RAGPOT: LLM-based Retrieval-Augmented and Generative Prompt Optimization for Time-Series Forecasting

Anonymous ACL submission

Abstract

Recent advances in large language models (LLMs) have demonstrated their strong generalization ability across diverse reasoning and prediction tasks. However, directly applying LLMs to time-series forecasting remains challenging due to their sensitivity to prompt design and lack of domain-specific adaptation. In this work, we explore prompt optimization for time-series forecasting through a systematic framework that enables self-refining prompt improvement for LLMs. Our method iteratively improves prompt quality by first searching for similar multivariate historical time-series segments and then automatically updating the prompt from LLM's feedback, enabling the model to identify more informative and structured instructions for temporal reasoning. We conduct extensive experiments across multiple real-world datasets from energy, weather, finance, etc. domains, and demonstrate that optimized prompts lead to consistent and significant improvements over standard zero-shot and few-shot baselines. The results highlight the potential of prompt-based adaptation as an efficient alternative to parameter fine-tuning for applying LLMs in quantitative forecasting tasks.

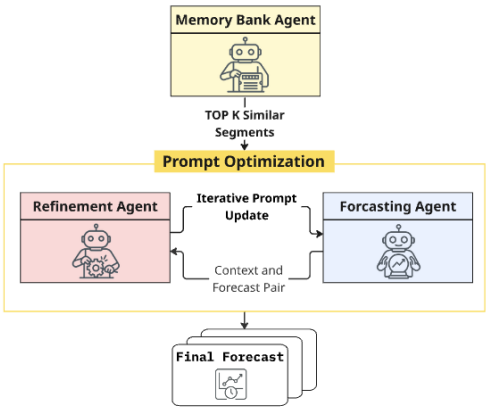


Figure 1: Overview of RAGPOT memory–driven forecasting framework.

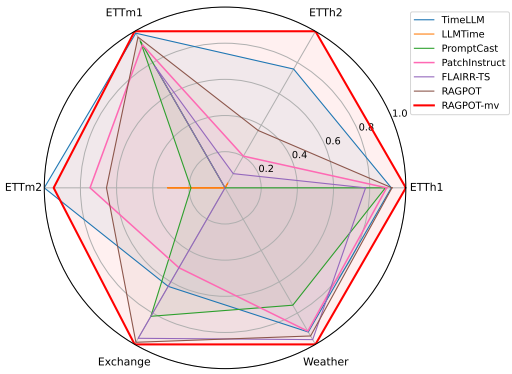


Figure 2: Cross dataset performance comparison of our approach using 1-normalized MSE

1 Introduction

Large language models (LLMs) have shown strong reasoning and generalization capabilities, motivating recent efforts to extend them to structured and quantitative domains such as time series forecasting (Bumb et al., 2025). Framing forecasting as a natural-language reasoning task offers a flexible and interpretable alternative to traditional models; however, its effectiveness is constrained by complex temporal dynamics and the sensitivity of LLMs to prompt design. Time series data often exhibit rich patterns including trends, seasonality, and stochastic fluctuations that demand careful contextual reasoning. Unlike task-specific forecasting models trained on labeled data, LLM based forecasters rely entirely on prompts to infer task structure, making them particularly vulnerable to small variations in prompt wording or structure, which can significantly limit robustness and generalization across datasets and forecasting horizons.

Recent work has explored iterative prompt refinement to reduce manual prompt engineering, notably through test-time optimization frameworks (Jalori et al., 2025). While effective, these methods operate on raw numerical sequences, implicitly assuming that LLMs can reason over time-series values as reliably as text. In contrast, we posit that representing historical temporal behavior through natural-language abstractions provides a more suitable interface for LLM-based forecasting than di-

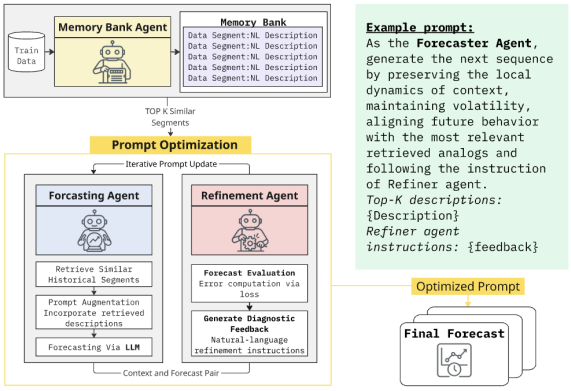


Figure 3: Architecture of the proposed RAGPOT framework.

rect numerical retrieval.

Motivated by this insight, we propose RAGPOT, a retrieval-augmented and generative prompt optimization framework that reinterprets historical time-series reasoning through a semantic memory perspective, as illustrated in 1. Instead of retrieving raw numerical segments at inference time, RAGPOT constructs a natural-language memory bank by converting historical segments into concise semantic descriptions during training. In inference, a forecasting agent composes prompts from the most relevant memory, while a refinement agent iteratively diagnoses forecasting errors and provides natural-language feedback to improve the reasoning context.

By unifying semantic memory construction, memory-driven forecasting, and iterative diagnostic refinement into a single workflow, our framework eliminates explicit numerical retrieval, reduces inference complexity, and improves interpretability through human-readable memory representations, while preserving the benefits of iterative prompt optimization.

We evaluate RAGPOT across six benchmark datasets using mean squared error (MSE). Figure 2 presents 1 – min–max normalized MSE in a radar-chart format, where larger enclosed areas indicate stronger and more consistent performance across datasets. RAGPOT and its multivariate variant achieve the largest overall areas, demonstrating robust generalization across diverse temporal characteristics.

2 Approach

In this section, we present our proposed approach. We summarize the approach in Algorithm 1 and provide the intuitive architecture in Figure 3.

2.1 Framework Overview

We propose LLM-based **R**etrieval-**A**ugmented and **G**enerative **P**rompt **O**ptimization for **T**ime-Series Forecasting (RAGPOT), a retrieval-augmented, zero-shot forecasting framework that that explicitly optimizes natural language prompts to reformulate time series prediction as a reasoning task for frozen large language models (LLMs). Instead of requiring learned domain-specific parameters, the system builds a compact semantic memory from historical data and employs iterative, feedback-driven prompt refinement to adapt forecasting behavior at test time. The framework consists of three cooperating agentsthe Memory Bank Agent, the Forecaster Agent, and the Refinement Agent.

At a high level, our approach is motivated by the observation that LLMs are far more effective at reasoning over structured natural-language concepts than over raw numeric sequences. Thus, our system constructs a semantic representation of historical patterns: each segment of the training series is annotated with a concise natural language description summarizing its trend, volatility, and characteristic future behavior. This enriched memory enables the LLM to perform analogical reasoning over linguistically meaningful abstractions of time-series shape. The key idea is to steer a pretrained LLM using structured prompts enriched by:

1. **Memory Bank Agent** — retrieves semantically and statistically similar historical patterns using an adaptive, correlation-weighted similarity function (Section 2.3.1)

2. **Forecasting Agent** — converts the contextual statistics and retrieved exemplars into a structured natural-language prompt for forecasting (Section 2.3.2)

3. **Refinement Agent** — iteratively assesses the coherence of the generated forecast and corrects the prompt through diagnostic feedback (Section 2.3.3)

Collectively, RAGPOT formulates time series forecasting as an iterative natural language reasoning process, in which a semantic memory constructed by the Memory Bank Agent guides extrapolation, and diagnostic feedback generated by the Refinement Agent progressively sharpens predictive behavior. This design enables frozen LLMs

to adapt to new time-series regimes in an efficient and interpretable manner, while avoiding the brittleness of numeric nearest-neighbor retrieval and the rigidity of static prompts.

2.2 Problem Formulation

Before introducing the details of our method, we first formalize the time series forecasting problem considered in this work. Specifically, we define the forecasting setup, notation, and assumptions under which our prompt optimization framework operates.

Time Series Forecasting Setup: Let

$$\mathcal{X} = \{x_1, x_2, \ldots, x_T\}$$

denote a univariate time series. For forecasting, we assume a **context window** of length L:

$$\mathbf{x}_{\text{ctx}} = \{x_{t-L+1}, \ldots, x_t\}.$$

The goal is to produce an H step forecast:

$$\hat{Y} = \{\hat{y_1}, ..., \hat{y_H}\}.$$

where $\hat{y_h} \approx x_{T+h}$. We evaluate forecasting quality using a generic loss (error) functional ℓ:

$$\ell : \mathbb{R}^H \times \mathbb{R}^H \to \mathbb{R}_{\geq 0}$$

where the functional ℓ can instantiate MAE, RMSE, pinball loss, CRPS, or any other proper scoring rule depending on the forecasting objective.

Prompt-conditioned Forecasting with LLMs: We model an LLM forecaster as a prompt-conditioned function

$$\hat{\mathbf{Y}} = \textsc{ForecasterLLM}\big(P,\, \mathbf{X}_{\text{ctx}},\, \mathcal{D}_k,\, H\big),$$

where $P \in \mathcal{P}$ is a natural-language prompt, $\mathcal{D}_k$ denotes retrieved descriptions from the memory bank(e.g., the top -K most similar historical segments), and H is the forecasting horizon. The LLM parameters are frozen throughout the process.

Given a prompt P, the forecasting loss E is

$$E(P) = \ell\big(\hat{Y}, \mathbf{X}_{t:t+H}\big).$$

Prompt Optimization Objective:

Unlike conventional forecasting methods that optimize model parameters, our objective is to optimize the prompt itself. Given a dataset $\mathcal{X}$, retrieval memory $\mathcal{M}$, and loss function ℓ, we seek an optimized prompt

$$P^* = \arg\min_{P \in \mathcal{P}} \mathbb{E}_{t \sim \mathcal{T}} \left[E_t(P) \right],$$

where $\mathcal{T}$ indexes forecasting windows sampled from the dataset.

Iterative Prompt Refinement:

Starting from an initial prompt P_0, the optimization proceeds iteratively. At refinement iteration r, the LLM produces a forecast

$$\hat{\mathbf{Y}}^{(r)} = \textsc{ForecasterLLM}\big(P^{(r)},\, \mathbf{X}_{\text{ctx}},\, \mathcal{D}_k,\, H\big)$$

with corresponding loss

$$E^{(r)} = \ell\big(\hat{\mathbf{Y}}^{(r)}, \mathbf{X}_{t:t+H}\big).$$

Based on forecast error and ground truth, a refinement module generates feedback via $\textsc{RefineLLM}(\widehat{Y}^{(r)}, X_{t:t+H}, E^{(r)})$ that updates the prompt:

$$P^{(r+1)} = \mathcal{U}\big(P^{(r)}, \text{Feedback}^{(r)}\big),$$

where $\mathcal{U}(\cdot)$ denotes a prompt update operator instantiated by ApplyFeedbackToPrompt in Algorithm 1.

We maintain the best-performing prompt:

$$P_{\text{best}} = \arg\min_r E^{(r)},$$

which is returned by Algorithm 1 and used to generate the final forecast.

2.3 RAGPOT

RAGPOT consists of three cooperating agents that communicate entirely through natural-language instructions: the Memory Bank Agent, the Forecasting Agent, and the Refinement Agent. Each agent encapsulates a logically distinct part of the pipeline. Below, we summarize the responsibilities of each agent and indicate where its behavior is implemented in the algorithms.

2.3.1 Memory Bank Agent

The Memory Bank Agent constructs a compact library of representative historical segments by sampling the training series at approximately uniform intervals, instead of enumerating all sliding windows. For each sampled segment s_i, the agent queries an LLM to produce a natural-language description d_i, and the pair (s_i, d_i) is stored in the memory bank. This procedure is implemented in the BuildMemoryBank function of Algorithm 2 (lines 1–9), where the step size Δ is chosen such that the N memory entries are spread roughly evenly across the training series.

Algorithm 1 RAGPOT

Input: Training data X, Historical series $X_{1:t-1}$, Horizon H, Initial prompt P_0, Context length L_{ctx}, Sequence length L_{seq}, Memory bank $\mathcal{M}$, Number of segments K, Max iterations N_{iter}, Convergence threshold ϵ_{conv}, Diagnostic tolerance τ_{diag}, loss function ℓ (MAE, RMSE, etc),
Output: Optimized prompt P_{out}

1: $P_{\text{curr}} \leftarrow P_0$; $P_{\text{best}} \leftarrow P_0$; $E_{\min} \leftarrow \infty$; $\hat{Y}_{\text{best}} \leftarrow$ nil; $X_{\text{ctx}} \leftarrow X_{t-L_{ctx}:t-1}$; $Instr \leftarrow [\,]$
2: $\mathcal{M} \leftarrow \textsc{BuildMemoryBank}(X)$ ▷ Algorithm 2 store time series segmentation, and corresponding natural language description
3: **for** $r = 1$ **to** N_{iter} **do**
4: $D_k \leftarrow \textsc{Retrieve}(X_{\text{ctx}}, \mathcal{M}, K)$ ▷ Fetch top-K description for the most similar segments with X_{ctx}
5: $\hat{Y}^{(r)} \leftarrow \textsc{ForecasterLLM}(P_{\text{curr}}, X_{\text{ctx}}, D_k, H)$
6: $\mathrm{E}^{(r)} \leftarrow \ell(\hat{Y}^{(r)}, X_{t:t+H})$
7: **if** $\mathrm{E}^{(r)} < \mathrm{E}_{\min}$ **then**
8: $\mathrm{E}_{\min} \leftarrow \mathrm{E}^{(r)}$, $P_{\text{best}} \leftarrow P_{\text{curr}}$, $\hat{Y}_{\text{best}} \leftarrow \hat{Y}^{(r)}$
9: **if** $\frac{|\mathrm{E}^{(r)} - \mathrm{E}^{(r-1)}|}{\mathrm{E}^{(r-1)}} < \epsilon_{\text{conv}}$ **then break** ▷ Early stop: diagnostics and MAE have converged
10: Feedback $\leftarrow \textsc{RefineLLM}(\widehat{Y}^{(r)}, X_{t:t+H}, E^{(r)})$
11: $Instr \leftarrow Instr \cup \{\text{Feedback}\}$ ▷ Append refinement feedback to instruction buffer
12: $P_{\text{curr}} \leftarrow \textsc{ApplyFeedbackToPrompt}(P_{\text{curr}}, Instr)$ ▷ Update prompt using accumulated instructions (Sec. 2.3.3)
13: **return** P_{best}

Algorithm 2 Agent 1: Memory Bank Agent

Input: Training series $X = \{x_1, \ldots, x_T\}$, segment number N, segment length τ, number of retrieved segments K
Output: Memory bank $\mathcal{M}$ and retrieval set D_j for a given context window X_{ctx}

1: **function** $\textsc{BuildMemoryBank}(X)$
2: Initialize $\mathcal{M} \leftarrow \varnothing$
3: $A \leftarrow T - \tau + 1$
4: $\Delta \leftarrow \max(1, \lfloor A/N \rfloor)$
5: **for** $i = 1$ to A **step** Δ **do**
6: $s_i \leftarrow \{x_i, \ldots, x_{i+\tau-1}\}$
7: $d_i \leftarrow \textsc{Describe}(s_i)$
8: $\mathcal{M} \leftarrow \mathcal{M} \cup \{(s_i, d_i)\}$
9: **return** $\mathcal{M}$

10: **function** $\textsc{Retrieve}(X_{\text{ctx}}, \mathcal{M}, K)$
11: $d_{\text{ctx}} \leftarrow \textsc{DescribeContext}(X_{\text{ctx}})$
12: **for** each $(s_j, d_j) \in \mathcal{M}$ **do**
13: $\text{score}_j \leftarrow \text{Pearson}(s_j,\ s_{\text{ctx}})$
14: $D_j \leftarrow$ top-K items ranked by score_j
15: **return** D_j

At inference time, the agent retrieves those memories whose descriptions are most similar to the current context window X_{ctx}. The retrieval procedure is implemented in the RETRIEVE function of Algorithm 2 (lines 10–15). The agent first constructs a description d_{ctx} of X_{ctx}, then computes a Pearson-correlation similarity between the raw numeric segments s_j and s_{ctx}. The top-K descriptions ranked by this score form the retrieval set D_j, which is passed to the Forecasting Agent to condition its prediction. The complete prompt template used by the Memory Bank Agent is provided in Appendix F (Figure 9)

2.3.2 Forecasting Agent

The Forecasting Agent is responsible for producing an H-step forecast from the current context window and the retrieved analog memories, conditioned on the accumulated refinement instructions. Its behavior is captured in Algorithm 1. At iteration r, the agent receives the current prompt P_{curr}, the context window X_{ctx}, the retrieval set D_k returned by the Memory Bank Agent, and the forecast horizon H. The working prompt P_{curr} always encodes four pieces of information:

1. a numeric summary of the recent context X_{ctx},
2. natural-language descriptions of the retrieved analog segments D_k,
3. the task specification and horizon H,
4. the current instruction buffer $Instr$, which aggregates all feedback produced by the Refinement Agent in previous iterations.

Given this prompt, the FORECASTERLLM call in Algorithm 1 (lines 5) queries the frozen LLM to obtain an H-step prediction $\widehat{Y}^{(r)}$. The forecast $\widehat{Y}^{(r)}$ is then evaluated against the ground-truth window $X_{t:t+H}$ (line 6), and if it improves upon the best loss observed so far, the corresponding prompt P_{curr} is stored as the current best prompt (lines 7–8). In the next iteration, the Forecasting Agent uses the updated P_{curr}, whose instruction component has been augmented by the Refinement Agent, enabling the LLM to progressively adjust its reasoning behavior over time. The complete prompt template used by the Forecaster Agent is provided in Appendix F (Figure 10)

2.3.3 Refinement Agent

The Refinement Agent implements the feedback mechanism that enables RAGPOT to improve its forecasts across iterations without updating model parameters. After the Forecasting Agent produces the current prediction $\widehat{Y}^{(r)}$ and the corresponding error $E^{(r)}$ is computed (line 6), the Refinement Agent examines the forecast together with the ground-truth window $X_{t:t+H}$ and generates targeted natural-language feedback.

This process is carried out by the RefineLLM call in Algorithm 1 (line 10), which prompts the LLM to identify structural inconsistencies such as incorrect trend direction, volatility mismatch, boundary discontinuities, or deviations from patterns present in the recent context. Instead of overwriting prior messages, the resulting feedback snippet is appended to the instruction buffer $Instr$, allowing refinement signals to accumulate across iterations.

In line 12 of Algorithm 1, the updated instruction buffer is merged back into the working prompt through the ApplyFeedbackToPrompt procedure. The resulting prompt P_{curr} is then passed to the next iteration of the Forecasting Agent, enabling the LLM to adjust its reasoning trajectory based on the accumulated diagnostics. Through this iterative prompt-space update mechanism, the Refinement Agent gradually steers the LLM toward more accurate and structurally coherent predictions without requiring any gradient-based training or modification of the underlying model parameters. The complete prompt template used by the Refiner Agent is provided in Appendix F (Figure 11)

2.4 RAGPOT (MV)

We also developed another variant of RAGPOT for multivariate forecasting settings. Let the multivariate series be

$$\mathbf{X}_t = \big(x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(V)}\big),$$

where $x_t^{(1)}$ is the target variable and $x_t^{(2:V)}$ are auxiliary variables. The goal remains to forecast the target over horizon H using the same iterative prompt optimization procedure as in Algorithm 1.

In the univariate setting, the Memory Bank Agent ranks memory segments using similarity computed only on the target series (Algorithm 2, line 10–15). In RAGPOT (MV), we modify the scoring step in Retrieve to incorporate auxiliary variables by computing per-variable Pearson similarities and aggregating them:

$$\text{score}_j^{\text{MV}} = 0.5 \cdot \text{Pearson}\big(s_j^{(1)},\, s_{\text{ctx}}^{(1)}\big) + 0.5 \cdot \frac{1}{V-1} \sum_{v=2}^{V} \text{Pearson}\big(s_j^{(v)},\, s_{\text{ctx}}^{(v)}\big),$$

The fixed weighting is used as a naive yet stable choice in our current experiments. The top-K items ranked by $\text{score}_j^{\text{MV}}$ form the retrieval set D_k, used in Algorithm 1 at line 4.

We further augment the Refinement Agent by conditioning diagnostic feedback on the multivariate trajectories. Concretely, we modify the RefineLLM call in Algorithm 1 (line 10) to take the auxiliary-variable window $X_{t:t+H}^{(2:V)}$ in addition to the target forecast and ground truth. This enables the refiner to detect cross-variable inconsistencies.

RAGPOT (MV) is effective when forecasting performance depends on capturing joint patterns across variables, so multivariate similarity improves retrieval and multivariate diagnostics improve prompt refinement.

3 Experiments

In this section, we systematically investigate our approach across a diverse set of forecasting tasks.

3.1 Experimental Setup

Across all experiments, forecasting is formulated as a prediction task conditioned on a fixed-length historical context, without any parameter training or fine-tuning of the underlying large language model. Unless otherwise mentioned, we adopt a rolling-origin evaluation scheme to construct multiple forecasting windows from the held-out portion of each time series. Specifically, the first 10% of each series is reserved for training, and all remaining timestamps are used for evaluation through rolling windows. Each input window consists of the most recent 96 observations (*context length* = 96), and the prediction target covers the next $h \in \{1, 2, 4, 8, 16, 32\}$ steps (*prediction length*). Starting from the earliest available evaluation point, the forecast origin is advanced by 30 timestamps (*stride* = 30). To balance temporal diversity and computational cost, we cap the number of sampled windows per series at 100 (*max_windows* = 100). This setup ensures consistent context–forecast pairs across all horizons, enabling a fair comparison under both short- and long-term forecasting settings.

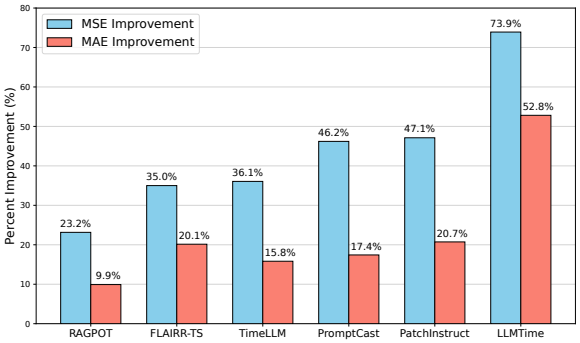


Figure 4: Percent improvement of our RAGPOT-MV compared to the other approaches. Notably, our approaches significantly improve over the others.

Baselines. We compare RAGPOT against five baselines spanning two categories: single-prompt LLM forecasting methods and prompt optimization approaches. In particular, we compared our approaches to LLMTime (Gruver et al., 2024), Time-LLM (Jin et al., 2024), PromptCast (Bumb et al., 2025), PatchInstruct (Bumb et al., 2025), and FLAIRR-TS (Jalori et al., 2025). Further implementation details are provided in Appendix B.

Benchmark Datasets. We evaluate RAGPOT on six widely used time-series forecasting benchmarks spanning diverse temporal resolutions, domains, and dynamics: ETTh1, ETTh2, ETTm1, ETTm2, Weather, and Exchange Rate. Table 3 reports key statistics for all six datasets, including the number of variables, sampling frequency, sequence length, and application domain. Across datasets, we follow standard experimental protocols, using all available variables as model inputs and adopting commonly used target variables for evaluation. More detailed dataset descriptions are provided in Appendix C.

Metrics. We evaluate forecasting performance using Mean Squared Error (MSE) and and Mean Absolute Error (MAE). All metrics are computed on z-score normalized series, where each forecast window is standardized using statistics from its corresponding context window. This normalization ensures fair comparison across datasets with different scales and prevents high-variance series from dominating absolute error values. Evaluating in the normalized domain provides consistent and meaningful metrics across horizons and datasets.

3.2 Results

In this section, we compare our approaches to the state-of-the-art baselines across six datasets from a wide range of domains. The full results are provided in Table 2. For the overall results, please see Table 1 where we average MSE and MAE for all approaches across all datasets. Notably, we find that our approaches almost always achieve the best average performance across all benchmarks and horizons (Table 1). RAGPOT (MV) achieves the best forecasting performance compared to all other approaches for each horizon, as shown in Table 1. Furthermore, in nearly all cases, our other simpler approach, called RAGPOT, achieves the next best performance compared to the other state-of-the-art baselines (Table 1). We see that across all forecasting horizons, our approaches, RAGPOT-MV and RAGPOT achieve the best and second best average performance (Table 1 last row). Further, we summarize the percent improvement of our approach, RAGPOT-MV across all other methods in Figure 4. Strikingly, our approach achieves non-trivial improvements in performance across all state-of-the-art methods, achieving a mean percent improvement in performance over the best baseline method of 35% (FLAIRR-TS) in terms of MSE and 15.8% in terms of MAE (TimeLLM).

To investigate how the performance of our approaches scales with forecast length, Table 1 aggregates errors across all datasets for each horizon. We find that our approach, RAGPOT (MV), attains the lowest MAE at every horizon, and its advantage becomes more pronounced at longer horizons, indicating improved long-range stability. When the forecasting horizon is small, all methods rely predominantly on local temporal patterns. In this regime, RAGPOT approaches perform competitively across datasets and consistently rank among the best approaches (Table 1). As the prediction horizon increases, the models must capture broader structural cues rather than rely solely on immediate local continuity, and this is where our approaches achieve the most reliable performance across datasets under both MSE and MAE (Table 1). The integration of semantic analog descriptions helps preserve meaningful trend direction, volatility regimes, and future behavior, allowing the forecasts to remain coherent even when temporal dependencies extend beyond the observed window.

3.3 Ablation Study

We conduct several ablation experiments to better understand the effectiveness of each component.

Memory Bank Ablation: As shown in Figure 7,

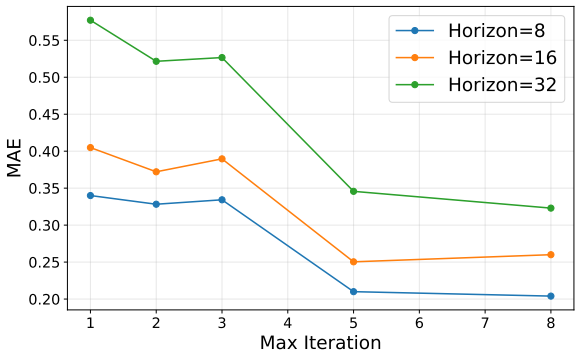
Table 1: Forecasting results averaged across datasets for each forecasting horizon. Lower values indicate better performance. The final row reports the overall average across all horizons and datasets.

		RAGPOT		RAGPOT (MV)		FLAIRR-TS		TimeLLM		LLMTime		PromptCast		PatchInstruct	
	Horizon	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
Horizon Avg	1	0.048	0.107	**0.027**	**0.093**	0.107	0.158	0.049	0.117	0.518	0.392	0.098	0.124	0.074	0.158
	2	0.061	0.129	**0.039**	**0.114**	0.116	0.174	0.059	0.127	0.516	0.404	0.105	0.142	0.076	0.162
	4	0.092	0.170	**0.069**	**0.153**	0.145	0.208	0.099	0.174	0.532	0.427	0.130	0.179	0.117	0.198
	8	0.167	0.238	**0.144**	**0.220**	0.219	0.269	0.184	0.243	0.594	0.466	0.215	0.248	0.303	0.279
	16	0.312	0.340	**0.271**	**0.318**	0.343	0.359	0.419	0.374	0.684	0.518	0.415	0.364	0.491	0.371
	32	0.539	0.465	**0.387**	**0.408**	0.513	0.468	0.653	0.517	0.743	0.565	0.779	0.525	0.711	0.486
Overall Avg	—	0.203	0.242	**0.156**	**0.218**	0.240	0.273	0.244	0.259	0.598	0.462	0.290	0.264	0.295	0.275

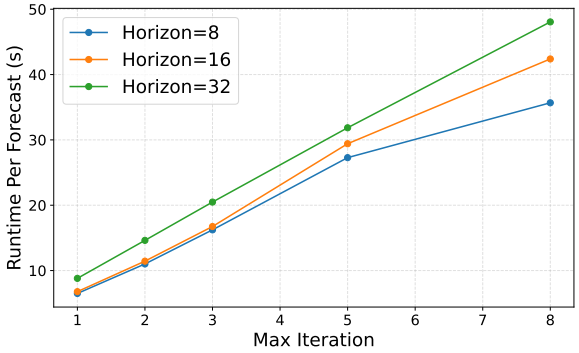
Table 2: Forecasting results across multiple datasets and horizons.

		RAGPOT		RAGPOT (MV)		FLAIRR-TS		TimeLLM		LLMTime		PromptCast		PatchInstruct	
Datasets	**Horizon**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
ETTh1	1	0.023	0.097	**0.015**	**0.083**	0.069	0.185	0.024	0.102	0.316	0.399	0.018	0.089	0.032	0.116
	2	0.025	0.111	0.019	0.097	0.065	0.184	**0.014**	**0.088**	0.315	0.407	0.022	0.104	0.035	0.125
	4	0.038	0.137	0.031	0.123	0.073	0.196	**0.027**	**0.122**	0.327	0.420	0.035	0.134	0.046	0.150
	8	0.070	0.189	0.058	0.171	0.097	0.229	**0.053**	**0.165**	0.350	0.433	0.071	0.188	0.078	0.199
	16	0.129	0.260	**0.105**	0.236	0.163	0.290	0.105	**0.232**	0.356	0.442	0.138	0.269	0.125	0.259
	32	0.188	0.325	**0.121**	**0.264**	0.249	0.349	0.256	0.362	0.329	0.436	0.250	0.366	0.206	0.328
	Avg	0.079	0.186	**0.058**	**0.162**	0.119	0.239	0.080	0.178	0.332	0.423	0.089	0.192	0.087	0.196
ETTh2	1	0.152	0.230	**0.086**	0.200	0.286	0.308	0.153	0.223	0.373	0.405	0.107	**0.138**	0.275	0.380
	2	0.200	0.287	**0.131**	0.258	0.322	0.348	0.186	0.245	0.423	0.450	0.136	**0.182**	0.286	0.395
	4	0.307	0.383	0.246	0.358	0.432	0.434	0.351	0.364	0.518	0.513	**0.222**	**0.267**	0.443	0.487
	8	0.591	0.546	0.557	0.532	0.736	0.597	0.619	0.506	0.804	0.646	**0.540**	**0.451**	0.811	0.666
	16	1.032	0.751	**0.980**	0.741	1.121	0.788	0.983	**0.678**	1.083	0.764	1.249	0.729	1.031	0.765
	32	1.519	0.882	**0.914**	0.730	1.288	0.873	0.962	**0.709**	1.065	0.781	2.054	0.954	1.182	0.829
	Avg	0.633	0.513	**0.486**	0.470	0.697	0.558	0.542	0.454	0.711	0.593	0.718	**0.454**	0.671	0.587
ETTm1	1	0.006	0.050	0.008	0.056	0.022	0.077	0.004	0.046	0.099	0.225	**0.003**	**0.043**	0.014	0.072
	2	0.011	0.058	0.011	0.060	0.026	0.084	**0.005**	0.051	0.102	0.232	0.007	**0.050**	0.019	0.075
	4	0.017	0.070	0.015	0.069	0.031	0.095	**0.010**	**0.065**	0.101	0.237	0.013	0.065	0.023	0.085
	8	0.023	0.090	0.019	0.088	0.033	0.112	**0.018**	**0.085**	0.101	0.241	0.021	0.092	0.028	0.106
	16	0.036	0.126	**0.031**	**0.123**	0.048	0.147	0.036	0.126	0.114	0.253	0.038	0.135	0.041	0.141
	32	0.076	0.190	**0.064**	**0.176**	0.097	0.213	0.083	0.199	0.122	0.261	0.097	0.210	0.066	0.183
	Avg	0.028	0.097	**0.025**	**0.095**	0.043	0.121	0.026	0.095	0.106	0.242	0.030	0.099	0.032	0.110
ETTm2	1	0.048	0.084	**0.010**	**0.053**	0.167	0.155	0.016	0.057	0.158	0.171	0.117	0.102	0.019	0.099
	2	0.052	0.100	**0.013**	0.070	0.173	0.170	0.019	**0.068**	0.152	0.176	0.119	0.114	0.019	0.100
	4	0.067	0.139	**0.028**	0.109	0.190	0.206	0.030	**0.100**	0.164	0.204	0.131	0.146	0.041	0.140
	8	0.108	0.205	0.067	0.177	0.228	0.268	**0.050**	**0.148**	0.197	0.254	0.162	0.204	0.097	0.218
	16	0.225	0.316	0.178	0.296	0.310	0.361	**0.152**	**0.259**	0.258	0.336	0.270	0.317	0.242	0.350
	32	0.547	0.513	0.482	0.498	0.589	0.548	0.463	**0.463**	**0.434**	0.476	0.681	0.545	0.550	0.552
	Avg	0.175	0.226	0.130	0.201	0.276	0.285	**0.122**	**0.182**	0.227	0.270	0.247	0.238	0.161	0.243
Weather	1	0.038	0.107	**0.022**	**0.086**	0.043	0.113	0.062	0.145	1.020	0.531	0.194	0.216	0.055	0.151
	2	0.051	0.128	**0.031**	**0.106**	0.056	0.134	0.051	0.143	1.006	0.547	0.190	0.226	0.051	0.150
	4	0.071	0.167	**0.046**	**0.138**	0.070	0.165	0.076	0.172	1.008	0.574	0.209	0.258	0.077	0.178
	8	0.119	0.223	**0.083**	**0.188**	0.111	0.215	0.136	0.238	1.029	0.600	0.273	0.305	0.145	0.242
	16	0.261	0.324	**0.174**	**0.273**	0.215	0.301	0.283	0.348	1.170	0.662	0.426	0.394	0.294	0.349
	32	0.570	0.507	**0.445**	**0.441**	0.480	0.451	0.648	0.543	1.249	0.732	0.933	0.599	0.665	0.546
	Avg	0.185	0.243	**0.134**	**0.205**	0.163	0.230	0.209	0.265	1.080	0.608	0.371	0.333	0.214	0.269
Exchange	1	**0.021**	**0.075**	0.021	0.081	0.052	0.111	0.034	0.131	1.141	0.621	0.147	0.158	0.051	0.130
	2	0.028	**0.091**	**0.027**	0.092	0.056	0.124	0.078	0.169	1.099	0.614	0.154	0.176	0.048	0.124
	4	0.049	0.125	**0.047**	**0.122**	0.072	0.152	0.097	0.219	1.075	0.614	0.171	0.206	0.069	0.146
	8	0.091	0.177	**0.081**	**0.165**	0.109	0.195	0.230	0.317	1.083	0.622	0.221	0.248	0.658	0.240
	16	0.187	0.265	**0.158**	**0.237**	0.198	0.269	0.957	0.604	1.121	0.650	0.372	0.339	1.216	0.361
	32	0.334	0.371	**0.293**	**0.337**	0.375	0.371	1.507	0.828	1.257	0.701	0.661	0.475	1.597	0.477
	Avg	0.118	0.184	**0.104**	**0.172**	0.144	0.204	0.484	0.378	1.129	0.637	0.288	0.267	0.606	0.246

removing the memory bank causes forecasting errors to grow rapidly with horizon length, indicating that the model over-relies on short-term continuity. Incorporating memory stabilizes long-range behavior by providing historical structural constraints, yielding significantly flatter error curves.



((a)) Error vs. Iteration



((b)) Runtime vs. Iteration

Figure 5: Results showing the performance and runtime of our approach as the max number of iterations varies.

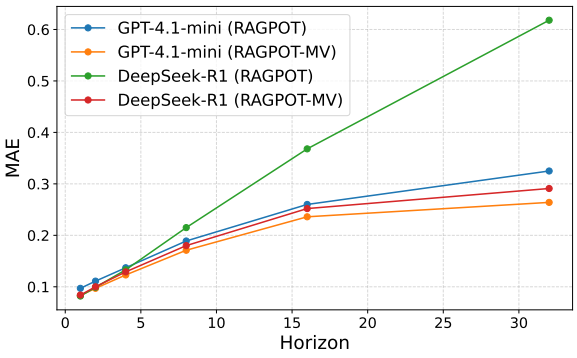


Figure 6: Results comparing our approach with various base LLMs.

Varying Top-k Retrieval Size. As shown in Figure 8, Very small k values underprovide context, while very large values introduce noise from weakly relevant segments. A moderate k achieves the best balance between relevant structural cues and retrieval precision.

Varying Iterations of Prompt Optimization: As shown in Figure 5, Most of the performance gain occurs within the first few refinement steps, with diminishing returns afterward. Longer horizons benefit more from additional iterations, while runtime scales nearly linearly with iteration count.

Multivariate Addition: Incorporating auxiliary variables consistently improves performance when cross-variable correlations are meaningful, especially at longer horizons. However, gains diminish when auxiliary signals are weak or noisy, indicating dataset-dependent utility.

Alternative LLMs: We compare RAGPOT under GPT-4.1-mini and DeepSeek-R1-0528 and find that both backbones produce similar accuracy trends, indicating that the framework is largely backbone-agnostic, shown in Figure 6 . GPT-4.1-mini provides more stable performance across horizons, while DeepSeek occasionally achieves lower short-term errors but with higher variance. Overall, the gains mainly stem from the RAGPOT framework itself, with backbone choice affecting stability rather than predictive pattern.

4 Conclusion

In this work, we introduced a retrieval-augmented prompt optimization framework for zero-shot time-series forecasting with large language models. By combining a semantic memory bank, analogy-guided forecasting, and iterative shape-aware refinement, the framework enables frozen LLMs to capture both local and long-range temporal structure without any parameter training. This design allows the method to adapt flexibly to different prediction lengths, making it suitable for real-world scenarios where forecasting horizons vary by application. The empirical results show that the framework delivers consistently strong performance across both short- and long-term horizons, reflecting its robustness to diverse datasets and temporal patterns. We also explored a multivariate extension, where auxiliary features are incorporated during forecasting. This variant offers additional benefits when cross-variable relationships are informative, while also highlighting the need for further study on multivariate integration strategies. Beyond accuracy, the method offers practical advantages: it avoids the computational cost of training and often requires only a small number of refinement iterations, enabling efficient inference and broad applicability. As future work, we plan to investigate more principled use of multivariate information, expand evaluation to wider temporal domains, and further analyze the interpretability of LLM-based analogical reasoning.

5 Limitations

While the proposed framework demonstrates strong zero-shot forecasting capability, several limitations remain. First, the method relies on natural-language descriptions generated from segmented historical data. Although these semantic representations enable more flexible analogical reasoning, their quality depends on how effectively the LLM captures salient temporal characteristics. In settings with highly irregular dynamics or weakly structured patterns, the memory descriptions may lose fidelity, potentially reducing retrieval usefulness. Second, the current multivariate extension incorporates auxiliary features only during the forecasting stage. This design is effective when cross-variable dependencies are strong, but can provide limited benefit—or even introduce noise—when correlations are weak or dataset-specific. A more principled mechanism for constructing multivariate memories or performing feature-aware retrieval could further improve robustness across heterogeneous domains. Finally, the approach inherits its broader limitations of LLM-based reasoning, such as sensitivity to prompt phrasing and potential variability across model families. While semantic prompting provides interpretability and adaptability, its behavior may differ depending on model scale or pretraining data. Systematically evaluating the framework across diverse LLM architectures is therefore an important direction for future research.

References

Paweł Batorski, Adrian Kosmala, and Paul Swoboda. 2025. Prl: Prompts from reinforcement learning.

Mayank Bumb, Anshul Vemulapalli, Sri Harsha Vardhan Prasad Jella, Anish Gupta, An La, Ryan A. Rossi, Hongjie Chen, Franck Dernoncourt, Nesreen K. Ahmed, and Yu Wang. 2025. Forecasting time series with llms via patch-based prompting and decomposition.

Ching Chang, Wei-Yao Wang, Wen-Chih Peng, and Tien-Fu Chen. 2025. Llm4ts: Aligning pre-trained llms as data-efficient time-series forecasters. *ACM Trans. Intell. Syst. Technol.*, 16(3).

Lichang Chen, Jiuhai Chen, Tom Goldstein, Heng Huang, and Tianyi Zhou. 2023. Instructzero: Efficient instruction optimization for black-box large language models.

Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. 2024. Black-box prompt optimization: Aligning large language models without model training.

Yumin Choi, Jinheon Baek, and Sung Ju Hwang. 2025. System prompt optimization with meta-learning.

Mohammadmahdi Ghasemloo and Alireza Moradi. 2025. Informed forecasting: Leveraging auxiliary knowledge to boost llm performance on time series forecasting.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. 2024. Large language models are zero-shot time series forecasters.

Jiaxin Guo, Daimeng Wei, Yuanchang Luo, Shimin Tao, Hengchao Shang, Zongyao Li, Shaojun Li, Jinlong Yang, Zhanglin Wu, Zhiqiang Rao, and Hao Yang. 2024. M-ped: Multi-prompt ensemble decoding for large language models.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2025. Evoprompt: Connecting llms with evolutionary algorithms yields powerful prompt optimizers.

Bairu Hou, Joe O'Connor, Jacob Andreas, Shiyu Chang, and Yang Zhang. 2023. PromptBoosting: Black-box text classification with ten forward passes. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 13309–13324. PMLR.

Cho-Jui Hsieh, Si Si, Felix X. Yu, and Inderjit S. Dhillon. 2023. Automatic engineering of long prompts.

Wenyang Hu, Yao Shu, Zongmin Yu, Zhaoxuan Wu, Xiangqiang Lin, Zhongxiang Dai, See-Kiong Ng, and Bryan Kian Hsiang Low. 2024. Localized zeroth-order prompt optimization.

Yash Jain and Vishal Chowdhary. 2025. Local prompt optimization. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 75–81, Albuquerque, New Mexico. Association for Computational Linguistics.

Gunjan Jalori, Preetika Verma, and Sercan Ö Arık. 2025. Flairr-ts – forecasting llm-agents with iterative refinement and retrieval for time series.

Haozhe Ji, Pei Ke, Hongning Wang, and Minlie Huang. 2024. Language model decoding as direct metrics optimization.

Yushan Jiang, Wenchao Yu, Geon Lee, Dongjin Song, Kijung Shin, Wei Cheng, Yanchi Liu, and Haifeng Chen. 2025. Explainable multi-modal time series prediction with llm-in-the-loop.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. 2024. Time-llm: Time series forecasting by reprogramming large language models.

Aya Kayal, Sattar Vakili, Laura Toni, Da shan Shiu, and Alberto Bernacchia. 2025. Bayesian optimization from human feedback: Near-optimal regret bounds.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2023. Dspy: Compiling declarative language model calls into self-improving pipelines.

Weize Kong, Spurthi Amba Hombaiah, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 2024. Prewrite: Prompt rewriting with reinforcement learning.

Yilun Kong, Hangyu Mao, Qi Zhao, Bin Zhang, Jingqing Ruan, Li Shen, Yongzhe Chang, Xueqian Wang, Rui Zhao, and Dacheng Tao. 2025. Qpo: Query-dependent prompt optimization via multi-loop offline reinforcement learning.

Belinda Z. Li, Alex Tamkin, Noah Goodman, and Jacob Andreas. 2023. Eliciting human preferences with language models.

Xiaoqiang Lin, Zhongxiang Dai, Arun Verma, See-Kiong Ng, Patrick Jaillet, and Bryan Kian Hsiang Low. 2024. Prompt optimization with human feedback.

Chenxi Liu, Qianxiong Xu, Hao Miao, Sun Yang, Lingzheng Zhang, Cheng Long, Ziyue Li, and Rui Zhao. 2025a. Timecma: Towards llm-empowered multivariate time series forecasting via cross-modality alignment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(18):18780–18788.

Jianhua Liu, Liwen Cao, Yanru Wu, Zijie Zhao, and Yang Li. 2025b. Learning optimal prompt ensemble for multi-source visual prompt transfer.

Peiyuan Liu, Hang Guo, Tao Dai, Naiqi Li, Jigang Bao, Xudong Ren, Yong Jiang, and Shu-Tao Xia. 2025c. Calf: Aligning llms for time series forecasting via cross-modal fine-tuning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(18):18915–18923.

Shengcai Liu, Caishun Chen, Xinghua Qu, Ke Tang, and Yew-Soon Ong. 2024a. Large language models as evolutionary optimizers.

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. 2024b. Large language models to enhance bayesian optimization. In *The Twelfth International Conference on Learning Representations*.

Junru Lu, Siyu An, Min Zhang, Yulan He, Di Yin, and Xing Sun. 2024a. Fipo: Free-form instruction-oriented prompt optimization with preference dataset and modular fine-tuning schema.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity.

Yao Lu, Jiayi Wang, Raphael Tang, Sebastian Riedel, and Pontus Stenetorp. 2024b. Strings from the library of babel: Random sampling as a strong baseline for prompt optimisation.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024. Eureka: Human-level reward design via coding large language models.

Kawin Mayilvaghanan, Varun Nathan, and Ayush Kumar. 2025. PROPEL: Prompt optimization with expert priors for small and medium-sized LLMs. In *Proceedings of the 4th International Workshop on Knowledge-Augmented Methods for Natural Language Processing*, pages 272–302, Albuquerque, New Mexico, USA. Association for Computational Linguistics.

John A. Miller, Mohammed Aldosari, Farah Saeed, Nasid Habib Barna, Subas Rana, I. Budak Arpinar, and Ninghao Liu. 2024. A Survey of Deep Learning and Foundation Models for Time Series Forecasting. ArXiv:2401.13912 [cs].

Zijie Pan, Yushan Jiang, Sahil Garg, Anderson Schneider, Yuriy Nevmyvaka, and Dongjin Song. 2024. $\mathbf{S}^2$ip-llm: Semantic space informed prompt learning with llm for time series forecasting.

Seonghwan Park, Jaehyeon Jeong, Yongjun Kim, Jaeho Lee, and Namhoon Lee. 2025. Zip: An efficient zeroth-order prompt tuning for black-box vision-language models.

Felipe Maia Polo, Ronald Xu, Lucas Weber, Mírian Silva, Onkar Bhardwaj, Leshem Choshen, Allysson Flavio Melo de Oliveira, Yuekai Sun, and Mikhail Yurochkin. 2024. Efficient multi-prompt evaluation of llms.

Mayk Caldas Ramos, Shane S. Michtavy, Marc D. Porosoff, and Andrew D. White. 2025. Bayesian optimization of catalysis with in-context learning.

Lennart Schneider, Martin Wistuba, Aaron Klein, Jacek Golebiowski, Giovanni Zappella, and Felice Antonio Merra. 2025. Hyperband-based bayesian optimization for black-box prompt selection.

KaShun Shum, Shizhe Diao, and Tong Zhang. 2024. Automatic prompt augmentation and selection with chain-of-thought from labeled data.

Tianxiang Sun, Zhengfu He, Hong Qian, Yunhua Zhou, Xuanjing Huang, and Xipeng Qiu. 2022. BBTv2: Towards a gradient-free future with large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3916–3930, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jialiang Tang, Shuo Chen, Chen Gong, Jing Zhang, and Dacheng Tao. 2025. Llm-ps: Empowering large language models for time series forecasting with temporal patterns and semantics.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P. Xing, and Zhiting Hu. 2023. Promptagent: Strategic planning with language models enables expert-level prompt optimization.

Zesen Wang, Lijuan Lan, and Yonggang Li. 2025. Timeprompt: Integrated heterogeneous prompts for unlocking llms in time series forecasting.

Malcolm L. Wolff, Shenghao Yang, Kari Torkkola, and Michael W. Mahoney. 2025. Using pre-trained llms for multivariate time series forecasting.

Hao Xue and Flora D. Salim. 2023. Promptcast: A new prompt-based learning paradigm for time series forecasting.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers.

Chun-Pai Yang, Kan Zheng, and Shou-De Lin. 2025a. Plhf: Prompt optimization with few-shot human feedback.

Silin Yang, Dong Wang, Haoqi Zheng, and Ruochun Jin. 2025b. Timerag: Boosting llm time series forecasting via retrieval-augmented generation. In *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

Zhuo Yang, Daolang Wang, Lingli Ge, Beilun Wang, Tianfan Fu, and Yuqiang Li. 2025c. Reasoning bo: Enhancing bayesian optimization with long-context reasoning power of llms.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. 2024. Self-taught optimizer (stop): Recursively self-improving code generation.

Yiming Zhang, Shi Feng, and Chenhao Tan. 2022a. Active example selection for in-context learning.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022b. Automatic chain of thought prompting in large language models.

Zhe Zhao, Pengkun Wang, Haibin Wen, Shuang Wang, Liheng Yu, and Yang Wang. 2025. Stem-lts: Integrating semantic-temporal dynamics in llm-driven time series analysis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(21):22858–22866.

Zixiao Zhu, Hanzhang Zhou, Zijian Feng, Tianjiao Li, Chua Jia Jim Deryl, Mak Lee Onn, Gee Wah Ng, and Kezhi Mao. 2025. Rethinking prompt optimizers: From prompt merits to optimization.

Appendix

A Related Work

A.1 Prompt Optimization

Research on prompt optimization has developed rapidly across a wide range of natural language and multimodal tasks, producing several complementary methodological families.

Preference-driven Prompt Optimization. One prominent direction replaces hard-to-specify numeric scores with human preference signals and efficient querying strategies. Early studies show that prompts can be optimized by directly eliciting pairwise preferences (Lin et al., 2024), single-round supervision (Yang et al., 2025a), or black-box alignment through human choices (Cheng et al., 2024), while other works build interactive elicitation frameworks that progressively refine prompts based on human feedback (Li et al., 2023). These approaches highlight the central role of preference data in defining what constitutes a 'good' prompt when ground truth labels or metrics are ambiguous.

Reinforcement Learning for Prompt Search. Building on this, a second family explores reinforcement learning based optimization, where prompt generation is framed as a policy learning problem. Methods in this line include training dedicated prompt generators that maximize rewards (Batorski et al., 2025), query-dependent schemes that adapt to task-specific signals via offline multi loop RL (Kong et al., 2025), and rewriting-based approaches where RL agents iteratively refine candidate prompts (Kong et al., 2024). Related work also illustrates that LLM-guided reward search can outperform expert-engineered heuristics (Ma et al., 2024), further underscoring the synergy between reinforcement learning and language model reasoning in designing effective prompts.

Zeroth-Order / Gradient-Free Methods. Orthogonal to RL, gradient-free or zeroth-order methods have been proposed to improve query efficiency in black-box API settings. These methods explore the prompt space without requiring gradients, using strategies such as low-rank reparameterization (Park et al., 2025) to reduce dimensionality, Gaussian-process guidance to escape local optima (Hu et al., 2024), and layer-wise continuous prompts that can be tuned in a modular fashion (Sun et al., 2022). Others introduce token-level locality search mechanisms (Jain and Chowd-

hary, 2025), further reducing the cost of optimizing prompts for large models accessed only through API calls.

Ensembles and Evaluation Protocols. Another active area focuses on ensembles and evaluation protocols to mitigate the sensitivity of model outputs to individual prompts. By combining multiple prompt candidates through probability-space aggregation (Guo et al., 2024), boosting weak prompt learners (Hou et al., 2023), or learning weighted combinations of different prompts (Liu et al., 2025b), ensemble approaches achieve more robust performance. Complementary work has also developed systematic evaluation frameworks for multi-prompt settings (Polo et al., 2024), aiming to provide more reliable measurements of prompt effectiveness across tasks and domains.

ICL-Oriented Optimization. Prompt optimization has also been closely linked to advances in in-context learning (ICL). A large body of work aims to optimize how demonstrations, rationales, and instructions are selected and organized. Representative methods include automatic rationale generation (Zhang et al., 2022b; Shum et al., 2024), analysis of order sensitivity in few-shot examples (Lu et al., 2022), and reinforcement-based strategies for demonstration selection (Zhang et al., 2022a). Other approaches automate the engineering of long prompts to scale in-context learning to larger contexts (Hsieh et al., 2023), or even establish stochastic baselines that rival more structured optimization procedures (Lu et al., 2024b).

Programmatic and Agentic Pipelines. Beyond single-prompt optimization, researchers have also proposed programmatic and agentic frameworks that automate entire pipelines. For instance, declarative programming approaches compile model calls into self-improving modules (Khattab et al., 2023), recursive self-taught optimizers refine prompts by continuously revisiting earlier outputs (Zelikman et al., 2024), and planning-based agents perform strategic search with tree-based reasoning (Wang et al., 2023). In parallel, data- and principle-driven frameworks have introduced expert-prior role decomposition (Mayilvaghanan et al., 2025), merit-based optimization criteria (Zhu et al., 2025), modular fine-tuning schemas (Lu et al., 2024a), and cross-model instruction transfer (Chen et al., 2023), further diversifying the design space of APO.

Bayesian Optimization for Prompts. From a more statistical perspective, Bayesian optimization (BO) has been adapted to prompt search. Multi-fidelity scheduling has been explored to allocate search budget efficiently (Schneider et al., 2025), while BO enhanced with LLM reasoning capabilities has been shown to accelerate convergence (Liu et al., 2024b). Works on regret analysis with human feedback formalize guarantees for preference-based optimization (Kayal et al., 2025), while reasoning-augmented BO leverages LLMs' long-context reasoning to guide search (Yang et al., 2025c). Other studies demonstrate domain-specific adaptations (Ramos et al., 2025), collectively showing how prompt optimization can be framed as a structured search problem with theoretical underpinnings.

LLMs as Optimizers. Finally, a distinct and rapidly growing body of research frames LLMs themselves as optimizers, closing the loop between natural language reasoning and optimization. In this paradigm, the LLM is not only the target model, but also the agent conducting the search. Iterative optimizers such as OPRO generate candidate prompts from histories of scored attempts (Yang et al., 2024), while evolutionary hybrids like EvoPrompt (Guo et al., 2025) and LLM-based evolutionary optimizers (Liu et al., 2024a) connect LLM reasoning with evolutionary search strategies. Other approaches focus on aligning decoding distributions directly with target metrics (Ji et al., 2024), showing that prompts can be optimized by leveraging the generative and reasoning capabilities of LLMs without external machinery. This perspective reframes APO as a higher-order application of LLMs in which they reason about and optimize their own interfaces.

Closest Work and Gap. Closest to our work is system prompt optimization with meta-learning (Choi et al., 2025), which also formulates prompt optimization as a bilevel problem. However, that line of work optimizes general-purpose system prompts across NLP datasets, without targeting domain-specific forecasting tasks. Within time-series research, recent efforts have proposed prompt-based forecasting paradigms such as PromptCast (Xue and Salim, 2023), which reformulates forecasting as text generation, FLAIRR-TS (Jalori et al., 2025), which iteratively refines prompts with retrieval and reasoning at test time,

and S^2 IP-LLM (Pan et al., 2024), which aligns semantic and temporal spaces to improve forecasting accuracy. Other works like LLM-Prompt / Time-Prompt (Wang et al., 2025) investigate prompt-driven interfaces for time-series prediction. Yet, none of these combine LLMs with meta-learning to construct adaptive prompts across tasks, nor do they explicitly formulate forecasting as a meta-optimization problem over both representations and in-context learning.

A.2 LLM-based Forecasting

Goal. Our goal is to learn such prompt given all these factors in a time-efficient manner, that achieves near-optimal performance while significantly reducing the cost (of searching over a large set of prompts to find reasonable prompt for a specific dataset, task, LLM, etc).

Zero/Few-shot Forecasting as Text. Many works show that large language models can act as zero/few-shot forecasters by recasting numeric sequences as text and exploiting in-context learning. Early evidence demonstrates strong zero-shot extrapolation when time series are digit tokenized and treated as next-token prediction, including handling side information and missing values (Gruver et al., 2024); Follow-ups refine prompting with long/short decomposition and re-assessment loops (Bumb et al., 2025) and provide systematic analyses of when LLMs succeed (periodic, trend-rich data) and how input phrasing/knowledge injection helps (Miller et al., 2024). Work on domain applications echoes these findings; for example, energy systems reformulate load as textual sequences for autoregressive generation (Chang et al., 2025), human mobility forecasting introduces time-aware prompts and contextual stays (Liu et al., 2025a) and a language foundation pipeline for POI flows (Zhao et al., 2025), while finance studies highlight both potential and current gaps, from negative zero shot results on return prediction (Ghasemloo and Moradi, 2025) to domain-specialized pre-training that excels in financial NLP (Jiang et al., 2025).

Interfaces Between Series and Tokens. A parallel thread designs interfaces between numeric series and LLM token spaces, often through tokenization, pre-training, or patching. Tokenized probabilistic modeling with T5-style architectures yields strong seen-task and zero-shot accuracy across 42 datasets (Wolff et al., 2025); a universal, VQVAE-based tokenization plus Transformers supports forecasting/classification/translation across five domains with notable zero-shot transfer (Tang et al., 2025). Frequency-vectorization treats spectra as a shared dictionary for tokenization and pre-training objectives to align the time / frequency domains (Wang et al., 2025), while cross-domain SSL explores PLMs as encoder initializations with a reconstruction-driven discretizer (Liu et al., 2025c). Patch-level ideas further adapt LLMs: multi-patch self-supervision with patch-wise decoding for transferable representations (Bumb et al., 2025), and promptable patch/decomposition strategies that exploit neighbors and inter-series correlations (Yang et al., 2025b). Decoder-only LLMs are explicitly repurposed as autoregressive forecasters with series-to-token projections and textual timestamps for multivariate alignment (Jin et al., 2024).

Multi-/Cross-Modal with Knowledge. Because many real settings couple numbers with text/knowledge, multi-modal and cross-modal approaches align temporal signals with language semantics. Unified prompt paradigms, soft / hard prompts and cross-modal fusion aim to bridge modality gaps (Liu et al., 2025a; Zhao et al., 2025; Liu et al., 2025c); Semantic anchor retrieval aligns LLM word embeddings with series embeddings to select informative prompts (Yang et al., 2025b); Cross-modal fine-tuning reduces distribution discrepancies with matching/regularization/consistency losses (Liu et al., 2025c). Multimodal pipelines integrate external news/knowledge; for example, a GDELT-based dataset and prompting framework for news-impact forecasting (Ghasemloo and Moradi, 2025); explainable stock forecasting that jointly reasons over prices, metadata, and news (Jiang et al., 2025); RAG-style entity extraction/alignment with PrimeKG for EHR readmission/mortality (Tang et al., 2025); contrastive + masked multimodal pretraining for ICU tasks (Tang et al., 2025); and text-conditioned physiological synthesis (Tang et al., 2025). Beyond raw fusion, LLM-derived knowledge graphs augment structure learning and representation quality in multivariate time series (Chang et al., 2025).

Spatiotemporal Forecasting and Planning. Several works combined LLMs to spatiotemporal forecasting. Tokenizers/adapters convert graph-temporal inputs into tokens aligned with LLM semantics (Liu et al., 2025a), and partially frozen

attention plus spatial-temporal embeddings enable robust traffic prediction, including few/zero-shot settings (Zhao et al., 2025). Language-driven simulation and planning further demonstrate controllable generative models or grounded reasoning for future trajectories: language-guided diffusion for scene-level traffic control (Tang et al., 2025), geometric grounding of LLM predictions in semantic maps for human-aware planning (Tang et al., 2025). More broadly, LLMs exhibit general sequence-pattern completion that spans numeric/state sequences and policy reasoning (Miller et al., 2024).

Test-time Agents, Supervision, and Tools. At the prompt/agent layer, test-time refinement and retrieval can close gaps without full tuning. An agentic forecaster with a refiner that iteratively updates the prompts, conditioned on past outputs and retrieval, improves accuracy, and approaches specialized models (Jalori et al., 2025). The series supervision and labeling pipelines show that LLMs can act as virtual annotators when paired with SSL encoders (Tang et al., 2025), guide anomaly detection students through knowledge distillation (Yang et al., 2024), and even exploit label semantics in sequence-to-text decoders for HAR (Tang et al., 2025). Tool-use data sets and decision tree search expand the ability of an LLM to call external APIs, useful for data access, diagnostics, and evaluation loops in forecasting agents (Khattab et al., 2023; Wang et al., 2023).

TS–Language Alignment Resources. Finally, time-series–language alignment resources support the descriptive and analytical capabilities needed for forecasting. A 100k pair dataset aligns series windows with GPT-generated trend descriptions to train an LMM that outperforms strong multimodal baselines in insight generation (Tang et al., 2025). In finance, hybrid models align LLM news embeddings and quantitative features using local-global modeling and reinforcement alignment for improved returns (Jiang et al., 2025).

Why a new methodology? Taken together, the above strands reveal both promise and persistent gaps that motivate our approach. (i) *Prompt sensitivity and transfer*: prior systems improve single-task performance yet remain brittle across datasets, horizons, and tokenizations, with small wording changes causing large variance; ensemble or retrieval remedies reduce but do not remove this instability. (ii) *Search cost and supervision*: black-box, RL, or BO-based optimizers (Cheng et al., 2024; Kong et al., 2024; Schneider et al., 2025; Liu et al., 2024b) can be sample- and budget-intensive, and many methods still assume task-specific heuristics, reward shaping, or auxiliary labels. (iii) *Temporal structure mismatch*: ICL- and agent-style refinements (Gruver et al., 2024; Bumb et al., 2025; Jalori et al., 2025) seldom *explicitly* encode multi-horizon, long/short components, or dataset-level priors that generalize across domains. (iv) *Objective misalignment*: most pipelines optimize a single metric (e.g., MAE) per horizon, overlooking multi-objective trade-offs (stability vs. accuracy vs. cost) and meta-level generalization across tasks. (v) *Limited meta-adaptivity*: while meta-learning for prompts is emerging (Choi et al., 2025), it largely targets generic NLP system prompts rather than domain-specific, horizon-aware TS prompts. Our methodology addresses these issues by (a) casting prompt search as a *task-conditioned, horizon-aware optimization* over textual reasoning patterns; (b) using the LLM itself to generate *structured feedback* that jointly scores accuracy, stability, and budget, enabling *multi-objective* selection without parameter updates; and (c) introducing *cross-dataset priors* that regularize prompt evolution for robust transfer. This design preserves the training-free advantages of prompt-based TS forecasting while providing principled control over variance, cost, and generalization—thereby bridging the gap between ad-hoc prompt tuning and scalable, domain-adaptive forecasting.

B Baselines Description

We compare our proposed framework against five representative baselines from two general categories: (1) *LLM-based forecasting* approaches that leverage a single prompt to perform sequence continuation or prompt reprogramming, and (2) *prompt optimization* baselines that either evaluate a large prompt space or adopt different selection strategies for common-sense prompting.

- **LLMTime: zero-shot sequence continuation (Gruver et al., 2024)** presents the last L numeric values (comma-separated or digit strings) followed by a one-line instruction to predict the next h future values. This baseline evaluates the direct sequence continuation ability of LLMs without any task-specific adaptation.

- **Time-LLM: prompt reprogramming (PaP) (Jin et al., 2024)** introduces a fixed prefix with textual prototypes that "reprogram" the input series, followed by the numerical context and target horizon. This reprogramming helps align the LLM's text generation behavior with time-series forecasting semantics.

- **PromptCast: time-aware instruction prompting (Bumb et al., 2025)** encodes both temporal context (timestamps) and semantic hints (unit, subject label) in natural language, enabling the model to better capture periodic patterns and contextual meaning in multivariate series.

- **PatchInstruct: patch-based prompting (Bumb et al., 2025)** provides a few exemplars of (*input patch* $\rightarrow$ *next patch*) pairs as local temporal patterns, followed by the current input patch and an instruction to generate the next h values. This allows the model to generalize across similar local patterns.

- **FLAIRR-TS: retrieval-augmented forecasting (Jalori et al., 2025)** retrieves semantically similar past time-series contexts and incorporates them into the LLM prompt as relevant exemplars. This retrieval-augmented strategy enables the model to leverage historical analogs for improved generalization.

C Dataset Description

We evaluate RAGPOT on six widely used time-series forecasting benchmarks spanning diverse temporal resolutions, domains, and dynamics: ETTh1, ETTh2, ETTm1, ETTm2, Weather, and Exchange Rate. These datasets collectively test our framework across periodic, non-stationary, stochastic, and highly multi-dimensional scenarios. For a summary of the datasets and statistics, please see Table 3.

ETTh1, ETTh2 (Electricity Transformer Temperature—Hourly). The ETTh1 and ETTh2 datasets contain hourly measurements from electricity transformers, including oil temperature, ambient temperature, and load variables. Each dataset provides 7 variables over 17,420 timestamps, exhibiting strong daily and weekly periodicity. Following established protocols, the "oil temperature" variable is used as the prediction target while all variables are used as inputs.

ETTm1, ETTm2 (Electricity Transformer Temperature—15 min). The ETTm1 and ETTm2 datasets provide the same set of 7 variables but sampled at a finer 15-minute resolution, resulting in 69,680 timestamps. These datasets contain richer short-term fluctuations and more granular periodic patterns, offering a more challenging forecasting setting than their hourly counterparts.

Weather. The Weather dataset includes 21 meteorological indicators collected every 10 minutes throughout 2020 from a research station in Germany. Variables include air temperature, humidity, atmospheric pressure, wind components, shortwave radiation, and others. Compared to the ETT datasets, Weather exhibits stronger non-stationarity and rapid local variability, providing a stress test for generalization under fast-changing dynamics.

Exchange Rate. This dataset consists of daily exchange rates for 8 foreign currencies over several years. The series are noisy, weakly periodic, and driven by stochastic economic factors. Forecasting on this dataset primarily evaluates robustness to low-signal, non-seasonal financial dynamics.

D Additional Results

See table 2. These results indicate the effectiveness of our approaches as we achieve the best average performance across all horizons for nearly all datasets for both MSE and MAE. These results indicate the effectiveness of our approaches as we achieve the best average performance across all horizons for nearly all datasets for both MSE and MAE.

E Ablation Study

We conduct several ablation experiments to better understand the contribution of each component in our framework.

1. **Memory Bank Ablation:** We test the effect of removing or reducing the size of the memory bank to examine its role in maintaining temporal and contextual consistency across iterations, showing that memory contributes to long-term reasoning stability and forecast coherence.

   As shown in Figure 7, the memory bank appears to reshape how errors grow over time. Without memory, the model's error increases

Table 3: Summary of the six datasets used in our experiments. Each dataset varies in sampling frequency, number of variables, sequence length, and domain characteristics.

Dataset	# Vars	Freq.	Length	Domain
ETTh1	7	Hourly	17,420	Electricity (Transformer Temp.)
ETTh2	7	Hourly	17,420	Electricity (Transformer Temp.)
ETTm1	7	15 min	69,680	Electricity (Transformer Temp.)
ETTm2	7	15 min	69,680	Electricity (Transformer Temp.)
Weather	21	10 min	36,761	Meteorology (2020, Germany)
Exchange Rate	8	Daily	7,588	Finance (FX Rates)

Table 4: Comparison of our approaches with different base models (GPT-4.1-mini and DeepSeek-R1-0528)

		MSE ↓						RMSE ↓						MAE ↓					
LLM	**Method**	1	2	4	8	16	32	1	2	4	8	16	32	1	2	4	8	16	32
GPT-4.1-mini	RAGPOT	0.023	0.025	0.038	0.070	0.129	0.188	0.097	0.120	0.154	0.216	0.304	0.379	0.097	0.111	0.137	0.189	0.260	0.325
	RAGPOT(MV)	0.015	0.019	0.031	0.058	0.105	0.121	0.083	0.107	0.139	0.198	0.280	0.311	0.083	0.097	0.123	0.171	0.236	0.264
DeepSeek-R1	RAGPOT	0.015	0.020	0.036	0.096	0.322	1.249	0.082	0.109	0.152	0.252	0.439	0.738	0.082	0.099	0.133	0.215	0.368	0.618
	RAGPOT(MV)	0.015	0.019	0.033	0.065	0.122	0.157	0.084	0.111	0.147	0.208	0.297	0.342	0.084	0.100	0.129	0.180	0.252	0.291

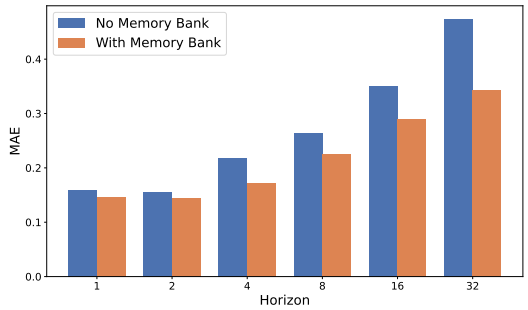


Figure 7: Memory bank ablation across forecasting horizons.

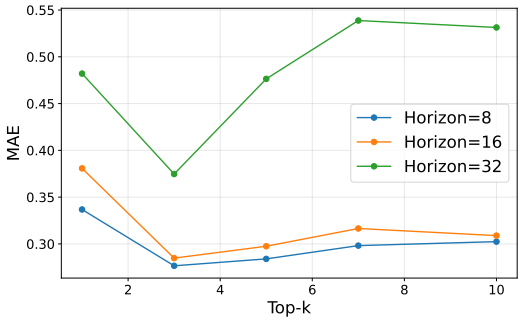


Figure 8: Results comparing our approach as top-k is increases.

almost exponentially as the forecasting window expands, suggesting that the model relies heavily on short-range patterns and struggles to maintain coherent temporal structure over long horizons. In contrast, the memory-augmented model exhibits a markedly different failure mode: while it still degrades at extreme horizons, its error growth curve is substantially flatter, indicating that retrieved historical contexts act as an anchor that constrains long-range drift.

2. **Vary top-k retrieval size**. Under a fixed memory size (mem=40), we further study how many retrieved candidates should be incorporated during the reasoning process ($k = 1, 3, 5, 7, 10$).

   As shown in Figure 8, we find that extremely small k values degrade performance, likely because the model receives insufficient contextual diversity. Conversely, very large k values also hurt accuracy, as they introduce noisy or weakly relevant segments that dilute the useful temporal signal. Overall, the results reveal a clear "sweet spot" at moderate values (e.g., $k = 3$), striking the right balance between informativeness and noise. This highlights that retrieval precision and contextual diversity must be jointly optimized for effective temporal reasoning.

3. **Vary number of iterations in approach (1, 2, 3, 5, 8) Refinement Iteration Reduction:** The number of refinement iterations will be reduced to one in order to assess whether the self-refinement process of the LLM genuinely contributes to better performance compared to a single-pass generation.

   As shown in Figure 5, the rapid improvement in the early refinement steps indicates that

the model corrects global trajectory inconsistencies first, with later iterations offering diminishing returns as predictions approach the limit imposed by model bias and signal noise. Longer horizons benefit more from refinement, suggesting that iterative prompting increases the model's effective reasoning depth rather than simply rephrasing the forecast.

From an efficiency perspective, runtime increases near-linearly with iteration count, reflecting the compounded generation cost in decoder-only LLM architectures. The balance between accuracy and inference cost favors a small number of refinement passes, which achieves most of the performance gain while maintaining competitive efficiency.

4. **Multivariate Addition:** Multivariate prompting yields consistent performance gains, particularly at medium–long horizons, demonstrating that cross-variable temporal signals help reduce forecast ambiguity beyond univariate context alone. Notably, our multivariate variant surpasses all or nearly all baseline models across multiple datasets, highlighting its effectiveness in exploiting cross-variable structure. However, its impact varies by dataset, critically depending on the relevance between auxiliary variables and the target series; when correlation is weak or noisy, multivariate inputs may provide limited or even distracting guidance.

5. **Alternative LLMs:** We evaluate RAG-POT using two different LLM backbones: DeepSeek-R1-0528 and GPT-4.1-mini. While DeepSeek achieves competitive performance and surpasses GPT-4.1-mini on certain datasets, its results exhibit noticeable variance across settings, indicating weaker stability and generalization in cross-domain forecasting. In contrast, GPT-4.1-mini delivers more consistent gains across different variant data, suggesting a stronger alignment with the refinement-driven reasoning paradigm of our framework. Furthermore, DeepSeek incurs substantially higher inference latency due to longer decoding traces, reducing its suitability for real-time or resource-constrained applications.

Ideally, we expect that removing any of these components will lead to a degradation in performance. By observing how much each ablation affects forecasting accuracy, we aim to identify which component contributes the most to the overall improvement. The insights from this analysis will guide our future work, where we plan to refine or redesign the components that show relatively smaller performance gains.

F Prompts

- Memory Bank Agent Prompt: The Memory Bank Agent is designed to extract human-interpretable summaries from historical time-series segments. Instead of storing raw windows, which are expensive to search and difficult for LLMs to reason over, the agent converts each segment into a semantic "shape descriptor" that captures trend direction, local volatility, oscillatory behavior, and characteristic patterns. These summaries form a lightweight memory repository that supports efficient analogy-based forecasting.

- Forecaster Agent Prompt: The forecaster Agent prompt retrieved analogous pattern from the Memory Bank, and explicit forecasting instructions refined over previous iterations. The Agent is instructed to synthesize these elements by following forecasting rules, such as maintaining shape consistency, avoiding unrealistic numeric drift, and aligning with the analog futures description without copying them.

- Refiner Agent Prompt: The Refiner Agent evaluates the Forecaster's recent prediction and produces targeted corrections that update the Forecasting Instructions. In the prompt, the agent compares the ASCII overlay and numeric error statistics to reveal the qualitative shape mismatches and feed back into the reasoning process.

Memory Bank Agent Prompt

```
You are an expert time-series analyst.
You will be given:
- A recent normalized time-series segment (input)
- A following future segment (outcome)
Your task:
1) Analyze the input segment's overall trend, volatility, and pattern.
2) Describe how the future segment behaves relative to the input (continues,
reverses, oscillates).
3) Output a concise JSON object with the following fields ONLY:
{
   "desc":  "...short natural language summary (1-2 sentences), describe in
terms of trend, volatility, pattern, oscillation...",
}
IMPORTANT:
- Return ONLY valid JSON, no extra commentary.
- The numeric arrays are normalized (mean0, std1).
============================================================
INPUT SEGMENT (length = {len(seg)})
============================================================
{seg.tolist()}
============================================================
FUTURE SEGMENT (length = {len(fut)})
============================================================
{fut.tolist()}
```

Figure 9: *Prompt used for the Memory Bank agent in time-series forecasting.*

Forecaster Agent Prompt

```
You are the Forecaster Agent in time-series forecasting system.  Your objective
is to generate the next {horizon} numeric values.  Output ONLY a JSON array of
{horizon} numbers and nothing else.
==================================================
CONTEXT WINDOW (most recent {len(ctx)} points)
==================================================
{ctx.tolist()}
==================================================
FORECASTING INSTRUCTIONS (from Refiner)
==================================================
{instructions}

These instructions are high priority.  They describe how your reasoning should
adjust based on previous forecasting errors.  Follow them carefully.
==================================================
RETRIEVED ANALOG PATTERNS
==================================================
You are given several past time-series segments that resemble the current
pattern.  Each memory includes a concise shape summary and a short preview of
its future.

Use these analogs to guide trend direction, short-term fluctuations, and turning
points.  DO NOT copy the numbers; use them to form analogical intuition.

For each memory #{idx}:

  • description:  {mem['desc']}

  • future_preview:  {future_vec.tolist()}

==================================================
MANDATORY FORECASTING RULES
==================================================

  • Preserve key shape properties:  short-term volatility seen in the last
    context points, local curvature and turning-point structure, slope direction
    consistency unless strong reversal cues exist.

  • Align with analog futures:  If multiple memories show similar early-future
    behavior, your forecast should reflect a compatible pattern.

  • Maintain numeric plausibility:  Forecast values must remain within
    reasonable deviation of the context mean/range, and avoid unrealistic
    monotonic drift unless strongly justified.

==================================================
FINAL OUTPUT FORMAT (STRICT)
==================================================
Return ONLY: [ v1, v2, v3, ..., v{horizon} ]
```

Figure 10: *Prompt used for the Forecaster agent in time-series forecasting.*

Refiner Agent Prompt

```
You are the Refiner Agent in forecasting system.  Your task is to improve the
forecasting instructions used by the Forecaster.
You are given:

   • the ground-truth refinement window

   • the Forecaster's prediction for this window

   • the instruction history

==================================================
CURRENT INSTRUCTIONS
==================================================
{current_instructions}
==================================================
GROUND TRUTH vs PREDICTION
==================================================
TRUE (first 32):  {y_true.tolist()}
PRED (first 32):  {y_pred.tolist()}
Current MAE: {mae_curr:.3f}
==================================================
ASCII SHAPE OVERLAY
==================================================
{ascii_plot_dual(y_true, y_pred)}
Use the ASCII overlay to detect:

   • missing or exaggerated volatility

   • oversmoothing

   • missed turning points

   • slope or amplitude mismatch

Error statistics:

   • std_true = {np.std(y_true):.3f}

   • std_pred = {np.std(y_pred):.3f}

==================================================
YOUR TASK
==================================================

   • Identify specific shape or numeric issues in the prediction.  Reference
     mismatches in curvature, volatility, turning points, slope, or local
     oscillation amplitude.

   • Provide 1-3 concise and actionable Learnings that will change how the
     Forecaster should reason.  These must be specific and never generic (avoid:
     adjust trend, calibrate baseline, improve smoothing).

   • If no further meaningful refinement is possible, return Done:  True.

==================================================
OUTPUT FORMAT (STRICT)
==================================================
Learnings:  <your short actionable improvements>
Done:  <True or False>
```

Figure 11: *Prompt used for the Refiner agent in time-series forecasting.*