset based on lessons.
    Consider:
1. Which lessons best match the dataset's characteristics?
2. Which strategies have the strongest verified causal effects?
3. How do the meta-features align with the lesson conditions?

Your output should be Json format:
{
  "strategy": "{Selected_toolchain}",
  "reason": "{[Phenomenon]: describe how the key tool inside the
      selected strategy performs under this dataset's meta-features.
      [Causal Analysis]: explain why these meta-features affect this
      tool's behavior, based on its algorithmic properties.}"
}

Here is an example:
Input:
- Meta features of dataset:
  {seasonal_peak=24h, missing_rate=0.06, volatility=medium,
      KPSS=non-stationary}
- Causal lessons:
  1. Lesson1: KNN-FFT-PatchTST succeeds when seasonal peak=24h and
      missing_rate<0.1.
  2. Lesson2: TimesNet-EMD-TimeMixer fails when volatility is low and
      trend is smooth.
- Available Strategies:
  1. KNN-FFT-PatchTST,
  2. TimesNet-EMD-TimeMixer,
  3. CEEMDAN-TimeMixer,
  4. PatchTST

Output:
{
  "strategy": "KNN-FFT-PatchTST",
  "reason": "[Phenomenon]: FFT shows strong frequency stability when the
      dataset has a clear 24-hour seasonal peak and a low missing_rate
      (0.06). This matches the success condition described in Lesson1.
      [Causal Analysis]: FFT relies on clean periodic components to
      isolate dominant cycles. Because the dataset satisfies strong
      seasonal structure and low missingness, FFT can extract meaningful
      signals, making strategies that include FFT (such as
      KNN-FFT-PatchTST) effective. TimesNet is avoided because the
      dataset does not match the failure condition described in Lesson2."
}
```

## A.8 Computational Cost Analysis

To analyze the computational efficiency of our paradigm, we compare BECRA against a representative AutoML framework. To ensure a fair comparison, we standardized the set of candidate toolchains for both methods. The cost of a single trial—defined as a full train-evaluation cycle for one toolchain on a given prediction horizon—was held constant. Our primary metric for computational cost is therefore the total number of trials required to find a high-performing strategy. For the AutoML baseline, we employed a Bayesian Optimization framework, a powerful and widely-used strategy for efficient search. For each new dataset, the AutoML process was initiated from scratch and run until its performance on a validation set converged.

| Method | Warm-start trials | New dataset trials | Total trials |
|---|---|---|---|
| AutoML (cold start) | — | 1177 | 1177 |
| BECRA (warm-started) | 158 | 53 | 211 |

Table 5: Comparison of trial costs for AutoML (cold start) vs. BECRA (warm-started). BECRA(Contrast-aware UCB) need a warm start on the dataset, meaning that the agent first explores on one (or a few) dataset to collect causal lessons and to initialize the sampling statistics $\mu(a)$ and $\sigma(a)$. In contrast, AutoML has to perform a cold start from scratch on every dataset. We totally have 6 datsets(ETTh1, ETTh2, ETTm1, ETTm2, Electricity, Weather). **Warm-start trials** denotes the number of trials BECRA requires on a single dataset, summed over the four prediction horizons. For BECRA, **New dataset trials** denotes the sum of trials conducted on each of the remaining 5 datasets over four prediction lengths, averaged over the choice of warm-start dataset. For AutoML, it denotes the sum of trials for all 6 datasets under cold start. **Total trials** denotes total trials for all 6 datasets; for BECRA, the value is averaged over the choice of warm-start dataset.

Table 5 reports the comparative trial costs. The AutoML mechanism has no warm start, thus requiring a total of 1,177 trials on 6 datasets. In contrast, BECRA can have a initial warm start on a single dataset with 158 trials, after which $\mu(a)$, $\sigma(a)$ and lessons can tranfer to new datasets. For the remaining 5 datasets, BECRA only requires 53 trials in total. Both BECRA trials in warm-start and the remaining datasets is averaged result, becaue warm-start can begin in any of 6 datasets. Therefore, BECRA reduces total trials by 5.6× (82%) compared to AutoML (1177 vs 211).

This disparity stems from a fundamental difference in paradigm. AutoML performs a one-shot, task-specific search, meaning its high computational cost is incurred repeatedly for every new dataset. It lacks a mechanism for accumulating or transferring knowledge across tasks, making it inherently unscalable in dynamic environments. BECRA, on the other hand, operates on a principle of knowledge accumulation and amortization. It invests a one-time, offline computational budget during its exploration phase to build a reusable, symbolic knowledge base of "lessons." This initial investment is then amortized over a lifetime of subsequent tasks. For any new dataset, BECRA bypasses the expensive search process entirely, instead leveraging its accumulated knowledge for near-instantaneous, low-cost planning. As the number of forecasting tasks grows, the marginal cost per task for BECRA approaches zero, an advantage that is particularly crucial in industrial settings with hundreds of evolving time-series.

Therefore, our analysis concludes that BECRA is not only superior in terms of predictive performance but also represents a far more scalable and economical paradigm than conventional AutoML systems. By shifting the focus from repetitive searching to reusable reasoning, BECRA is significantly better suited for the demands of large-scale and dynamic real-world forecasting environments.

### A.9 Uncertainty estimation and statistical reliability

Experiments have been run for multiple independent trials. Margin of Error (MOE) with 95% confidence intervals were included in the Table 6, ensuring that the performance differences can be evaluated for statistical reliability.

MOE quantifies uncertainty stemming from randomness in model initialization, data sampling, or stochastic training dynamics, and directly reflects the width of the corresponding confidence interval. A smaller MOE indicates that the sample mean is tightly concentrated around the true mean, while a larger MOE signals greater statistical uncertainty. This relationship allows differences between models to be interpreted in terms of statistical significance rather than raw fluctuations.

**Margin of Error (MOE) Calculation**

$$\text{MOE} = z_{\alpha/2} \times \left( \frac{\sigma}{\sqrt{n}} \right) \tag{1}$$

**Notation**

- $z_{\alpha/2}$: Critical value for the desired confidence level (e.g., 1.96 for 95% confidence)
- $\sigma$: Sample standard deviation across repeated runs
- $n$: Number of runs used to estimate uncertainty

| Dataset | BECRA | | TimeMixer | | TimesNet | | PatchTST | | iTransformer | |
|---|---|---|---|---|---|---|---|---|---|---|
| | MOE (MSE) | MOE (MAE) | MOE (MSE) | MOE (MAE) | MOE (MSE) | MOE (MAE) | MOE (MSE) | MOE (MAE) | MOE (MSE) | MOE (MAE) |
| ETTH1 | 3.11E-03 | 2.09E-03 | 3.19E-03 | 2.18E-03 | 6.98E-03 | 4.59E-03 | 1.80E-03 | 9.73E-04 | 4.62E-04 | 6.03E-04 |
| ETTH2 | 4.26E-03 | 2.62E-03 | 3.25E-03 | 2.97E-03 | 4.37E-03 | 2.62E-03 | 7.58E-03 | 4.10E-03 | 1.83E-03 | 7.77E-04 |
| ETTm1 | 3.75E-03 | 2.40E-03 | 4.84E-03 | 4.56E-03 | 4.94E-03 | 2.61E-03 | 2.86E-03 | 1.53E-03 | 2.38E-03 | 8.97E-04 |
| ETTm2 | 1.66E-03 | 1.07E-03 | 1.73E-03 | 6.92E-04 | 2.88E-03 | 7.86E-04 | 1.27E-03 | 1.21E-03 | 7.78E-04 | 1.59E-03 |
| Electricity | 4.08E-04 | 4.50E-04 | 4.29E-04 | 6.01E-04 | 6.86E-04 | 6.78E-04 | 1.96E-04 | 3.28E-04 | 3.21E-04 | 1.93E-04 |
| Weather | 1.10E-03 | 9.79E-04 | 9.05E-04 | 6.13E-04 | 2.34E-03 | 1.64E-03 | 3.76E-04 | 8.04E-04 | 7.67E-04 | 8.63E-04 |

Table 6: MOE comparison across models.

We report multi-run uncertainty (95% CI / MOE) for methods whose outcomes are affected by stochastic training (e.g., BECRA and trainable baselines such as TimeMixer, TimesNet, PatchTST, and iTransformer), using $n = 5$ independent random seeds. For fixed-checkpoint foundation models evaluated with deterministic inference (e.g., Chronos, Moirai/Moirai-2.0, Time-LLM, and GPT4TS; with dropout disabled and decoding fixed), repeated runs yield (near-)identical predictions, making across-run confidence intervals degenerate and uninformative. For closed-source or hosted LLM APIs (e.g., ChatGPT-4 and Gemini, and LLaMA models accessed via hosted APIs), repeated calls may conflate serving-side nondeterminism and model-version drift that are not controllable (e.g., no exposed random seed), thus the resulting variance is not directly comparable to training-seed variance; we therefore use deterministic decoding and report a single run for these models.

### A.10 Experiments Setup Details

**Device.** Dual NVIDIA GeForce RTX 5090 GPUs.

**Datasets and Tasks.** Seven public benchmarks are used: ETTh1, ETTh2, ETTm1, ETTm2, Electricity, Weather, and M4. M4 is used for short-term forecasting, while the other datasets are used for long-term forecasting.

- **ETTh1**
  - Domain: Electricity transformer temperature and load.
  - Size: $\sim$17,420 time steps.
  - Sampling frequency: Hourly.
  - Feature columns: 6 additional variables (after removing time and target).
- **ETTh2**
  - Domain: Electricity transformer temperature and load.
  - Size: $\sim$17,420 time steps.
  - Sampling frequency: Hourly.
  - Feature columns: 6 additional variables.
- **ETTm1**
  - Domain: Electricity transformer temperature and load.
  - Size: $\sim$69,680 time steps.
  - Sampling frequency: 15 minutes.
  - Feature columns: 6 additional variables.
- **ETTm2**
  - Domain: Electricity transformer temperature and load.
  - Size: $\sim$69,680 time steps.
  - Sampling frequency: 15 minutes.
  - Feature columns: 6 additional variables.
- **Electricity**
  - Domain: Electricity consumption across multiple clients.
  - Size: 26,304 time steps $\times$ 321 series.
  - Sampling frequency: Hourly.
  - Feature columns: 320 additional series (each treated as a feature in multivariate forecasting).
- **Weather**
  - Domain: Meteorological measurements from the Max Planck Institute.
  - Size: 52,696 time steps.
  - Sampling frequency: 10 minutes.
  - Feature columns: 20 additional meteorological variables.
- **M4**
  - Domain: Mixed real-world economic, demographic, financial, and industrial time series.
  - Size: 100,000 univariate series.
  - Sampling frequency: Mixed (yearly, quarterly, monthly, weekly, daily, hourly).
  - Feature columns: 0 (strictly univariate).

All models operate on consistent data splits following standard practice for each dataset.

**Baselines Used.** Below is a list of baseline models used in the experiement.

1. **ChatGPT-4** (Achiam et al., 2023): Used as an untrained forecasting agent without applying the BECRA paradigm; it directly plans the toolchain based on dataset meta-features and available tools. This baseline separates intrinsic LLM capabilities from BECRA's structured reasoning improvements.
2. **Gemini** (Team et al., 2023): Same usage and purpose as ChatGPT-4.
3. **LLaMA** (Touvron et al., 2023): Same usage and purpose as ChatGPT-4.
4. **TimeMixer** (Wang et al., 2024a): A decomposable multi-scale mixing architecture capturing temporal patterns at different frequencies.
5. **TimesNet** (Wu et al., 2022): A temporal 2D-variation modeling framework converting time series into multi-scale 2D representations.
6. **PatchTST** (Nie et al., 2022): A patch-based Transformer applying channel-independent self-attention over time-series patches.
7. **iTransformer** (Liu et al., 2023): An inverted Transformer modeling series along the feature dimension instead of the temporal dimension.
8. **Time-LLM** (Jin et al., 2023): A reprogramming-based approach adapting large language models for time-series forecasting.
9. **GPT4TS** (Zhou et al., 2023): A generative LLM-based forecaster leveraging pretrained language models to predict future values.
10. **Chronos** (Ansari et al., 2024): A pretrained probabilistic time-series foundation model trained on large-scale datasets.
11. **Moira** (Woo et al., 2024): A modular instruction-routing architecture with strong generalization across diverse tasks including time-series reasoning.

**Hyper-Parameters.** For long-term forecasting, the input sequence length is 96, with prediction horizons of 96, 192, 336, and 720. For M4, the input lengths are 12 and 96, and the prediction lengths are 6 and 48.

This configuration follows the dominant evaluation protocol in recent forecasting literature. Mainstream models such as TimesNet (Wu et al., 2022), iTransformer (Liu et al., 2023), and TimeMixer (Wang et al., 2024a) adopt the same settings, ensuring comparability and consistency with prior studies.

**Evaluation Protocol.** All long-term forecasting results are evaluated using Mean Squared Error (MSE) and Mean Absolute Error (MAE):

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

**Notation**

- $N$: Number of forecasted data points
- $y_i$: Ground-truth value at index $i$
- $\hat{y}_i$: Predicted value at index $i$

For short-term forecasting (M4), the evaluation uses SMAPE, MASE, and OWA, the official M4 metrics:

$$\text{SMAPE} = \frac{100\%}{N} \sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|}{\left(|y_i| + |\hat{y}_i|\right)/2}$$

$$\text{MASE} = \frac{\frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i|}{\frac{1}{N-m}\sum_{j=m+1}^{N} |y_j - y_{j-m}|}$$

$$\text{OWA} = \frac{1}{2}\left(\frac{\text{SMAPE}}{\text{SMAPE}_{\text{naive2}}}\right) + \frac{1}{2}\left(\frac{\text{MASE}}{\text{MASE}_{\text{naive2}}}\right)$$

**Notation**

- $N$: Number of forecasted points
- $m$: Seasonal period
- $y_i$: Ground-truth value
- $\hat{y}_i$: Predicted value
- $\text{SMAPE}_{\text{naive2}}$: SMAPE of the seasonal naïve baseline
- $\text{MASE}_{\text{naive2}}$: MASE of the seasonal naïve baseline