MEMTIME: HIPPOCAMPUS-INSPIRED AGENTS FOR ADAPTIVE TIME SERIES FORECASTING WITH EVOLVING MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Pre-trained time-series foundation models have recently achieved remarkable zero-shot forecasting performance across diverse domains, providing a powerful paradigm for time-series prediction. However, most TSFMs remain fixed after pretraining, limiting their ability to incorporate new experiences as data distributions evolve. Inspired by hippocampal memory mechanisms, which enable rapid experience encoding, selective recall, and long-term consolidation, we investigate how dynamic memory processes can endow time-series foundation models with continual adaptation capabilities. Realizing such adaptation requires a mechanism that can effectively identify, incorporate, and evolve experiences throughout streaming forecasting. Motivated by these observations, we introduce MemTime, an autonomous self-evolving time-series agent framework that enables foundation models to adapt through dynamic memory augmentation without modifying their underlying parameters. Specifically, MemTime introduces a self-evolving memory mechanism that enables foundation models to acquire, utilize, and consolidate deployment-time experiences. It incorporates a reinforcement learning-based Cognitive Sentinel, which captures environmental changes and adaptively determines when memory retrieval is required. The Mental Retracer transforms retrieved experiences into memory-based correction signals, which adjust the forecasting process without updating the model parameters. To support long-term adaptation, a memory metabolism mechanism periodically consolidates the memory buffer by removing outdated experiences while preserving valuable knowledge. Extensive experiments on multiple non-stationary forecasting benchmarks demonstrate that MemTime consistently outperforms state-of-the-art adaptation and external-memory baselines, achieving improvements of up to 26.03% while maintaining cross-domain generalization. The code is publicly available at `https://anonymous.4open.science/r/MemTime-050A/`.

1 INTRODUCTION

IoT sensors, smart grids, and mobile telemetry generate massive time-series streams (Ogbodo et al., 2022; Liu et al., 2024b). Accurate modeling is critical for industrial fault diagnosis, financial risk, traffic control, and environmental monitoring (Hsu & Liu, 2021; Bolton & Hand, 2002; Liu et al., 2025b; Zhang et al., 2025). Pre-trained time-series foundation models (TSFMs) have emerged as a promising solution for scalable forecasting, owing to their zero-shot generalization capabilities enabled by large-scale pre-training (Zhou et al., 2023; Ansari et al., 2024; Das et al., 2023). By learning shared temporal patterns through large-scale multi-domain pre-training (Radford et al., 2019; Lu et al., 2022), they can generalize to unseen forecasting tasks without task-specific retraining.

However, the zero-shot capabilities of TSFMs do not guarantee effective adaptation in real-world streaming environments, where data distributions continuously evolve due to concept drifts (Liu et al., 2024b; Gama et al., 2014). When deployed with fixed parameters, frozen zero-shot TSFMs (Woo et al., 2024) cannot incorporate newly observed patterns or accumulate deployment-time experiences, making their pretrained knowledge outdated under distribution shifts (Liu & Zhang, 2025). Existing studies have explored test-time adaptation and memory-based mechanisms to improve model adaptability. Test-time adaptation methods (Zhang et al., 2023) adjust models using

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

newly observed data, but updating large-scale backbones introduces substantial computational costs and risks catastrophic forgetting. Memory-based retrieval approaches (Hu et al., 2025) avoid parameter updates by storing historical observations, yet their similarity-based retrieval may fail when current patterns deviate from past experiences. These limitations suggest the need for a mechanism that can incorporate new experiences while selectively utilizing and maintaining relevant knowledge over time. In biological systems, the hippocampal memory system enables such adaptive learning through experience encoding, selective recall, and memory consolidation (McClelland et al., 1995). Inspired by these mechanisms, incorporating hippocampal-like memory principles into time-series forecasting provides a potential pathway toward continual adaptation in streaming environments.

However, realizing such memory-driven adaptation for time-series foundation models still faces several challenges. **First, historical experiences may become unreliable as data distributions evolve over time.** When the current environment differs from previously observed patterns, directly reusing historical experiences may introduce conflicting information and degrade forecasting performance. Therefore, effective streaming adaptation requires a mechanism to selectively utilize relevant memories. **Second, newly observed experiences need to be incorporated without compromising the stability of pretrained foundation models.** Although test-time adaptation methods enable models to adjust to distribution shifts, continuously updating large-scale backbones introduces considerable computational costs and risks catastrophic forgetting. Therefore, adaptation mechanisms should leverage new experiences while preserving the pretrained knowledge. **Third, accumulated experiences need to be continuously evolved for long-term adaptation.** As streaming forecasting proceeds, some memories become outdated while others remain valuable for future adaptation. Simply storing all experiences leads to redundant memory growth, whereas indiscriminate removal may discard useful patterns. Therefore, an adaptive memory mechanism is required to maintain and refine informative experiences over time.

To enable continuous adaptation of time-series foundation models in streaming environments, we introduce MemTime, a memory-augmented adaptation framework inspired by the self-evolving properties of biological memory. Instead of updating backbone parameters, MemTime enables frozen foundation models to continuously incorporate new experiences while preserving previously acquired knowledge through a self-evolving memory loop. Specifically, MemTime addresses the uncertainty of experience utilization through the Cognitive Sentinel, which monitors the current forecasting state and determines whether historical experiences are beneficial for correcting the ongoing trajectory. Driven by reinforcement learning, the Sentinel learns an adaptive correction policy instead of relying on fixed memory access rules, enabling selective experience utilization under evolving data distributions. Beyond deciding when to use memory, MemTime introduces the Mental Retracer to incorporate new experiences without modifying pretrained parameters. The Retracer retrieves relevant memory traces and transforms them into residual velocity corrections, which are integrated into the frozen forecasting dynamics to refine prediction trajectories. This design allows newly acquired experiences to improve future forecasts while maintaining the stability of pretrained knowledge. Furthermore, MemTime maintains the accumulated experiences through the Memory Metabolism mechanism. By evaluating the evolving utility of stored memories, the mechanism reinforces informative experiences and removes outdated ones, preventing uncontrolled memory growth while preserving useful knowledge for future adaptation. Together, these components enable frozen time-series foundation models to continuously adapt under streaming environments.

The major contributions are summarized as follows:

- We propose MemTime, a memory-augmented time-series agent framework that enables frozen time-series foundation models to continuously adapt in streaming environments without updating backbone parameters.
- We introduce an adaptive memory utilization mechanism consisting of a reinforcement learning-based Cognitive Sentinel and a generative Mental Retracer, which enables selective experience invocation and parameter-efficient forecasting adaptation under distribution shifts.
- We develop an asynchronous Memory Metabolism mechanism for long-term streaming deployment, maintaining informative experiences while controlling memory growth.
- Extensive evaluations across seven real-world streaming benchmarks demonstrate that MemTime consistently outperforms state-of-the-art foundation models and test-time adaptation baselines, achieving improvements of up to 26.03%.

2 Related Work

Time Series Forecasting. Traditional forecasting methods learn direct mappings from historical observations to future values through task-specific models (Oreshkin et al., 2020; Nie et al., 2023; Liu et al., 2024a; Wang et al., 2024). Recent time-series foundation models (TSFMs) leverage large-scale pretraining to achieve strong zero-shot and cross-domain forecasting capabilities (Ansari et al., 2024; Das et al., 2023; Woo et al., 2024; Jin et al., 2024; Liu et al., 2025b), but most remain frozen after deployment and cannot effectively incorporate newly observed experiences (Lee et al., 2025). Existing adaptation methods address non-stationarity through online adaptation, concept drift mitigation, latent context modeling, dynamic experts, and generative trajectory evolution (Hoang et al., 2024; Lee et al., 2025; Zhao et al., 2026; Zhang et al., 2026; Zhu et al., 2026; Liu et al., 2026). However, these approaches often rely on explicit optimization or predefined adaptation strategies, limiting their ability to accumulate and utilize long-term experiences. Recently, Agentic time series forecasting extends forecasting models with memory and reasoning capabilities through experience accumulation, memory retrieval, trajectory selection, and reflective refinement (Tao et al., 2026). Nevertheless, existing agentic time series forecasting methods mainly rely on external memory construction and iterative reasoning, leaving efficient internal adaptation of frozen forecasting models under continuous distribution shifts unexplored.

Hippocampus-Inspired Memory and Adaptation. Hippocampal-inspired memory models explore adaptive learning mechanisms based on biological memory systems. The complementary learning systems theory characterizes fast hippocampal encoding and slow cortical consolidation, inspiring models that regulate memory consolidation for improved generalization (McClelland et al., 1995; Sun et al., 2023). Recent studies further investigate memory recall as generative reconstruction through replay and biologically motivated nonlinear mechanisms (Spens & Burgess, 2024; Kim et al., 2023). Based on these principles, various computational memory architectures have been developed, including unified associative memories, sparse memories, energy-based models, and scalable adaptive memory mechanisms (Millidge et al., 2022; Hu et al., 2023; Hoover et al., 2023; Cabannes et al., 2024; Wang et al., 2026; Shafiei Kafraj et al., 2026). In time-series modeling, STanHop introduces hippocampal-inspired memory for rapid adaptation to sudden events (Wu et al., 2024). Meanwhile, continual learning methods explore biologically motivated strategies such as active forgetting, neuron renewal, and slow–fast latent adaptation (Wang et al., 2023; Dohare et al., 2024; Farias & Jozefiak, 2025; Missel & Wang, 2025). However, existing approaches mainly focus on memory storage, retrieval, or parameter adaptation, leaving adaptive memory utilization and experience-driven correction for frozen foundation models less explored.

3 Methodology

MemTime is built upon a frozen time-series foundation model and introduces a memory-driven adaptation loop for streaming forecasting (Kim et al., 2025), as illustrated in Figure 1. The framework consists of three stages: Cognitive Sentinel, Mental Retracer, and Memory Metabolism. Specifically, the Cognitive Sentinel determines when memory intervention is required, the Mental Retracer transforms relevant experiences into residual corrections for trajectory refinement, and the Memory Metabolism module updates and consolidates the memory pool based on accumulated experiences. The following sections introduce these components in detail.

3.1 Cognitive Sentinel: Adaptive Memory Invocation

The Cognitive Sentinel adaptively regulates the activation of memory-based correction during streaming forecasting. It evaluates the reliability of the current forecasting trajectory and determines when the Mental Retracer should be activated for further refinement.

Invocation mechanism The Sentinel receives diagnostic signals extracted from the frozen forecasting dynamics as its input state:

$$s_m = [\tilde{\mathcal{U}}_t, \tilde{E}_t^{\text{cons}}, \widetilde{\|J_t\|}_2, t/K, \tilde{B}_t], \tag{1}$$

where $\mathcal{U}_t$ denotes the uncertainty of the velocity field, E_t^{cons} represents the multi-step consistency error, $\|J_t\|_2$ is the Jacobian spectral norm of the velocity field, t/K indicates the current ODE

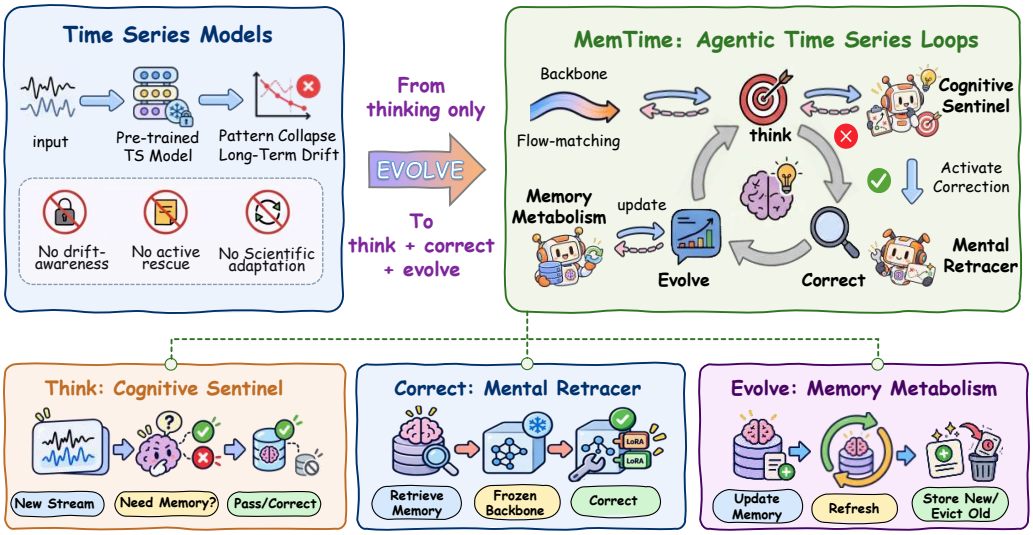


Figure 1: MemTime framework overview.

progress, and B_t denotes the remaining computational budget (Chen et al., 2018; Finlay et al., 2020). Based on the diagnostic state s_m, the Sentinel parameterizes the invocation decision using a lightweight multilayer perceptron:

$$\pi_\psi(a_m|s_m) = \text{Softmax}\left(\frac{W_2\sigma(W_1 s_m + b_1) + b_2}{\tau_a}\right) \in \Delta^1, \tag{2}$$

where W_1, W_2, b_1, b_2 denote the learnable parameters of the Sentinel policy, $\sigma(\cdot)$ is the non-linear activation function, τ_a controls the sharpness of the decision distribution, and $a_m \in \{\texttt{PASS}, \texttt{CORRECT}\}$ denotes the memory invocation action. The PASS action continues the original flow-matching trajectory without modification, whereas the CORRECT action activates the Mental Retracer to generate memory-based residual corrections. To reduce inference overhead, the Sentinel is evaluated only under potential instability conditions, i.e., $\mathcal{U}_t > \tau_U$, $E_t^{\text{cons}} > \tau_E$, $\|J_t\|_2 > \tau_J$.

Sentinel Policy Optimization. The Sentinel policy is initialized using supervision signals derived from the discrepancy between the frozen generator and target dynamics. Specifically, the velocity discrepancy is computed as $e_t = u_t^{\text{true}} - u_t^\theta$, where u_t^{true} and u_t^θ denote the target velocity and the velocity predicted by the frozen generator, respectively. Together with the numerical integration error E_{num}, the discrepancy is used to construct an invocation target:

$$a_m^* = \begin{cases} \texttt{CORRECT}, & \|e_t\| > \tau_{\text{model}} \text{ or } E_{\text{num}} > \tau_{\text{num}}, \\ \texttt{PASS}, & \text{otherwise.} \end{cases} \tag{3}$$

The Sentinel is initialized by minimizing the supervised objective:

$$\mathcal{L}_{\text{BC}}(\psi) = -\mathbb{E}_{s_m}[\log \pi_\psi(a_m^*|s_m)]. \tag{4}$$

To further optimize the trade-off between forecasting accuracy and adaptation cost, we further refine the Sentinel policy with GRPO while keeping the backbone and Retracer fixed. The reward function considers forecasting accuracy, memory activation sparsity, and computational overhead:

$$R = -\|\mathbf{y}_i - \hat{\mathbf{y}}_i^{(K)}\|^2 - \lambda \sum_m \max(0, d_m - \bar{d}) - \beta \cdot \text{NFE}, \tag{5}$$

where d_m is the correction indicator at step m, $\bar{d}$ is the reference activation rate, and NFE denotes the number of function evaluations. The Sentinel policy is optimized with the GRPO objective:

$$\nabla_\psi \mathcal{J} = \mathbb{E}\left[\sum_k A(\tau_k)\nabla_\psi \log \pi_\psi(\tau_k)\right] - \gamma \nabla_\psi \text{KL}(\pi_\psi || \pi_{\text{ref}}), \tag{6}$$

where τ_k denotes the k-th sampled invocation trajectory, and $A(\tau_k)$ is the group-relative advantage calculated from the corresponding rewards in GRPO.

3.2 MENTAL RETRACER: MEMORY-DRIVEN TRAJECTORY CORRECTION

The Mental Retracer utilizes accumulated experiences to refine the current forecasting trajectory without modifying the pretrained backbone. It retrieves relevant memory traces and transforms them into residual corrections for adapting the frozen forecasting dynamics.

Memory retrieval and trajectory correction The Retracer maintains a memory pool of low-rank residual experts attached to the frozen Flow-Matching network for trajectory correction. Specifically, the memory pool is defined as:

$$\mathcal{M} = \{(\mu_k, \mathrm{LoRA}_k, U_k, \tau_k, s_k, c_k, a_k)\}_{k=1}^{K}. \tag{7}$$

Each memory trace consists of a retrieval prototype $\mu_k \in \mathbb{R}^d$, a low-rank residual velocity expert LoRA_k, a utility score U_k, the latest activation time τ_k, and lifecycle variables (s_k, c_k, a_k) maintained by the Memory Metabolism module. Given the current forecasting state $\mathbf{y}_t$, ODE integration time t, and backbone representation $\mathbf{h}_i$, the Retracer constructs a query representation $q_t = \mathrm{MLP}_q(\mathbf{y}_t, t, \mathbf{h}_i)$. The relevance between the current state and each memory trace is measured by the similarity $\rho_{t,k} = \mathrm{sim}(q_t, \mu_k)$, where $\rho_{t,k}$ denotes the similarity score between the query representation and the prototype of the k-th memory trace. These similarity scores are normalized into soft retrieval weights:

$$\alpha_{t,k} = \frac{\exp(\rho_{t,k}/\tau)}{\sum_{j=1}^{K} \exp(\rho_{t,j}/\tau)}, \tag{8}$$

where $\alpha_{t,k}$ represents the contribution weight of the k-th memory trace and τ is the temperature coefficient. The retrieved residual experts are then aggregated according to these weights to generate the memory-induced velocity correction:

$$\Delta v_\phi(\mathbf{y}_t, t, \mathbf{h}_i) = \sum_{k=1}^{K} \alpha_{t,k} \cdot \mathrm{LoRA}_k(\mathbf{y}_t, t, \mathbf{h}_i), \tag{9}$$

where Δv_ϕ denotes the residual velocity correction generated by the Retracer through memory retrieval. When memory intervention is activated, the generated residual field is injected into the frozen flow-matching dynamics, allowing deployment-time experiences to refine the forecasting trajectory without modifying the pretrained backbone.

Retracer Optimization. The Mental Retracer is optimized to learn residual corrections that compensate for the discrepancy between the frozen generator and the target forecasting dynamics. We first train a residual generator:

$$\Delta v_\phi^{\text{gen}}(\mathbf{y}_t, t, \mathbf{h}_i) = \mathrm{LoRA}_{\text{gen}}(\mathbf{y}_t, t, \mathbf{h}_i), \tag{10}$$

where $\Delta v_\phi^{\text{gen}}$ denotes the generated residual velocity field, $\mathbf{y}_t$ represents the intermediate state at ODE time t, and $\mathbf{h}_i$ denotes the backbone representation of the i-th input patch. During training, the target velocity along the conditional optimal transport path is available as $u_t^{\text{true}} = \mathbf{y}_i - \mathbf{y}_i^{(0)}$. We sample intermediate states from the reference path:

$$\mathbf{y}_t^* = (1-t)\mathbf{y}_i^{(0)} + t\mathbf{y}_i, \tag{11}$$

and optimize the residual generator by minimizing:

$$\mathcal{L}_{\text{gen}}(\phi) = \mathbb{E}_{t, \mathbf{y}_i^{(0)}, \mathbf{y}_i} \left[\left\| \mathrm{LoRA}_{\text{gen}}(\mathbf{y}_t^*, t, \mathbf{h}_i) - \left(u_t^{\text{true}} - u_t^\theta(\mathbf{y}_t^* | \mathbf{h}_i)\right) \right\|_2^2 \right]. \tag{12}$$

This objective enables the residual generator to capture the correction between the frozen dynamics and the target trajectory. After obtaining the general residual correction capability, we construct the memory pool from residual patterns collected on validation trajectories. Similar residual patterns are grouped into clusters, and each cluster C_k initializes one memory trace. The corresponding residual expert is optimized as:

$$\mathcal{L}_{\text{trace}}(\phi_k) = \mathbb{E}_{(\mathbf{y}_t, t, \mathbf{h}_i) \in C_k} \left[\left\| \mathrm{LoRA}_k(\mathbf{y}_t, t, \mathbf{h}_i) - \left(u_t^{\text{true}} - u_t^\theta(\mathbf{y}_t | \mathbf{h}_i)\right) \right\|_2^2 \right]. \tag{13}$$

Meanwhile, the query network MLP_q is optimized to align forecasting states with their corresponding memory prototypes through an InfoNCE objective:

$$\mathcal{L}_{\text{query}} = -\mathbb{E}_{(q_t, k^+)} \log \frac{\exp(\mathrm{sim}(q_t, \mu_{k^+})/\tau)}{\sum_j \exp(\mathrm{sim}(q_t, \mu_j)/\tau)}, \tag{14}$$

where μ_k denotes the memory prototype of the k-th trace, and τ is the temperature parameter.

3.3 MEMORY METABOLISM: ADAPTIVE MEMORY MAINTENANCE

The Memory Metabolism module maintains memory traces during long-term streaming adaptation by updating their utility, consolidating related experiences, and removing obsolete traces. Each memory trace is associated with a utility score U_k, which reflects its contribution to forecasting performance. After a memory trace is activated, its utility is updated according to the forecasting improvement induced by the corresponding correction:

$$U_k \leftarrow (1-\eta)U_k + \eta r_k, \quad U_k \leftarrow U_k \cdot \exp(-\gamma(t-\tau_k)), \tag{15}$$

where $r_k = \mathcal{L}_{\text{before}} - \mathcal{L}_{\text{after}}$ denotes the short-term reduction in forecasting error after applying the k-th memory trace, η is the exponential moving average coefficient, τ_k records the latest activation time of the trace, and γ controls the temporal decay rate.

Each memory trace follows four lifecycle states: `NEW`, `ACTIVE`, `DORMANT`, and `PRUNED`. Newly created traces are temporarily protected, useful traces remain active for retrieval, and inactive traces enter dormant status for potential reuse before eventual pruning. Under a fixed capacity $K_{\max}$, the Metabolism module removes obsolete traces first and consolidates similar active traces when necessary; otherwise, the least useful trace is moved to the dormant state. Detailed method descriptions are provided in Appendix B.

3.4 MEMORY-DRIVEN ADAPTATION LOOP

During streaming forecasting, MemTime performs iterative memory adaptation as new observations arrive. Given the current forecasting state $\mathbf{y}_t$ and backbone representation $\mathbf{h}_i$, the Cognitive Sentinel first extracts the diagnostic state s_m and determines whether memory intervention is required through a_m. When $a_m = \texttt{CORRECT}$, the Mental Retracer retrieves relevant memory traces from $\mathcal{M}$ according to the retrieval weights $\alpha_{t,k}$ and generates residual corrections $\Delta v_\phi(\mathbf{y}_t, t, \mathbf{h}_i)$ to refine the forecasting trajectory. Otherwise, the original flow-matching trajectory is preserved without modification. After adaptation, the Memory Metabolism module updates the utility of activated traces and consolidates the memory pool according to accumulated experiences. This closed-loop process enables MemTime to continuously acquire, utilize, and maintain experiences under evolving streaming environments. The complete adaptation procedure is summarized in the Appendix C.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Evaluation and Benchmarks. Following standard stream forecasting protocols in time-series foundation models models process test streams sequentially (Liu et al., 2025b). Given a context window $L \in \{512\}$, the objective is to forecast future trajectories across horizons $P \in \{96, 192, 336, 720\}$, evaluated by MSE and MAE. To rigorously evaluate adaptation under varying environmental dynamics, we conduct experiments across seven standard real-world benchmarks spanning two regimes: (1) *Severe Non-Stationary Regime*: *Exchange*, characterized by rapid and continuous concept drifts; and (2) *Periodic and Environmental Regimes*: *Weather*, *ECL*, and the complete Electricity Transformer Temperature suite (*ETTm1*, *ETTm2*, *ETTh1*, *ETTh2*) (Liu et al., 2024a), featuring strong periodicities to test whether MemTime selectively suppresses unnecessary memory invocation.

Baselines. To evaluate the zero-shot forecasting and adaptation capability of MemTime, we compare against SOTA pre-trained time-series models, including *Sundial* (Liu et al., 2025b), *Timer-XL* (Liu et al., 2025a), *Moirai* (Woo et al., 2024), *Chronos* (Ansari et al., 2024), and *UniTS* (Gao et al., 2024). All baseline foundation models are evaluated under a zero-shot forecasting protocol without parameter updates. Detailed experimental settings are provided in the Appendix A.

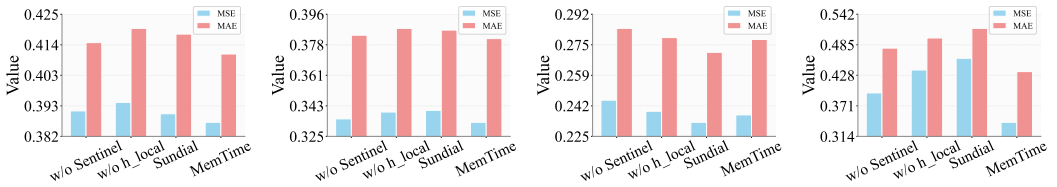
4.2 MAIN RESULTS

To evaluate the forecasting efficacy and adaptation capability of MemTime under streaming dynamics, we conduct comprehensive evaluations across seven standard real-world benchmarks. As shown in Table 1, MemTime achieves consistent improvements on the Exchange benchmark, which exhibits severe non-stationary distribution shifts. Static foundation models and existing test-time

Table 1: Overall performance comparison for time-series forecasting. Best results are shown in **bold**, and second-best results are underlined.

Dataset		Memtime		Sundial		Timer-XL		Moirai_Base		Moirai_Large		Chronos_Base		Chronos_Large		UNITS	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	0.293	**0.340**	**0.292**	0.342	0.317	0.356	0.363	0.356	0.380	0.361	0.454	0.408	0.457	0.403	0.407	0.434
	192	**0.328**	**0.371**	0.337	0.376	0.358	0.381	0.388	0.375	0.412	0.383	0.567	0.477	0.530	0.450	0.442	0.456
	336	**0.368**	**0.404**	0.370	0.401	0.386	0.401	0.416	0.392	0.436	0.400	0.662	0.525	0.577	0.481	0.470	0.474
	720	**0.407**	**0.425**	0.418	0.433	0.430	0.431	0.460	0.418	0.462	0.420	0.900	0.591	0.660	0.526	0.518	0.513
	Avg	**0.349**	**0.385**	0.354	0.388	0.373	0.392	0.406	0.385	0.422	0.391	0.645	0.500	0.555	0.465	0.459	0.469
ETTm2	96	**0.174**	**0.260**	0.178	**0.260**	0.189	0.277	0.205	0.273	0.211	0.274	0.199	0.274	0.197	0.271	0.336	0.400
	192	**0.229**	**0.291**	0.235	0.304	0.241	0.315	0.275	0.316	0.281	0.318	0.261	0.322	0.254	0.314	0.430	0.465
	336	**0.270**	**0.332**	0.287	0.342	0.286	0.348	0.329	0.350	0.341	0.355	0.326	0.366	0.313	0.353	0.495	0.512
	720	**0.336**	**0.370**	0.360	0.390	0.375	0.402	0.437	0.411	0.485	0.428	0.455	0.439	0.416	0.415	0.652	0.598
	Avg	**0.252**	**0.313**	0.265	0.324	0.273	0.336	0.311	0.337	0.329	0.343	0.310	0.350	0.295	0.338	0.478	0.494
ETTh1	96	**0.335**	**0.375**	0.341	0.381	0.369	0.391	0.376	0.392	0.381	0.388	0.440	0.393	0.441	0.390	0.416	0.390
	192	**0.379**	**0.401**	0.381	0.408	0.405	0.413	0.412	0.413	0.434	0.415	0.492	0.426	0.502	0.524	0.452	0.412
	336	**0.408**	**0.417**	0.405	0.424	0.418	0.423	0.433	0.428	0.485	0.445	0.550	0.462	0.576	0.467	0.486	0.433
	720	**0.424**	**0.450**	0.433	0.458	0.441	0.447	0.444	0.444	0.611	0.510	0.882	0.591	0.835	0.583	0.542	0.468
	Avg	**0.387**	**0.411**	0.390	0.418	0.404	0.417	0.417	0.419	0.480	0.439	0.591	0.468	0.588	0.466	0.474	0.426
ETTh2	96	**0.267**	**0.323**	0.272	0.332	0.283	0.342	0.294	0.330	0.296	0.330	0.308	0.343	0.320	0.345	0.284	0.307
	192	**0.327**	**0.366**	0.329	0.374	0.340	0.379	0.365	0.375	0.361	0.371	0.384	0.392	0.406	0.399	0.341	0.350
	336	**0.354**	**0.392**	0.357	0.399	0.366	0.400	0.376	0.390	0.390	0.390	0.429	0.430	0.492	0.453	0.398	0.389
	720	**0.386**	**0.448**	0.401	0.442	0.397	0.431	0.416	0.433	0.423	0.418	0.501	0.477	0.603	0.511	0.514	0.468
	Avg	**0.333**	**0.382**	0.340	0.387	0.347	0.388	0.362	0.382	0.367	0.377	0.405	0.410	0.455	0.427	0.384	0.379
ECL	96	0.148	0.230	**0.134**	**0.231**	0.141	0.237	0.160	0.250	0.153	0.241	0.154	0.231	0.152	0.229	0.167	0.260
	192	0.164	0.263	**0.154**	**0.251**	0.159	0.254	0.175	0.263	0.169	0.255	0.179	0.254	0.172	0.250	0.188	0.288
	336	**0.174**	**0.271**	0.182	0.276	0.177	0.272	0.187	0.277	0.187	0.273	0.214	0.284	0.203	0.276	0.211	0.315
	720	**0.215**	**0.307**	0.220	0.311	0.219	0.308	0.228	0.309	0.237	0.313	0.311	0.346	0.289	0.337	0.288	0.385
	Avg	0.175	**0.268**	**0.172**	0.267	0.174	0.278	0.187	0.274	0.186	0.270	0.214	0.278	0.204	0.273	0.214	0.312
Weather	96	0.160	0.223	**0.158**	**0.206**	0.171	0.225	0.220	0.217	0.199	0.211	0.203	0.238	0.194	0.235	0.184	0.235
	192	0.207	0.251	**0.205**	**0.253**	0.221	0.271	0.271	0.259	0.246	0.251	0.256	0.290	0.249	0.285	0.226	0.279
	336	**0.252**	**0.298**	0.254	0.299	0.274	0.311	0.286	0.297	0.274	0.291	0.314	0.336	0.302	0.327	0.274	0.312
	720	**0.315**	**0.336**	0.322	0.340	0.356	0.370	0.373	0.354	0.337	0.340	0.397	0.396	0.372	0.378	0.348	0.365
	Avg	**0.233**	0.277	0.235	**0.275**	0.256	0.294	0.287	0.281	0.264	0.273	0.292	0.315	0.279	0.306	0.258	0.297
Exchange	96	**0.147**	**0.205**	0.237	0.269	0.257	0.280	0.295	0.280	0.308	0.284	0.368	0.321	0.371	0.317	0.330	0.341
	192	**0.230**	**0.375**	0.362	0.415	0.385	0.421	0.417	0.414	0.443	0.423	0.609	0.526	0.569	0.497	0.475	0.503
	336	**0.422**	**0.509**	0.558	0.598	0.582	0.598	0.627	0.585	0.658	0.597	0.998	0.783	0.870	0.717	0.709	0.707
	720	**0.562**	**0.652**	0.683	0.782	0.703	0.778	0.752	0.755	0.755	0.759	1.471	1.067	1.078	0.950	0.846	0.926
	Avg	**0.340**	**0.435**	0.460	0.516	0.482	0.519	0.523	0.509	0.548	0.520	0.862	0.674	0.721	0.618	0.590	0.619

adaptation methods suffer from increasing errors as streaming evolves due to their limited experience utilization. In contrast, MemTime reduces MSE by 26.03% compared with the strongest baseline by selectively recalling relevant experiences and correcting trajectory deviations through memory-based adaptation. Across the four Electricity Transformer Temperature benchmarks, MemTime maintains robust performance across different forecasting horizons ($P \in \{96, 192, 336, 720\}$). For example, on ETTh1, MemTime reduces MSE by 4.21%, 7.19%, and 18.35% compared with Timer-XL, Moirai-Base, and UniTS, respectively. These results demonstrate that the self-evolving memory mechanism improves adaptation robustness across various temporal patterns.



(a) ETTh1 (b) ETTh2 (c) Weather (d) Exchange

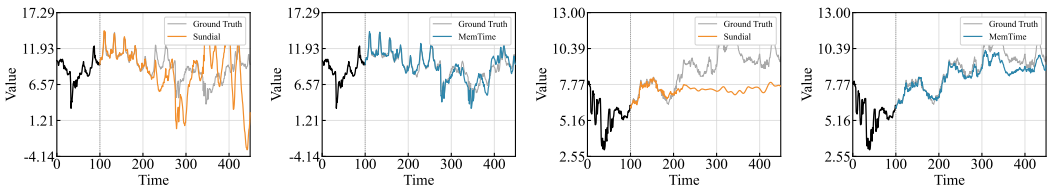
Figure 2: Performance of MemTime and its variants on four datasets.

4.3 Ablation Study

To evaluate the contribution of each core component, we remove the Cognitive Sentinel and Mental Retracer separately. As shown in Figure 2, removing either component consistently degrades performance across datasets. On the non-stationary Exchange set, removing the Sentinel and Retracer increases MAE by 0.73% and 0.40%, respectively, while maintaining similar MSE values.

This indicates that the Sentinel improves adaptation by selectively activating memory correction, whereas the Retracer reduces trajectory deviation through experience-based residual adjustment. On Weather, which exhibits strong periodic patterns, removing the Sentinel increases MAE by 1.16%, as unnecessary memory activation may introduce redundant corrections under stable regimes. On ETTh2, the full model reduces MAE by 1.74% compared with the Retracer-only variant despite similar MSE, indicating more consistent predictions through adaptive memory utilization.

4.4 Case Study



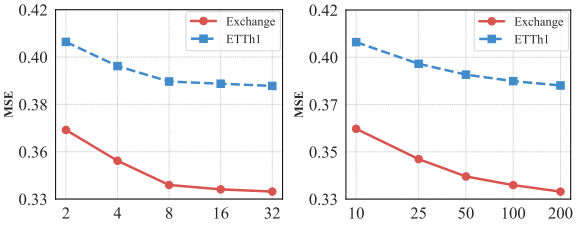
(a) Sundial on ETTh1. (b) MemTime on ETTh1. (c) Sundial on ETTm1. (d) MemTime on ETTm1.

Figure 3: Case Study.

To illustrate MemTime's advantage under non-stationary dynamics, Figure 3 contrasts the zero-shot backbone Sundial with MemTime on ETTh1 and ETTm1, which stress different failure modes: long-range trend drift and high-frequency fluctuations. As shown in Figure 3a, Sundial gradually deviates from the ground truth on ETTh1 and suffers severe trajectory divergence after step 100, with the error accumulating monotonically because the frozen backbone has no mechanism to compensate for the distribution shift. On ETTm1, Figure 3c shows a different failure: Sundial collapses into a nearly flat line and loses the high-frequency component of the signal. In contrast, Figure 3b and Figure 3d demonstrate that MemTime closely tracks the ground-truth trajectory on both datasets. On ETTh1, the divergence is suppressed even after step 100, indicating that the Sentinel policy triggers retrieval before the deviation becomes irreversible; on ETTm1, the flat-line collapse is replaced by a trajectory that preserves local oscillatory structure. This confirms that MemTime's adaptive memory mechanism does not uniformly rescale the prediction but targets the specific failure mode of each dataset, sustaining prediction fidelity under evolving temporal patterns.

4.5 Parameter Sensitivity Analysis

To further analyze the impact of memory configuration on MemTime, we vary the Mental Retracer rank $r \in \{2, 4, 8, 16, 32\}$ and memory capacity $K \in \{10, 25, 50, 100, 200\}$ on Exchange and ETTh1. As shown in Figure 4, increasing r from 2 to 8 reduces the Exchange MSE by 6.85%, as a larger rank provides stronger residual correction. Further increasing r brings limited gains due to redundant capacity and computation. Similarly, increasing K improves performance by covering more historical patterns, while gains saturate beyond $K = 50$ due to redundant memory traces. These results indicate that $r = 8$ achieves a good balance between correction capability and efficiency.

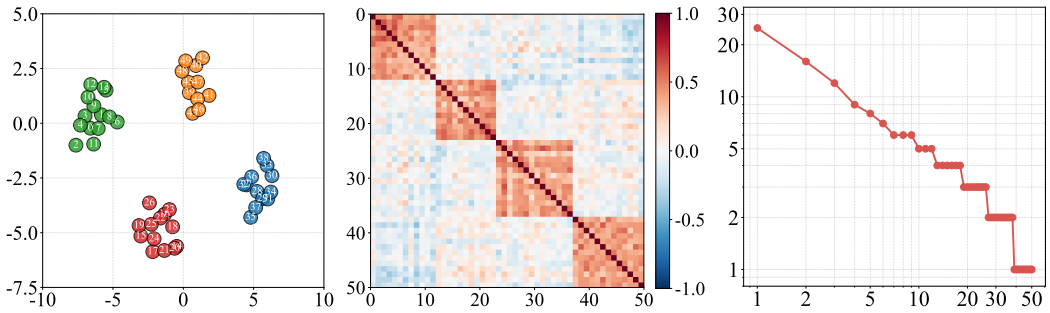


(a) Matrix Pool Rank r (b) Memory Capacity K

Figure 4: Parameter Sensitivity Analysis.

4.6 Efficiency and Memory Cost

We evaluate the inference latency and memory footprint of MemTime under streaming forecasting. As shown in Table 2, unlike gradient-based TTA methods such as CoTTA and TENT, MemTime keeps the backbone frozen and requires only forward computation. Its overhead mainly comes from the Cognitive Sentinel and low-rank parameter synthesis ($r = 8$), while selective gating avoids activating the Mental Retracer during stable periods. Consequently, MemTime introduces less



(a) t-SNE of Memory Prototypes (b) Cosine Similarity Matrix (c) Slot Usage Distribution

Figure 5: Memory tructure analysis.

than 5% latency overhead compared with the zero-shot backbone and achieves over $10\times$ speedup over gradient-based adaptation methods. MemTime also maintains stable memory consumption by encoding experiences in a fixed-size pool of K low-rank anchors with dimension $D \times r$ and asynchronously removing obsolete anchors through Memory Metabolism. Unlike approaches with buffers or activation caches that grow with the streaming horizon T, MemTime keeps VRAM usage stable for long streaming sequences ($T > 5{,}000$). Overall, these results demonstrate that MemTime achieves effective non-stationary adaptation with limited computational and memory overhead.

Table 2: Computational and memory efficiency profiling ($L = 512$, $P = 96$).

Method Category	**Model**	**Latency (ms/step)** $\downarrow$	**VRAM (MB)** $\downarrow$
Static Foundation	Sundial (Frozen)	24.5	1,450
Parameter-Updating TTA	TENT	285.4	4,280
	CoTTA	312.0	4,850
External Memory Buffer	ExpeL	$68.2 \sim \mathcal{O}(T)$	$2{,}100 \sim \mathcal{O}(T)$
Ours (Latent Generation)	**MemTime**	**25.6** ($< 5\%$ **overhead**)	**1,485 (Constant)**

4.7 Emergence of Hippocampal Memory

Figure 5 reveals that the memory pool learns structured representations rather than storing isolated historical patterns. For the $K{=}50$ memory prototypes μ_k, the t-SNE visualization in Figure 5a shows four distinct clusters, indicating that MemTime autonomously organizes experiences into different environmental regimes without supervision. This organization is further supported by the cosine similarity matrix in Figure 5b, where the block-diagonal structure demonstrates strong intra-regime consistency and limited interference across regimes, enabling more targeted memory retrieval. Moreover, Figure 5c shows a long-tailed slot usage distribution, suggesting that MemTime allocates more retrieval resources to frequent patterns while preserving less frequently activated memories for future adaptation. Together, these observations indicate that the self-evolving memory mechanism can discover, organize, and maintain diverse experiences under streaming dynamics.

More experimental results are provided in Appendix D.

5 Conclusion

This work presents MemTime, a hippocampus-inspired agent framework for adaptive time-series forecasting. To enable frozen foundation models to adapt without parameter updates, we introduce a Cognitive Sentinel for selective memory invocation, a Mental Retracer for memory-driven trajectory correction, and a Memory Metabolism mechanism for long-term consolidation. These components establish a self-evolving memory loop that allows forecasting models to acquire, utilize, and maintain experiences under continuously changing environments. Extensive experiments on seven real-world benchmarks show that MemTime achieves state-of-the-art accuracy with low latency and constant memory consumption in streaming scenarios.

AI USE STATEMENT

In this work, we used generative AI tools for improving the accuracy of writing by identifying and correcting grammatical issues. We have not used generative AI tools for research ideation, methodological development, experiments, or manuscript writing, and no other required disclosure tasks are applicable to this work. Additionally, no recommended disclosure tasks are applicable to this work. We have reviewed all AI-assisted work. The authors reviewed all AI-assisted writing and verified that the AI tools were used solely for grammar correction and writing accuracy; all research ideas, methodological developments, experiments, and the main body of the manuscript were independently conceived, conducted, and written by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series. *TMLR*, 2024.

Richard J Bolton and David J Hand. Statistical fraud detection: A review. *Statistical science*, 17(3): 235–255, 2002.

Vivien Cabannes, Elvis Dohmatob, and Alberto Bietti. Scaling laws for associative memories. In *ICLR*, 2024.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural ordinary differential equations. In *NeurIPS*, volume 31, pp. 6571–6583, 2018.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *ICML*, volume 235, pp. 10148–10167, 2023.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632:768–774, 2024.

Vivek Farias and Adam Jozefiak. Self-normalized resets for plasticity in continual learning. In *ICLR*, 2025.

Chris Finlay, Joern-Henrik Jacobsen, Levon Nurbekyan, and Adam Oberman. How to train your neural ODE: The world of Jacobian and kinetic regularization. In *PMLR*, volume 119, pp. 3154–3164, 2020.

João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *CSUR*, 46(4):1–37, 2014.

Shanghua Gao, Teddy Koker, Owen Queen, Thomas Hartvigsen, Theodoros Tsiligkaridis, and Marinka Zitnik. UniTS: A unified multi-task time series model. In *NeurIPS*, volume 37, 2024.

Trung-Hieu Hoang, Duc Minh Vo, and Minh N. Do. Persistent Test-Time Adaptation in Recurring Testing Scenarios. In *NeuraIPS*, 2024.

Benjamin Hoover, Yuchen Liang, Bao Pham, Rameswar Panda, Hendrik Strobelt, Duen Horng Chau, Mohammed J. Zaki, and Dmitry Krotov. Energy transformer. In *NeurIPS*, volume 36, pp. 27532–27559, 2023.

Chia-Yu Hsu and Wei-Chen Liu. Multiple time-series convolutional neural network for fault detection and diagnosis and empirical study in semiconductor manufacturing. *Journal of Intelligent Manufacturing*, 32(3):823–836, 2021.

Jerry Yao-Chieh Hu, Donglin Yang, Dennis Wu, Chenwei Xu, Bo-Yu Chen, and Han Liu. On sparse modern hopfield model. In *NeurIPS*, volume 36, 2023.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, et al. Memory in the age of ai agents: A survey. *arXiv preprint arXiv:2512.13564*, 2025.

Ming Jin Jin, Shiyu Wang, Lujia 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. In *ICLR*, 2024.

Dong-Kyum Kim, Jea Kwon, Meeyoung Cha, and C. Justin Lee. Transformer as a hippocampal memory consolidation model based on NMDAR-inspired nonlinearity. In *NeurIPS*, volume 36, 2023.

HyunGi Kim, Siwon Kim, Jisoo Mok, and Sungroh Yoon. Battling the non-stationarity in time series forecasting via test-time adaptation. *AAAI*, 39(17):17868–17876, 2025.

Thomas L Lee, William Toner, Rajkarn Singh, Artjom Joosen, and Martin Asenov. Lightweight online adaption for time series foundation model forecasts. In *ICML*, volume 267, pp. 33736–33764, 2025.

Aoyu Liu and Yaying Zhang. Crossst: An efficient pre-training framework for cross-district pattern generalization in urban spatio-temporal forecasting. In *ICDE*, pp. 2935–2948, 2025.

Yangyou Liu, Zezhi Shao, Xinyu Chen, Hu Chen, Fei Wang, and Yuankai Wu. PULSE: Generative phase evolution for non-stationary time series forecasting. In *ICML*, 2026.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted transformers are effective for time series forecasting. In *ICLR*, 2024a.

Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Generative pre-trained transformers are large time series models. *ICLR*, 2024b.

Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer-XL: Long-context transformers for unified time series forecasting. In *ICLR*, 2025a.

Yong Liu, Guo Qin, Ziquan Shi, Zewen Chen, Chen Yang, Xiaoyu Huang, Jianmin Wang, and Mingsheng Long. Sundial: A family of highly capable time series foundation models. *ICML*, pp. 39295–39317, 2025b.

Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Frozen pretrained transformers as universal computation engines. In *AAAI*, volume 36, pp. 7628–7636, 2022.

James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the failures of connectionist models of learning and memory. *Psychological review*, 102(3):419–457, 1995.

Beren Millidge, Tommaso Salvatori, Yuhang Song, Thomas Lukasiewicz, and Rafal Bogacz. Universal hopfield networks: A general framework for single-shot associative memory models. In *ICML*, volume 162, pp. 15561–15583, 2022.

Ryan Missel and Linwei Wang. Continual slow-and-fast adaptation of latent neural dynamics (CoS-Fan): Meta-learning what-how & when to adapt. In *ICLR*, 2025.

Yuqi Nie, Nam H Nguyen, Pattarawat Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *ICLR*, 2023.

Emmanuel Utochukwu Ogbodo, Adnan M Abu-Mahfouz, and Anish M Kurien. A survey on 5g and lpwan-iot for improved smart cities and remote area applications: From the aspect of architecture and security. *Sensors*, 22(16):6313, 2022.

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural Basis Expansion Analysis for Interpretable Time Series Forecasting. In *ICLR*, 2020.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Mohadeseh Shafiei Kafraj, Dmitry Krotov, and Peter Latham. A biologically plausible dense associative memory with exponential capacity. In *ICLR*, 2026.

Eleanor Spens and Neil Burgess. A generative model of memory construction and consolidation. *Nature Human Behaviour*, 8:526–543, 2024.

Weinan Sun, Madhu Advani, Nelson Spruston, Andrew Saxe, and James E. Fitzgerald. Organizing memories for generalization in complementary learning systems. *Nature Neuroscience*, 26:1438–1448, 2023.

Xiaoyu Tao, Mingyue Cheng, Ze Guo, Shuo Yu, Yaguo Liu, Qi Liu, and Shijin Wang. MemCast: Memory-driven time series forecasting with experience-conditioned reasoning. In *ICML*, 2026.

Liyuan Wang, Xingxing Zhang, Qian Li, Mingtian Zhang, Hang Su, Jun Zhu, and Yi Zhong. Incorporating neuro-inspired adaptability for continual learning in artificial intelligence. *Nature Machine Intelligence*, 5:1356–1368, 2023.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Zhang, and Jun Zhou. TimeMixer: Decomposable multiscale mixing for time series forecasting. In *ICLR*, 2024.

Shurong Wang, Yuqi Pan, Zhuoyang Shen, Meng Zhang, Hongwei Wang, and Guoqi Li. Adaptive hopfield network: Rethinking similarities in associative memory. In *ICLR*, 2026.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. *ICML*, pp. 53140–53164, 2024.

Yu-Hsuan Wu, Jerry Yao-Chieh Hu, Weijian Li, Bo-Yu Chen, and Han Liu. STanHop: Sparse tandem hopfield model for memory-enhanced time series prediction. In *ICLR*, 2024.

Fan Zhang, Shijun Chen, and Hua Wang. L-Drive: Beyond a single mapping—latent context drives time series forecasting. In *ICML*, 2026.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. Memgen: Weaving generative latent memory for self-evolving agents. *ICLR*, 2025.

Jiayun Zhang, Xiyuan Zhang, Xinyang Zhang, Dezhi Hong, Rajesh K Gupta, and Jingbo Shang. Navigating alignment for non-identical client class sets: A label name-anchored federated learning framework. In *SIGKDD*, pp. 3297–3308, 2023.

Zhiyuan Zhao, Haoxin Liu, and B. Aditya Prakash. Tackling time-series forecasting generalization via mitigating concept drift. In *ICLR*, 2026.

Tian Zhou, Peisong Niu, Liang Sun, Rong Jin, et al. One fits all: Power general time series analysis by pretrained lm. *NeurIPS*, 36:43322–43355, 2023.

Jiawen Zhu, Shuhan Liu, Di Weng, and Yingcai Wu. Dynamic TMoE: A drift-aware dynamic mixture of experts framework for non-stationary time series forecasting. In *ICML*, 2026.

A Notation and Causal Streaming Protocol

A.1 Separating observation time from generative time

We distinguish the index of a forecasting request from the internal integration time of the generative model. Let o_i denote the observation-time origin of request i, and let $X_t \in \mathbb{R}^C$ denote a multivariate observation at timestamp t. The context and target are

$$x_i = X_{o_i-L+1:o_i}, \qquad y_i = X_{o_i+1:o_i+P} \tag{16}$$

where L and P are the context length and prediction horizon, respectively. The paper uses $L = 512$ and $P \in \{96, 192, 336, 720\}$. The backbone produces a context representation $h_i = \mathrm{Encoder}_\theta(x_i)$. We write $z_i(s)$ for an intermediate generative state, where $s \in [0, 1]$ is flow time. The numerical solver evaluates the field at $s_0, \ldots, s_N$, and the terminal state gives the prediction $\hat{y}_i = z_i(1)$.

The symbol N denotes the number of integration steps; K_i denotes the number of stored memory traces; and $K_{\max}$ denotes the memory capacity. This separates three quantities denoted by overlapping symbols in the main-text formulation. If the backbone generates several patches autoregressively, the same notation applies within each patch, and computational costs are summed across all generated patches and all forecast samples.

Symbol	Meaning
i, o_i	Forecast request and its observation-time origin
s, n, N	Continuous flow time, solver-step index, and number of solver steps
$z_i(s), h_i$	Generative state and context representation
$d_{i,n} \in \{0, 1\}$	Actual correction action at step n
π_ψ	Cognitive Sentinel policy
$\alpha_{i,n,k}$	Retrieval weight of memory trace k
$\mathcal{M}_i, K_i, K_{\max}$	Memory snapshot, its size, and capacity
V_k, ℓ_k, b_k	Trace utility, last access time, and last utility-decay time
τ_a, τ_r	Policy and retrieval temperatures

We use V_k for memory utility to avoid confusion with the Sentinel's velocity-uncertainty diagnostic $U_{i,n}$. Access and aging timestamps refer to observation time or a documented stream counter, never to the internal ODE step index, which restarts for each forecast.

A.2 Information available at a forecast origin

A causal streaming evaluation requires the prediction at o_i to depend only on observations and feedback already available at that origin. If complete targets are released with an additional delay $\delta_i \geq 0$, the target of request i becomes available at

$$t_i^{\text{release}} = o_i + P + \delta_i.$$

Consequently, the prediction has the form

$$\hat{y}_i = F(x_i, \mathcal{M}_i, \xi_i), \qquad \mathcal{M}_i \in \mathcal{F}_{o_i},$$

where $\mathcal{F}_{o_i}$ is the information available by o_i and ξ_i contains model randomness. Supervised feedback from request j may enter $\mathcal{M}_i$ only if $t_j^{\text{release}} \leq o_i$.

We use the following causal streaming protocol. Forecasts are issued with stride $S = 1$. Timestamps are processed in nondecreasing order. A request at origin o_i may use only information in $\mathcal{F}_{o_i}$. Complete targets are released at $o_i + P$, i.e., $\delta_i = 0$, and we use full-target feedback only. At each timestamp, eligible feedback whose targets have been released is processed before issuing a new forecast at the same timestamp. Memory is reset only at dataset boundaries; it is not reset during the test stream. All adaptive baselines follow the same protocol.

For example, with unit forecast stride and $P = 96$, the full target of a request cannot update memory at the immediately following forecast origin. Offline access to the entire test file does not change this release schedule.

Evaluation errors may be computed after the stream finishes, but this retrospective scoring must not affect predictions already issued. Partial-target feedback is a separate protocol: it requires an explicitly masked loss over revealed entries and identical information access for all adaptive baselines.

A.3 Offline fitting, calibration, and testing

The backbone weights remain frozen. The Retracer experts, query network, and Sentinel require supervised offline training as described in Sections 3.1 and 3.2 of the main text. We use a chronological split into a training partition, a validation partition, and a test partition. The validation partition serves as the adaptation-fitting set for the auxiliary modules. A calibration subset carved from the training partition is used for hyperparameter selection and threshold calibration. The test partition forms the chronologically subsequent test stream. Training windows do not contain target timestamps belonging to the test partition. Context overlap with earlier observed history is permissible; future target overlap is not.

Memory experts are fitted from residual patterns collected on the validation trajectories. The validation partition therefore functions as adaptation-training data for the Retracer experts and the query network. We name it accordingly and use the separate calibration subset for hyperparameter selection. The frozen backbone does not by itself make the complete fitted system zero-shot with respect to those validation targets.

We evaluate under target-domain adaptation. The auxiliary modules use the target domain's validation partition, which is a permitted pre-test segment of the target domain. The test partition is excluded from fitting the auxiliary modules and from initializing the memory bank. We do not claim held-out-domain transfer.

B Model Training Details

B.1 Conditional residual target

Let $\mathbf{y}_i^{(0)}$ be sampled from the backbone's initialization distribution. On the linear conditional path used in the main text,

$$\mathbf{y}_t^\star = (1-t)\,\mathbf{y}_i^{(0)} + t\,\mathbf{y}_i, \qquad \frac{\mathrm{d}\mathbf{y}_t^\star}{\mathrm{d}t} = \mathbf{y}_i - \mathbf{y}_i^{(0)} = u_t^{\mathrm{true}}. \tag{17}$$

The corresponding samplewise residual target is

$$e_t^\star = u_t^{\mathrm{true}} - u_t^\theta(\mathbf{y}_t^\star|\mathbf{h}_i). \tag{18}$$

The general residual generator is trained by

$$\mathcal{L}_{\mathrm{gen}}(\phi) = \mathbb{E}\left[\left\|\Delta v_\phi^{\mathrm{gen}}(\mathbf{y}_t^\star, t, \mathbf{h}_i) - e_t^\star\right\|_2^2\right], \tag{19}$$

where $\Delta v_\phi^{\mathrm{gen}}(\mathbf{y}_t, t, \mathbf{h}_i) = \mathrm{LoRA}_{\mathrm{gen}}(\mathbf{y}_t, t, \mathbf{h}_i)$. This is algebraically equivalent to minimizing the squared conditional flow-matching error of $u_t^\theta + \Delta v_\phi^{\mathrm{gen}}$ while holding θ fixed. Under squared loss, an unrestricted population minimizer estimates the conditional mean of the samplewise residual given the model inputs. The target $u_t^{\mathrm{true}} = \mathbf{y}_i - \mathbf{y}_i^{(0)}$ serves only as a training target; it is not an observable ground-truth velocity at deployment. Neither this target nor the future endpoint $\mathbf{y}_i$ is an input to the online Sentinel or query network.

B.2 Corrected dynamics and fallback

Let $\mathcal{M}$ be the memory pool defined in the main text, and let $K = |\mathcal{M}|$. At ODE step m, the query and retrieval weights are

$$q_{t_m} = \mathrm{MLP}_q(\mathbf{y}_{t_m}, t_m, \mathbf{h}_i), \qquad \alpha_{t_m,k} = \frac{\exp(\mathrm{sim}(q_{t_m}, \mu_k)/\tau)}{\sum_{j=1}^K \exp(\mathrm{sim}(q_{t_m}, \mu_j)/\tau)}. \tag{20}$$

The memory correction and the resulting field are

$$\begin{aligned} \Delta v_\phi(\mathbf{y}_{t_m}, t_m, \mathbf{h}_i) &= \sum_{k=1}^K \alpha_{t_m,k}\,\mathrm{LoRA}_k(\mathbf{y}_{t_m}, t_m, \mathbf{h}_i), \\ v_{t_m} &= u_{t_m}^\theta(\mathbf{y}_{t_m}|\mathbf{h}_i) + d_m\,\Delta v_\phi(\mathbf{y}_{t_m}, t_m, \mathbf{h}_i), \end{aligned} \tag{21}$$

where $d_m = \mathbf{1}[a_m = \texttt{CORRECT}]$ is the correction indicator. Using explicit Euler integration, the state update is

$$\mathbf{y}_{t_{m+1}} = \mathbf{y}_{t_m} + \Delta t_m\, v_{t_m}. \tag{22}$$

When memory is empty or correction is disabled, $\Delta v_\phi(\mathbf{y}_{t_m}, t_m, \mathbf{h}_i) = 0$. If every action is `PASS`, the same initialization, solver, numerical precision, and sampling convention recover the frozen-backbone trajectory. A `PASS` action after earlier corrections only follows the backbone field from the already modified state; it does not undo earlier corrections.

B.3 COGNITIVE SENTINEL

The Sentinel is trained via a two-stage recipe: behavior-cloning warm-start followed by Group Relative Policy Optimization (GRPO). This is motivated by the need to balance two competing desiderata: ensuring that critical memory invocations improve forecasting, while avoiding unnecessary or spurious invocations that could disrupt the ODE trajectory or incur computational overhead.

Stage 1: Behavior Cloning. During training, the ground-truth velocity along the conditional optimal-transport path is available as

$$u_t^{\text{true}} = \mathbf{y}_i - \mathbf{y}_i^{(0)}. \tag{23}$$

The velocity discrepancy is

$$e_t = u_t^{\text{true}} - u_t^{\theta}. \tag{24}$$

We also compute the numerical error E_{num} by comparing a fine-step and a coarse-step integration of the same velocity field. A heuristic label is generated as

$$a_m^* = \begin{cases} \texttt{CORRECT}, & \text{if } \|e_t\| > \tau_{\text{model}} \text{ or } E_{\text{num}} > \tau_{\text{num}}, \\ \texttt{PASS}, & \text{otherwise.} \end{cases} \tag{25}$$

The thresholds τ_{num} and τ_{model} are set as quantiles of the validation error distribution, ensuring balanced label proportions across the two actions. The Sentinel is then warm-started by minimizing the cross-entropy loss:

$$\mathcal{L}_{\text{BC}}(\psi) = -\mathbb{E}_{s_m}\left[\log \pi_\psi(a_m^*|s_m)\right]. \tag{26}$$

Stage 2: GRPO Fine-Tuning. After warm-start, the Sentinel is fine-tuned with GRPO while the backbone and the Retracer remain frozen. For each task, we sample K trajectories with different Sentinel action sequences and compute the group-relative advantage:

$$A(\tau_k) = R(\tau_k) - \frac{1}{K}\sum_{j=1}^{K} R(\tau_j). \tag{27}$$

To encourage sparse yet strategically critical memory invocation, we introduce a data-adaptive sparsity penalty that discourages unnecessary activations while preserving essential ones. The reward combines a terminal forecasting metric with this penalty:

$$R = -\|\mathbf{y}_i - \hat{\mathbf{y}}_i^{(K)}\|^2 - \lambda \sum_m \max(0, d_m - \bar{d}) - \beta \cdot \text{NFE}, \tag{28}$$

where $d_m = \mathbf{1}[a_m = \texttt{CORRECT}]$ is the correction indicator of the current trajectory, $\bar{d}$ is the average correction rate of high-reward trajectories in the batch, and NFE is the number of function evaluations. The policy is updated by

$$\nabla_\psi \mathcal{J} = \mathbb{E}\left[\sum_k A(\tau_k)\nabla_\psi \log \pi_\psi(\tau_k)\right] - \gamma \nabla_\psi \text{KL}(\pi_\psi \| \pi_{\text{ref}}). \tag{29}$$

B.4 MENTAL RETRACER: GENERATIVE MEMORY POOL

The Retracer is trained via a three-stage recipe: generator pre-training, memory pool initialization, and online consolidation. This is motivated by the need to balance generalization to unseen drift patterns with specialization to recurring ones, while avoiding catastrophic forgetting.

Stage 1: Generator Pre-Training. We first train a dedicated generator LoRA that produces a residual velocity field from the current flow state:

$$\Delta v_\phi^{\text{gen}}(\mathbf{y}_t, t, \mathbf{h}_i) = \text{LoRA}_{\text{gen}}(\mathbf{y}_t, t, \mathbf{h}_i). \tag{30}$$

During training, the true velocity along the conditional optimal-transport path is available as $u_t^{\text{true}} = \mathbf{y}_i - \mathbf{y}_i^{(0)}$. We sample points on the reference path

$$\mathbf{y}_t^* = (1-t)\mathbf{y}_i^{(0)} + t\mathbf{y}_i, \tag{31}$$

and train the generator to match the residual between the true velocity and the frozen generator's prediction:

$$\mathcal{L}_{\text{gen}}(\phi) = \mathbb{E}_{t,\mathbf{y}_i^{(0)},\mathbf{y}_i}\left[\left\|\text{LoRA}_{\text{gen}}(\mathbf{y}_t^*, t, \mathbf{h}_i) - \left(u_t^{\text{true}} - u_t^\theta(\mathbf{y}_t^*|\mathbf{h}_i)\right)\right\|_2^2\right]. \tag{32}$$

Training on the reference path avoids distribution shift and provides dense, clean supervision.

Stage 2: Memory Pool Initialization. We cluster the residual patterns on a validation set and initialize one memory trace per cluster. The prototype is set to the cluster centroid, and the residual expert is initialized from the generator and fine-tuned on the cluster:

$$\mathcal{L}_{\text{trace}}(\phi_k) = \mathbb{E}_{(\mathbf{y}_t,t,\mathbf{h}_i)\in C_k}\left[\left\|\text{LoRA}_k(\mathbf{y}_t, t, \mathbf{h}_i) - \left(u_t^{\text{true}} - u_t^\theta(\mathbf{y}_t|\mathbf{h}_i)\right)\right\|_2^2\right]. \tag{33}$$

The query network MLP_q is jointly trained with an InfoNCE objective that aligns queries with their cluster prototypes:

$$\mathcal{L}_{\text{query}} = -\sum_k \log \frac{\exp(\text{sim}(q_t, \mu_k)/\tau)}{\sum_j \exp(\text{sim}(q_t, \mu_j)/\tau)}. \tag{34}$$

C Algorithm

We summarize MemTime in two algorithms. Algorithm 1 is the online inference path: given a segment, it runs the ODE rollout, queries the Sentinel at a sparse schedule, and applies memory corrections only when CORRECT is selected. Algorithm 2 is the offline update path: after the ground-truth target becomes available, it computes the trajectory-level improvement, updates trace utilities, and either creates a new trace or applies lifecycle transitions. The two paths share the memory pool $\mathcal{M}$ and the global step counter g_t, and the memory snapshot used for inference is always frozen at the forecast origin.

Algorithm 1 MemTime: Online Streaming Inference

Require: frozen FM-Net u_θ; memory pool $\mathcal{M} = \{(\mu_k, \text{LoRA}_k, U_k, \tau_k, s_k, c_k, a_k)\}$; Sentinel policy π_ψ; query network MLP_q; thresholds τ_U, τ_E, τ_J; Sentinel interval M; capacity $K_{\max}$; budget B

Ensure: forecast $\hat{\mathbf{y}}_i$ for each stream segment

for each incoming stream segment $\mathbf{x}_i$ **do**
  $\mathbf{h}_i \leftarrow \text{Encoder}(\mathbf{x}_i)$; $\mathbf{y}_0 \leftarrow \text{InitState}(\mathbf{x}_i)$
  $B_t \leftarrow B$; $\mathcal{A} \leftarrow \emptyset$; $w_k \leftarrow 0\ \forall k$
  for $t = 0$ to $K-1$ **do**
    $g_t \leftarrow iK + t$
    $u_t^\theta \leftarrow u_\theta(\mathbf{y}_t, t, \mathbf{h}_i)$; $B_t \leftarrow B_t - \text{cost}(u_\theta)$
    $a_t \leftarrow \textsc{SentinelStep}(\mathbf{y}_t, \mathbf{h}_i, B_t, t)$
    if $a_t = \texttt{CORRECT}$ **and** $B_t > 0$ **then**
      $(v_t, \mathcal{A}, w_k) \leftarrow \textsc{QueryMemory}(\mathbf{y}_t, \mathbf{h}_i, u_t^\theta, \mathcal{M}, \mathcal{A}, w_k)$
    else
      $v_t \leftarrow u_t^\theta$
    end if
    $\mathbf{y}_{t+1} \leftarrow \text{ODEStep}(\mathbf{y}_t, v_t, \Delta t)$
  end for
  $\hat{\mathbf{y}}_i \leftarrow \mathbf{y}_K$
  if ground-truth $\mathbf{y}_i$ becomes available **then**
    $\textsc{FeedbackUpdate}(\mathcal{M}, \mathcal{A}, w_k, \mathbf{y}_i, \hat{\mathbf{y}}_i, g_t, K_{\max})$
  end if
end for

Algorithm 2 Memory Metabolism: Offline Update and Maintenance

procedure FEEDBACKUPDATE($\mathcal{M}, \mathcal{A}, w_k, \mathbf{y}_i, \hat{\mathbf{y}}_i, g_t, K_{\max}$)
  $\Delta L \leftarrow$ UNCORRECTEDLOSS($\mathbf{x}_i$) $- \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2$
  for $k \in \mathcal{A}$ **do** ▷ utility update
    $U_k \leftarrow (1-\eta)U_k + \eta\, w_k \Delta L$
    $U_k \leftarrow U_k \exp(-\gamma(g_t - \tau_k))$; $\tau_k \leftarrow g_t$
  end for
  if $\Delta L > \tau_{\text{create}}$ **and** no matching trace **then**
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{$NEWTRACE($\mathbf{y}_i, \hat{\mathbf{y}}_i, \Delta L, g_t$)$\}$
  end if
  UPDATELIFECYCLE($\mathcal{M}, K_{\max}, g_t$)
end procedure

procedure UPDATELIFECYCLE($\mathcal{M}, K_{\max}, g_t$)
  for all $k \in \mathcal{M}$ **do** ▷ state transitions
    $s_k \leftarrow$ NEXTSTATE(s_k, U_k, τ_k, g_t)
  end for
  while $|\mathcal{M}| > K_{\max}$ **do** ▷ capacity control
    Evict by priority: `PRUNED` $\rightarrow$ low-U `DORMANT` $\rightarrow$ merge similar `ACTIVE` $\rightarrow$ least useful
  end while
end procedure

function NEWTRACE($\mathbf{y}_i, \hat{\mathbf{y}}_i, \Delta L, g_t$)
  $\mu \leftarrow \text{Desc}(\mathbf{y}_i - \hat{\mathbf{y}}_i)$; LoRA $\leftarrow \text{Fit}(\mathbf{y}_i - \hat{\mathbf{y}}_i)$
  return (μ, LoRA, $\Delta L, g_t$, `NEW`)
end function

D MORE EXPERIMENTAL RESULTS

D.1 ABLATION STUDY ON LONG-TERM FORECASTING

Table 3: Long-horizon streaming stability on Exchange ($T = 5000$). Sliding-window MSE is reported.

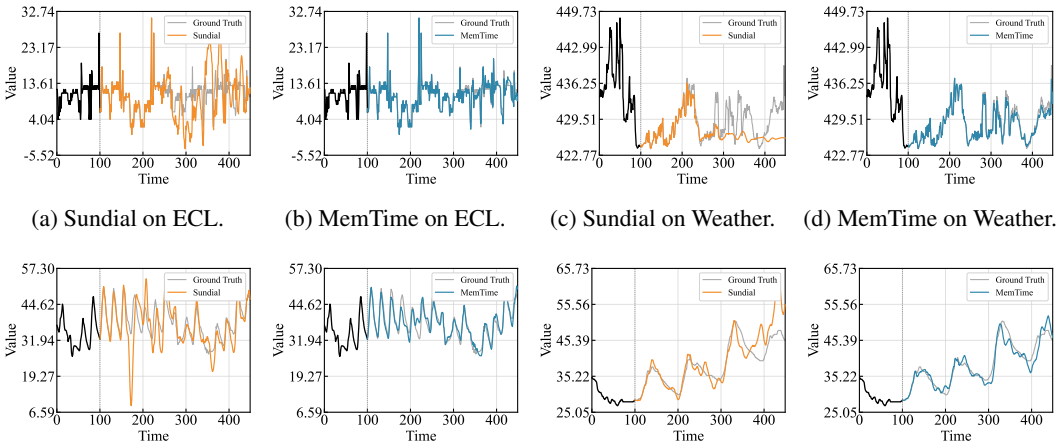
Streaming Steps	Sundial	w/o Sentinel	w/o h_local	w/o Memory Metabolism	w/o GRPO	MemTime
1000	0.463	0.407	0.442	0.356	0.371	**0.343**
3000	0.514	0.453	0.517	0.382	0.408	**0.354**
5000	0.608	0.546	0.623	0.457	0.472	**0.362**

Table 3 should not be read merely as a ranking of variants. More revealing is the behavior of *w/o Memory Metabolism*: it is competitive at step 1,000 but degrades sharply by step 5,000. This growing gap suggests that memory metabolism is not a short-term accuracy booster; its role is to preserve the relevance and plasticity of the memory pool as the environment drifts. Without consolidation and pruning, useful traces are gradually diluted by stale ones, and the correction signal becomes misaligned with the current regime. Similarly, *w/o GRPO* remains better than *w/o Sentinel* but worse than MemTime, indicating that the learned invocation policy matters beyond a fixed heuristic: the Sentinel must decide not only whether to retrieve, but when retrieval is likely to help. Overall, the table supports a view of long-term adaptation as a coordination problem among invocation, correction, and consolidation, rather than a purely representational one.

D.2 CASE STUDY

Figure 6 compares Sundial and MemTime on ECL, Weather, ETTh2, and ETTm2, where the historical segment and ground truth are identical across the two methods, so the forecast difference stems from the model itself. On ECL, Sundial drifts upward and remains above the ground truth

for most of the horizon, while MemTime stays close with only small local fluctuations; on Weather, Sundial stays below the ground truth and fails to follow its upward trend, whereas MemTime tracks the rising trend accurately. The same pattern appears on ETTh2 and ETTm2: Sundial exhibits systematic bias and suppresses local fluctuations or trend changes, while MemTime almost overlaps the ground truth. Across all cases, Sundial produces a smooth but rigid forecast whose initial deviation accumulates over the horizon, whereas MemTime is locally adaptive and preserves high-frequency and nonlinear variations. Overall, the case study suggests that static pre-trained knowledge alone is insufficient for long-term forecasting, and dynamic inference-time error correction is essential for robust performance on non-stationary time series.

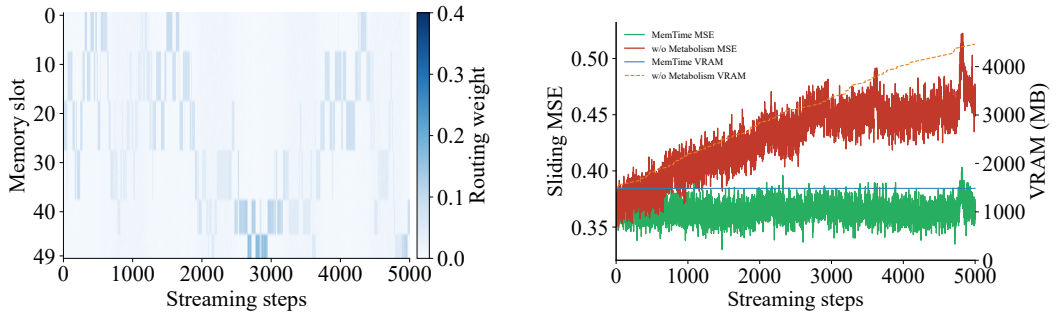


(a) Sundial on ECL. (b) MemTime on ECL. (c) Sundial on Weather. (d) MemTime on Weather.

(e) Sundial on ETTh2. (f) MemTime on ETTh2. (g) Sundial on ETTm2. (h) MemTime on ETTm2.

Figure 6: Case Study.

D.3 More intelligent emergence tests



(a) Memory rounting matrix $\alpha_{t,k}$ ($K = 50$). (b) Long-horizon stability and memory cost.

Figure 7: Memory Routing Dynamics and Long-Horizon Stability. (a) The memory routing matrix $\alpha_{t,k}$ ($K = 50$) over streaming steps. (b) Sliding MSE and VRAM consumption of MemTime and its variants without the memory mechanism.

Figure 7 shows MemTime's advantages in adaptive memory routing and stable long-horizon forecasting. As shown in Figure 7a, the routing weights $\alpha_{t,k}$ over $K = 50$ slots are sparse and dynamic: at each step, MemTime attends to only a small subset of slots, and the activated slots shift over time. Thus, the memory pool acts as an adaptive retrieval module rather than a static cache, activating relevant historical residuals only when needed. This avoids redundant memory access and enables fine-grained responses to non-stationary shifts. Figure 7b confirms the benefit: without MemTime, sliding MSE rises steadily as errors accumulate, whereas MemTime remains low and stable. This stability comes at a modest, bounded memory cost: MemTime uses more VRAM initially but stays nearly flat, while the baseline grows continuously and eventually exceeds it.