ADAPTATION AS MEMORY: LEARNING TO CORRECT TIME SERIES FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Time series foundation models (TSFMs) generalize across domains, but adapting them to target-domain dynamics remains challenging. Existing adaptation methods use diverse mechanisms, yet lack a unified framework for understanding how target-domain knowledge is retained, accessed, updated, and applied. We address this gap by viewing TSFM adaptation as a memory process of construction, writing, reading, and execution. We instantiate this view with **TS-CorM**, a structured fast–slow corrective memory for frozen TSFMs. TS-CorM represents forecast corrections with compact coefficients, addresses them using joint history–forecast states, and retains their associations in explicit memory matrices. Slow Memory consolidates offline knowledge, while Fast Memory incorporates delayed feedback through recursive online updates without backpropagation. Across eight datasets and five TSFM backbones, TS-CorM improves forecasting in all 40 backbone–dataset pairs, with MSE reductions of up to 22.28%; compared with the best adaptation baseline, it reduces average MSE by 3.78% with 98.32% fewer parameters and 10.23% lower inference latency. Anonymous code: https://anonymous.4open.science/r/TS-CorM-23CC/.

1 INTRODUCTION

Time series foundation models (TSFMs) learn transferable temporal patterns through large-scale pretraining and achieve strong zero-shot forecasting performance across domains (Ansari et al., 2024; Das et al., 2024; Woo et al., 2024). Yet target-specific scales and evolving dynamics can leave a gap between pretrained forecasting capabilities and downstream requirements (Zhao et al., 2025; Ning et al., 2025). TSFM adaptation seeks to close this gap by using target-domain knowledge to refine a pretrained forecaster.

Existing TSFM adaptation methods fall into three categories. (1) Parametric adaptation updates backbone weights or learns auxiliary adapters (Qiao et al., 2025; Benechehab et al., 2025). (2) Retrieval augmentation accesses historical context–future pairs to inform predictions (Han et al., 2025; Ning et al., 2025). (3) Parameterized memory retains domain knowledge in learned modules: TS-Memory distills retrieval-derived supervision into a learned adapter (Lyu et al., 2026), while MEMTS encodes domain-specific patterns in learnable latent prototypes (Yu et al., 2026). Viewed together, these approaches share a common function: retaining knowledge from the target domain and reusing it for subsequent forecasts. This observation motivates a unified view of TSFM adaptation as a memory process: **Adaptation as Memory**.

We formalize TSFM adaptation through four memory operations: construct, write, read, and execute. Construct initializes memory from offline domain data, write updates it with newly available information, read accesses relevant knowledge for the current forecast, and execute determines how the readout affects the final prediction. This decomposition separates how domain knowledge is retained, updated, accessed, and applied, allowing different adaptation paradigms to be compared under a common interface. Figure 1 shows how existing adaptation paradigms realize these operations with different memory representations and mechanisms.

This perspective also exposes several design gaps. **What should memory retain?** Parametric adaptation retains domain knowledge in model parameters, which may weaken pretrained capabilities. Retrieval augmentation stores historical knowledge in an external database, introducing growing storage and search costs. Parameterized memory distills retrieval process into learned modules, but this still leaves the retained knowledge tied to the teacher retrieval rather than explicitly organized around forecast correction. This brings a question: what knowledge should memory retain for fore-

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.

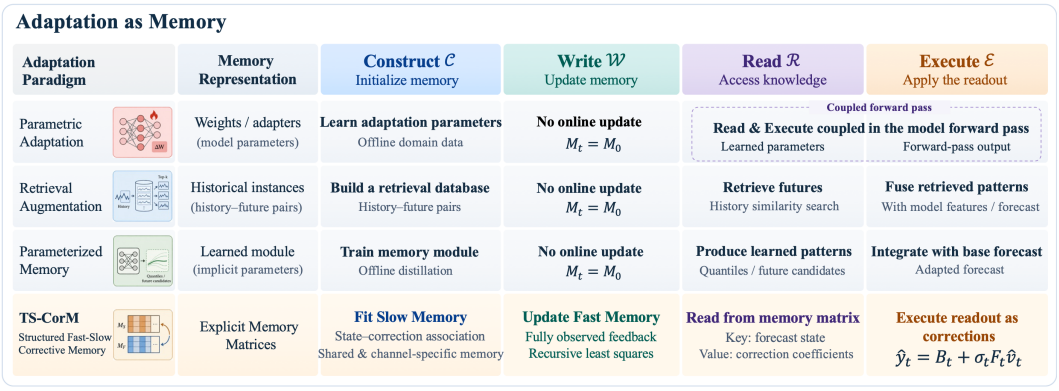


Figure 1: Adaptation paradigms under the **Adaptation as Memory** framework. We characterize TSFM adaptation through four memory operations: construct, write, read, and execute. We further show how parametric adaptation, retrieval augmentation, parameterized memory, and our TS-CorM realize these operations with different memory representations and mechanisms.

cast correction? **How should memory be read?** Historical similarity alone may not identify the adjustment required by the current forecast, since correction needs depend on both the observed history and the model's own prediction. **How should memory be updated?** In online forecasting, temporal dynamics evolve over time, yet existing offline memories provide limited ways to adapt to these changes without retraining or expanding external storage.

Motivated by these gaps, we propose TS-CorM, a structured fast–slow corrective memory for frozen TSFMs. TS-CorM retains compact forecast-relative corrections in Slow Memory, addresses them with forecast states that combine history and model predictions, and updates its associations online through Fast Memory while preserving offline Slow Memory. Specifically, we represent forecast corrections as compact coefficients over a structured **correction dictionary** and use **forecast states** for memory addressing. The dictionary combines fixed smooth directions with history-derived correction directions. We encode observed training residuals as oracle correction coefficients over this dictionary and use them to supervise memory learning. A frozen encoder separately embeds the historical context and the model's forecast; their concatenation forms a forecast state that serves directly as the memory key. Slow Memory learns state–coefficient associations through shared and channel-specific ridge regression. Once the corresponding target horizons are fully observed, Fast Memory encodes the remaining errors of Slow Memory forecasts as incremental coefficients in the same coefficient space and updates its associations through recursive least squares. Both memories store associations in explicit matrices. Their linear readouts are combined and executed through the current dictionary, enabling forecast correction without external retrieval and online updates without backpropagation.

Our contributions are threefold:

- **Adaptation as Memory.** We introduce a unified framework that views TSFM adaptation through memory construction, writing, reading, and execution, providing a common language for how domain knowledge is retained, accessed, and applied.
- **TS-CorM.** We propose a structured fast–slow corrective memory that associates forecast states with compact corrective coefficients. Slow Memory captures offline domain knowledge, while Fast Memory incorporates delayed feedback through recursive updates, enabling retrieval-free and backpropagation-free online adaptation.
- **Comprehensive Evaluation.** Extensive experiments show that TS-CorM improves forecasting performance across diverse TSFMs, with MSE reductions of up to 22.28%. Compared with the best adaptation baseline, TS-CorM reduces average MSE by 3.78% with with 98.32% fewer parameters and 10.23% lower inference latency.

2 Related Work

2.1 Time Series Foundation Models

TSFMs learn transferable temporal patterns through pretraining on heterogeneous time series. Early efforts, including TimeGPT and Lag-Llama, demonstrated the potential of cross-domain zero-shot forecasting (Garza et al., 2023; Rasul et al., 2023), followed by TimesFM, Moirai, and Chronos

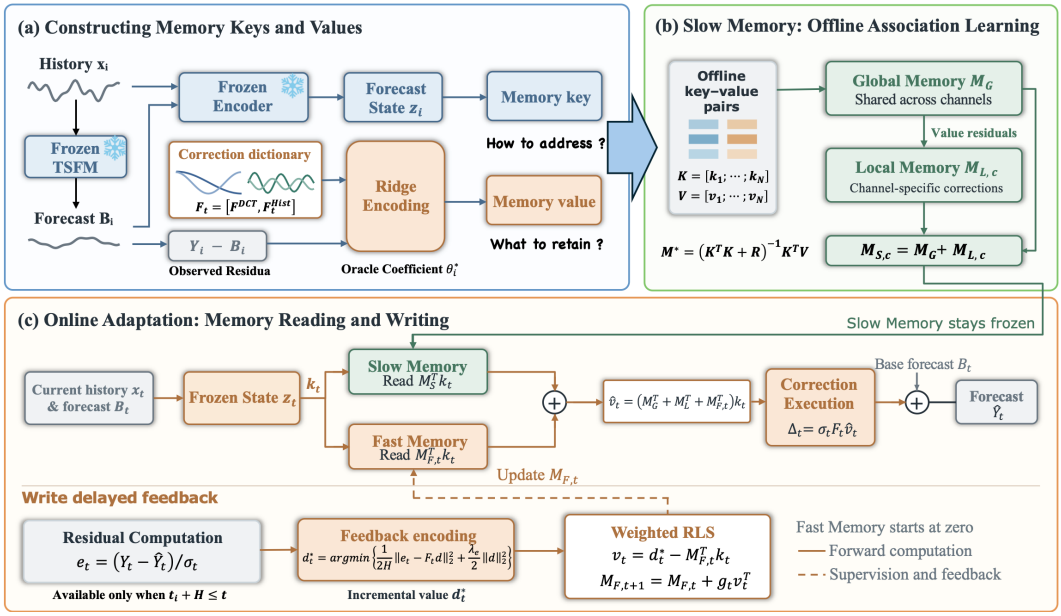


Figure 2: Overview of TS-CorM. (a) Forecast states and oracle correction coefficients form memory keys and values. (b) Shared and channel-specific Slow Memory learn offline associations. (c) Fast Memory incorporates delayed feedback, and both memories provide corrections for the current forecast.

(Das et al., 2024; Woo et al., 2024; Ansari et al., 2024). In parallel, MOMENT extended pretraining to broader time series analysis tasks (Goswami et al., 2024). Subsequent work explored compact architectures with Tiny Time Mixers (Ekambaram et al., 2024), capacity scaling through sparse experts with Time-MoE and Moirai-MoE (Shi et al., 2025; Liu et al., 2025b), and recurrent in-context forecasting with TiRex (Auer et al., 2025). Sundial and Moirai 2.0 further advanced probabilistic forecasting through flow-based and quantile-based modeling, respectively (Liu et al., 2025c;a). More recently, Chronos-2, TiRex-2, and TimesFM-3 have strengthened multivariate and covariate-informed zero-shot forecasting (Ansari et al., 2025; Podest et al., 2026; Jain & Sen, 2026), with TiRex-2 additionally supporting streaming inference through recurrent state updates.

2.2 Adaptation of Time Series Foundation Models

Adapting TSFMs to target-domain distributions and temporal dynamics has motivated three main approaches. **Fine-tuning and adapters** encode domain knowledge through backbone updates or task-specific transformations, including feature-space adapters for probabilistic multivariate forecasting with univariate models (Benechehab et al., 2025), multiscale fine-tuning (Qiao et al., 2025), and structured pruning followed by fine-tuning (Zhao et al., 2025). **Retrieval-augmented methods** select historical examples from external knowledge bases and incorporate them into current forecasts (Zhang et al., 2025; Ning et al., 2025; Tire et al., 2026). **Parameterized memory methods** internalize forecasting knowledge: TS-Memory distills retrieval-induced distributional corrections into a lightweight adapter (Lyu et al., 2026), while MEMTS encodes domain-specific future patterns in parameterized memory (Yu et al., 2026). Both freeze the backbone and eliminate inference-time retrieval.

These approaches differ in how they retain, access, and update target-domain knowledge. In contrast, our framework explicitly separates these functions and motivates a corrective memory that stores forecast-relative knowledge, accesses it using joint history–forecast states, and supports online updates without retraining or inference-time retrieval.

3 Methodology

Problem setup. For a multivariate time series with C channels, let $\mathbf{X}_t = \mathbf{x}_{t-L:t} \in \mathbb{R}^{L\times C}$ denote the look-back window and $\mathbf{Y}_t = \mathbf{x}_{t:t+H} \in \mathbb{R}^{H\times C}$ the future target. A frozen pretrained TSFM f_θ produces a baseline forecast $B_{t,c} \in \mathbb{R}^H$. Given offline target-domain data $\mathcal{D}_{\text{off}}$, we learn a Slow Memory matrix $M_{S,c}$ for forecast correction. During online forecasting, a zero-initialized Fast

Memory matrix $M_{F,t,c}$ learns incremental corrections from fully observed past targets satisfying $t_i + H \leq t$. The two memories then refine $B_{t,c}$ into the adapted forecast $\widehat{Y}_{t,c} \in \mathbb{R}^H$.

Overview. As illustrated in Figure 2, we propose TS-CorM, a structured fast–slow corrective memory with three components. (a) *Constructing memory keys and values:* a frozen encoder embeds the history and TSFM forecast to form a forecast state, which serves as the key. Using a correction dictionary, we encode the observed residual into oracle correction coefficients, which serve as the value. (b) *Slow Memory construction:* offline ridge regression learns a shared state-to-coefficient mapping followed by channel-specific residual adjustments. (c) *Online adaptation:* during online forecasting, Slow Memory remains frozen. Fast Memory learns incremental correction coefficients via weighted recursive least squares when past target horizons become fully observed. This enables online adaptation to evolving temporal dynamics.

3.1 Memory values as structured correction coefficients

Structured Correction Expansion. Direct residual prediction requires an H-dimensional output whose size grows with the horizon. Residual trajectories may also contain complex temporal structure and instance-specific fluctuations. We therefore seek a compact structured representation that captures reusable correction patterns while reducing output dimensionality. Drawing on the idea of basis expansion (Oreshkin et al., 2020), we represent each residual correction as a combination of a small set of structured patterns and learn only their expansion coefficients. At origin t, these patterns form a correction dictionary $F_t \in \mathbb{R}^{H\times d}$, with $d \ll H$. Low-dimensional coefficients $\theta_t \in \mathbb{R}^d$ yield

$$\Delta_t = \sigma_t F_t \theta_t, \qquad \widehat{Y}_t = B_t + \Delta_t, \tag{1}$$

where B_t is the TSFM forecast and σ_t is a history-derived scale.

Complementary correction patterns. Forecast errors exhibit heterogeneous structures, including smoothly varying deviations from the baseline forecast and recurring patterns tied to recent history. We therefore combine discrete cosine transform (DCT) directions with history-dependent directions: $F_t = [F^{\mathrm{DCT}}, F_t^{\mathrm{hist}}] \in \mathbb{R}^{H\times d}$. The DCT component uses a constant term and low-frequency cosine functions to capture broad, smooth deviations from the baseline forecast, while history-dependent directions model more flexible recurring structures.

For historical directions, we identify dominant periods p_j using the fast Fourier transform (FFT) and average recent phase-aligned historical trajectories to obtain a reference $\bar{x}_{t,j,m}$. Its deviation from the current baseline defines a normalized historical correction direction:

$$u_{t,j,m} = \frac{\bar{x}_{t,j,m} - B_t}{\sigma_t}, \qquad F^{\mathrm{hist}}_{t,:,(j,m)} = \frac{u_{t,j,m}}{\max(\mathrm{RMS}(u_{t,j,m}), \epsilon_F)}. \tag{2}$$

Normalization controls the scale of dictionary columns. The construction rule is shared across channels, while the resulting historical directions remain instance-specific through their dependence on the current history and forecast.

Oracle coefficients for memory supervision. Given the correction dictionary, we encode each observed training residual into oracle correction coefficients. For each labeled instance i, let $r_i = (Y_i - B_i)/\sigma_i$ denote the scale-normalized residual. The oracle coefficients are obtained by ridge regression:

$$\theta_i^\star = \arg\min_\theta \left\{ \frac{1}{2H}\|F_i\theta - r_i\|_2^2 + \frac{\lambda_o}{2}\|\theta\|_2^2 \right\}. \tag{3}$$

The regularization parameter $\lambda_o > 0$ stabilizes estimation for correlated directions. The problem admits the closed-form solution $\theta_i^\star = (F_i^\top F_i/H + \lambda_o I)^{-1} F_i^\top r_i/H$. These coefficients provide supervision for memory learning. "Oracle" denotes access to observed training targets when constructing supervision; at inference, coefficients must be predicted using only information available at the forecast origin.

3.2 Memory Addressing with Forecast States

Forecast Context for Memory Addressing. Correction coefficients specify what to remember, while memory addressing determines how retained correction associations are accessed for a given forecast. Similar histories can require different corrections when the model produces different forecasts. Because corrections are defined relative to the model's forecast, memory addresses should characterize both the observed dynamics and the model's response. We therefore encode the historical context and forecast with the same frozen encoder and concatenate their embeddings to form a

forecast state. The resulting state serves directly as the memory key, allowing memory to associate the observed context and the model's forecast with relevant corrective coefficients.

Constructing the forecast state. We construct the forecast state from two embeddings produced by the same frozen encoder. Let $\mathcal{E}_{\mathrm{REG}}(\cdot)$ denote the final REG-token representation produced by the frozen encoder. We encode the historical context $x_{t,c}^{\mathrm{ctx}}$ and the baseline forecast $B_{t,c}$ separately:

$$h_{t,c}^{\mathrm{ctx}} = \mathcal{E}_{\mathrm{REG}}(x_{t,c}^{\mathrm{ctx}}), \qquad h_{t,c}^{\mathrm{pred}} = \mathcal{E}_{\mathrm{REG}}(B_{t,c}), \qquad z_{t,c} = [h_{t,c}^{\mathrm{ctx}}; h_{t,c}^{\mathrm{pred}}] \in \mathbb{R}^p. \tag{4}$$

The history embedding describes the observed context, while the forecast embedding describes the trajectory anticipated by the TSFM. Appendix A.1 specifies this configuration.

3.3 Slow Memory: Offline Construction

We construct Slow Memory from offline forecast states as keys and oracle correction coefficients as values:

$$k_i = z_i \in \mathbb{R}^p, \qquad v_i^\star = \theta_i^\star \in \mathbb{R}^d, \tag{5}$$

A memory matrix $M \in \mathbb{R}^{p\times d}$ maps each key to predicted correction coefficients, $\widehat{\theta}_t = \widehat{v}_t = M^\top k_t$. Following the associative regression view (Wang et al., 2025), we fit this mapping by ridge regression. Let $K \in \mathbb{R}^{N\times p}$ and $\mathcal{V} \in \mathbb{R}^{N\times d}$ stack the keys and oracle coefficients as rows. For a positive definite regularization matrix $R \in \mathbb{R}^{p\times p}$,

$$M^\star = \arg\min_M \left\{ \frac{1}{2}\|\mathcal{V} - KM\|_F^2 + \frac{1}{2}\operatorname{tr}(M^\top R M) \right\} \quad = (K^\top K + R)^{-1} K^\top \mathcal{V}. \tag{6}$$

This regression consolidates offline state–coefficient associations into an explicit memory matrix, while regularization stabilizes estimation when keys are correlated. At inference, $M^\top k_t$ retrieves correction coefficients for execution in the current correction dictionary, yielding the correction $\sigma_t F_t M^\top k_t$.

Channels may share correction patterns while also requiring channel-specific adjustments. We therefore define the Slow Memory for channel c as

$$M_{S,c} = M_G + M_{L,c}, \qquad \widehat{\theta}_{t,c}^{\mathrm{slow}} = \widehat{v}_{t,c}^{\mathrm{slow}} = M_{S,c}^\top k_{t,c}, \tag{7}$$

where M_G is shared across channels and $M_{L,c}$ captures channel-specific residual structure. We first fit M_G jointly on offline state–coefficient pairs from all channels using Eq. (6). With M_G fixed, each $M_{L,c}$ is fitted to residual coefficient targets $E_c = \mathcal{V}_c - K_c M_G$ with stronger regularization. This yields a shared state-to-correction mapping with channel-specific residual adjustments. Sharing is restricted to channels within the same dataset and forecast horizon.

3.4 Fast Memory: Online Writing

As forecasting conditions evolve, Fast Memory writes delayed feedback into the same key–coefficient space while keeping the backbone, offline representations, and Slow Memory frozen. Updates use only fully observed target horizons. For a fixed channel c, we omit channel subscripts and write $M_S = M_{S,c}$, $M_{F,t} = M_{F,t,c}$.

Encoding incremental corrections. For episode i, the Slow Memory forecast at issuance is $S_i = B_i + \sigma_i F_i M_S^\top k_i$. Once Y_i is fully observed, we encode its normalized residual $e_i = (Y_i - S_i)/\sigma_i$ in the original dictionary F_i:

$$d_i^\star = \arg\min_{d\in\mathbb{R}^d} \left\{ \frac{1}{2H}\|e_i - F_i d\|_2^2 + \frac{\lambda_e}{2}\|d\|_2^2 \right\}. \tag{8}$$

The resulting $d_i^\star$ is an incremental correction in the same coefficient space as the Slow Memory readout. It corrects the residual of S_i directly, without subtracting the Slow Memory readout.

Causal associative writing. Before forecasting at t, we process unwritten episodes with $t_i + H \le t$ chronologically, using their recorded keys, dictionaries, scales, and Slow Memory forecasts. Weighted recursive least squares (RLS) maintains $M_{F,t} \in \mathbb{R}^{p\times d}$:

$$M_{F,t} = \arg\min_M \left\{ \frac{1}{2}\sum_{i\in\mathcal{I}_t} w_i \left\| d_i^\star - M^\top k_i \right\|_2^2 + \frac{\lambda_F}{2}\|M\|_F^2 \right\}, \tag{9}$$

where $\lambda_F > 0$ and $\mathcal{I}_t$ contains the written episodes. Each weight $w_i \ge 0$ is computed once from $e_i - F_i(M_F^-)^\top k_i$, normalized by a fixed channel-specific scale, to downweight unusually large errors.

Table 1: Forecasting results comparing different adaptation baselines on ChronosBolt (lower is better). Forecast horizons are $H \in \{96, 192, 336, 720\}$, with a lookback window of 512 time steps. **Red**: the best. Blue: the second best. TS-CorM (offline) achieves the best performance in nearly all evaluated settings.

Dataset	Horizon	TS-CorM (offline)		TSFM		RAFT		TS-RAG		LoRA		TS-Memory	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.359**	**0.372**	0.385	0.380	0.378	0.380	0.378	0.379	0.376	0.378	0.374	0.377
	192	**0.405**	**0.403**	0.442	0.412	0.431	0.411	0.432	0.411	0.425	0.410	0.410	0.407
	336	0.440	**0.422**	0.481	0.433	0.465	0.430	0.470	0.434	0.455	0.429	**0.439**	0.425
	720	**0.445**	**0.442**	0.484	0.449	0.473	0.450	0.489	0.462	0.457	0.447	0.462	0.445
ETTh2	96	**0.279**	**0.323**	0.290	0.325	0.286	0.325	0.283	0.326	0.284	0.324	0.282	0.324
	192	**0.335**	**0.363**	0.360	0.370	0.352	0.366	0.351	0.370	0.352	0.369	0.351	0.366
	336	**0.369**	**0.393**	0.403	0.403	0.390	0.397	0.393	0.402	0.390	0.397	0.388	0.396
	720	**0.381**	**0.411**	0.414	0.421	0.402	0.423	0.418	0.443	0.401	0.419	0.396	0.416
ETTm1	96	**0.293**	**0.321**	0.332	0.333	0.313	0.327	0.324	0.334	0.307	0.328	0.303	0.325
	192	**0.336**	**0.349**	0.387	0.366	0.369	0.360	0.376	0.365	0.362	0.359	0.359	0.358
	336	**0.385**	**0.376**	0.436	0.393	0.415	0.388	0.421	0.391	0.411	0.386	0.403	0.383
	720	**0.450**	**0.413**	0.528	0.439	0.493	0.432	0.503	0.436	0.487	0.431	0.459	0.418
ETTm2	96	**0.166**	**0.239**	0.178	0.245	0.174	0.243	0.179	0.249	0.170	0.242	0.167	0.240
	192	**0.229**	**0.284**	0.251	0.294	0.243	0.290	0.244	0.294	0.238	0.290	0.232	0.288
	336	**0.283**	**0.321**	0.317	0.336	0.306	0.332	0.304	0.334	0.301	0.330	0.292	0.328
	720	**0.362**	**0.374**	0.416	0.395	0.401	0.394	0.405	0.396	0.401	0.392	0.378	0.386
Exchange	96	0.090	**0.209**	0.099	0.219	0.096	0.216	0.093	0.213	0.092	0.211	**0.088**	0.209
	192	**0.174**	**0.297**	0.194	0.314	0.188	0.311	0.190	0.312	0.188	0.310	0.189	0.310
	336	**0.307**	**0.401**	0.342	0.424	0.334	0.418	0.336	0.418	0.335	0.418	0.335	0.419
	720	**0.821**	**0.680**	0.849	0.692	0.857	0.689	0.848	0.689	0.850	0.689	0.846	0.687
Weather	96	**0.153**	**0.192**	0.178	0.210	0.167	0.202	0.176	0.215	0.159	0.200	0.157	0.198
	192	**0.195**	**0.233**	0.224	0.254	0.214	0.246	0.222	0.257	0.208	0.246	0.202	0.245
	336	**0.246**	**0.272**	0.284	0.295	0.276	0.287	0.284	0.298	0.267	0.287	0.252	0.283
	720	**0.316**	**0.321**	0.367	0.347	0.359	0.337	0.364	0.347	0.355	0.341	0.346	0.341
Electricity	96	**0.119**	**0.200**	0.124	0.211	0.123	0.210	0.122	0.210	0.121	0.211	0.121	0.208
	192	**0.138**	**0.225**	0.143	0.229	0.141	0.229	0.141	0.229	0.141	0.229	0.138	0.226
	336	**0.154**	**0.245**	0.161	0.248	0.159	0.248	0.159	0.248	0.159	0.248	0.159	0.246
	720	**0.201**	**0.284**	0.207	0.288	0.203	0.290	0.204	0.291	0.203	0.287	0.202	0.285
Traffic	96	**0.364**	**0.235**	0.387	0.241	0.384	0.240	0.383	0.240	0.379	0.239	0.376	0.237
	192	**0.386**	**0.246**	0.414	0.253	0.410	0.252	0.410	0.252	0.407	0.251	0.406	0.248
	336	**0.401**	**0.254**	0.436	0.263	0.429	0.262	0.429	0.262	0.427	0.260	0.427	0.259
	720	**0.453**	**0.278**	0.503	0.293	0.484	0.289	0.485	0.289	0.489	0.290	0.490	0.289
Average	–	**0.314**	**0.324**	0.344	0.337	0.335	0.333	0.338	0.337	0.331	0.333	0.326	0.330

Initialize $M_{F,0} = 0$ and $C_0 = \lambda_F^{-1} I_p$, where $C \in \mathbb{R}^{p \times p}$ maintains the inverse regularized key Gram matrix required for recursive ridge updates. For each matured pair $(k_i, d_i^\star)$, superscripts $-$ and $+$ denote states before and after writing:

$$\begin{aligned} q_i &= C^- k_i, & \nu_i &= d_i^\star - (M_F^-)^\top k_i, \\ M_F^+ &= M_F^- + \frac{w_i q_i \nu_i^\top}{1 + w_i k_i^\top q_i}, & C^+ &= C^- - \frac{w_i q_i q_i^\top}{1 + w_i k_i^\top q_i}. \end{aligned} \tag{10}$$

Here, ν_i is the pre-update coefficient residual of Fast Memory. Conditional on the realized weights, these rank-one updates exactly solve Equation (9) without backpropagation.

After processing available feedback, Slow and Fast Memory are jointly executed through the current dictionary:

$$\widehat{Y}_t = B_t + \sigma_t F_t \left(M_S + \eta_F M_{F,t}\right)^\top k_t. \tag{11}$$

The shrinkage factor $\eta_F \in [0, 1]$ controls the contribution of Fast Memory at execution. When no feedback has matured, $M_{F,t} = 0$, reducing Eq. (11) to the Slow Memory forecast.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate eight benchmark datasets covering electricity transformer temperature (ETTh1/2 and ETTm1/2), exchange rates (Exchange), meteorological measurements (Weather), electricity consumption (Electricity), and road occupancy (Traffic) (Liu et al., 2024). Dataset details and splits are provided in Appendix A.2.

TSFM Backbones & Adaptation Baselines. We evaluate five representative TSFM backbones: ChronosBolt-Base (Ansari et al., 2024), TimesFM 3.0 (Jain & Sen, 2026), Moirai 2.0 (Liu et al.,

2025a), Sundial (Liu et al., 2025c), and TiRex-2 (Podest et al., 2026). Adaptation baselines comprise low-rank fine-tuning with LoRA (Hu et al., 2022), retrieval augmentation with RAFT (Han et al., 2025) and TS-RAG (Ning et al., 2025), and retrieval-distilled TS-Memory (Lyu et al., 2026).

Implementation Details. We evaluate forecasting performance using MSE and MAE, with a lookback window of 512 and forecast horizons $H \in \{96, 192, 336, 720\}$. For baseline methods, we follow their original hyperparameter settings to ensure fair comparison. For all backbones, forecast states are constructed using the same frozen Chronos encoder. Fast Memory is initialized to zero and updates only after full target horizons are observed. All experiments are conducted on an NVIDIA A800 GPU with 80 GB. Further implementation details are provided in Appendix A.

4.2 Comparison with Adaptation Baselines

We compare TS-CorM (offline) with RAFT, TS-RAG, LoRA, and TS-Memory on ChronosBolt across eight datasets and four forecast horizons (Table 1). For a fair comparison, we disable Fast Memory and keep the adaptation parameters fixed throughout testing. TS-CorM achieves the lowest average MSE/MAE (0.314/0.324), ranking first on 30/32 MSE and all 32 MAE tasks. Compared with TS-Memory, the strongest baseline on average, it reduces average MSE and MAE by 3.78% and 1.84%, respectively. These gains are consistent across datasets and horizons: TS-CorM performs best on every dataset when averaged over horizons and at every horizon when averaged across datasets. Its average MSE reduction over TS-Memory rises from 2.41% at $H = 96$ to 4.22% at $H = 720$, highlighting its effectiveness for longer-horizon forecasting. These results demonstrate effective downstream adaptation through offline corrective memory without backbone updates, inference-time retrieval, or test-time feedback.

4.3 Performance Across Backbones

Table 2 evaluates the performance of TS-CorM across different TSFM backbones, averaged over forecasting horizons within each dataset. TS-CorM improves both metrics on all 40 backbone–dataset pairs. Across backbones, average MSE falls by 4.213%–10.362% and MAE by 1.423%–3.840%. The largest MSE gain occurs on Moirai 2.0 with ETTm2 (22.279%; MAE falls by 8.850%). Weather gains are substantial across all five backbones (5.947%–15.096% MSE), while Electricity shows smaller but consistent reductions (0.567%–3.574% MSE). These results demonstrate the effectiveness of TS-CorM across diverse backbones and datasets, supporting its broad applicability to TSFM adaptation.

Table 2: Forecasting results across different TSFM backbones (lower is better). Results are averaged over $H \in \{96, 192, 336, 720\}$, except TiRex-2† ($H \in \{96, 192\}$), with a lookback window of 512 time steps. **Red**: the best. Blue: the second best within each backbone. Percentages show relative error changes from Origin; negative values indicate lower errors. TS-CorM (offline) consistently improves forecasting performance across all evaluated backbones.

Dataset	Metric	ChronosBolt		TimesFM 3.0		Moirai 2.0		Sundial		TiRex-2†	
		Origin	TS-CorM (offline)	Origin	TS-CorM (offline)	Origin	TS-CorM (offline)	Origin	TS-CorM (offline)	Origin	TS-CorM (offline)
ETTh1	MSE	0.448	**0.412** (-7.98%)	0.430	**0.406** (-5.51%)	0.444	**0.400** (-9.84%)	0.400	**0.388** (-3.04%)	0.393	**0.379** (-3.47%)
	MAE	0.419	**0.410** (-2.10%)	0.412	**0.407** (-1.07%)	0.415	**0.407** (-1.89%)	0.412	**0.405** (-1.75%)	0.387	**0.384** (-0.67%)
ETTh2	MSE	0.367	**0.341** (-7.00%)	0.377	**0.348** (-7.61%)	0.388	**0.346** (-11.04%)	0.344	**0.328** (-4.65%)	0.314	**0.304** (-3.44%)
	MAE	0.380	**0.372** (-1.88%)	0.391	**0.380** (-2.73%)	0.393	**0.378** (-3.77%)	0.380	**0.372** (-2.03%)	0.342	**0.341** (-0.26%)
ETTm1	MSE	0.421	**0.366** (-13.08%)	0.371	**0.347** (-6.36%)	0.446	**0.378** (-15.32%)	0.369	**0.349** (-5.50%)	0.342	**0.315** (-7.82%)
	MAE	0.383	**0.365** (-4.73%)	0.365	**0.359** (-1.59%)	0.393	**0.369** (-6.08%)	0.369	**0.361** (-2.31%)	0.341	**0.333** (-2.30%)
ETTm2	MSE	0.290	**0.260** (-10.48%)	0.323	**0.300** (-7.22%)	0.357	**0.277** (-22.28%)	0.276	**0.260** (-5.73%)	0.213	**0.203** (-4.71%)
	MAE	0.317	**0.305** (-4.03%)	0.319	**0.311** (-2.53%)	0.346	**0.316** (-8.85%)	0.317	**0.310** (-2.37%)	0.268	**0.262** (-2.11%)
Exchange	MSE	0.371	**0.348** (-6.26%)	0.379	**0.327** (-13.71%)	0.380	**0.372** (-2.22%)	0.555	**0.530** (-4.38%)	0.131	**0.129** (-1.46%)
	MAE	0.412	**0.397** (-3.72%)	0.418	**0.390** (-6.48%)	0.411	**0.407** (-0.80%)	0.494	**0.482** (-2.39%)	0.250	**0.249** (-0.48%)
Weather	MSE	0.263	**0.228** (-13.54%)	0.739	**0.677** (-8.39%)	0.269	**0.228** (-15.10%)	0.244	**0.229** (-5.95%)	0.188	**0.173** (-7.78%)
	MAE	0.276	**0.255** (-7.86%)	0.290	**0.272** (-6.04%)	0.274	**0.255** (-6.91%)	0.269	**0.260** (-3.48%)	0.215	**0.208** (-3.30%)
Electricity	MSE	0.159	**0.153** (-3.57%)	0.187	**0.186** (-0.96%)	0.162	**0.160** (-1.69%)	0.148	**0.147** (-0.57%)	0.138	**0.137** (-0.78%)
	MAE	0.244	**0.239** (-2.14%)	0.248	**0.247** (-0.56%)	0.247	**0.244** (-1.02%)	0.242	**0.240** (-0.83%)	0.226	**0.224** (-0.82%)
Traffic	MSE	0.435	**0.401** (-7.73%)	0.392	**0.386** (-1.48%)	0.388	**0.380** (-1.96%)	0.461	**0.416** (-9.66%)	0.374	**0.364** (-2.50%)
	MAE	0.263	**0.253** (-3.73%)	0.241	**0.240** (-0.32%)	0.242	**0.240** (-1.07%)	0.286	**0.262** (-8.25%)	0.237	**0.231** (-2.17%)
Average	MSE	0.344	**0.314** (-8.90%)	0.400	**0.372** (-6.91%)	0.354	**0.318** (-10.36%)	0.349	**0.331** (-5.31%)	0.261	**0.250** (-4.21%)
	MAE	0.337	**0.324** (-3.67%)	0.335	**0.326** (-2.82%)	0.340	**0.327** (-3.84%)	0.346	**0.337** (-2.81%)	0.283	**0.279** (-1.42%)

4.4 Ablation Studies

Figure 3 reports ablations of three TS-CorM design choices. The full model achieves the lowest mean MSE/MAE (0.3162/0.3435). Replacing the correction dictionary with direct residual targets increases mean MSE by 8.91% and MAE by 4.03%. This result identifies the correction dictionary as a major contributor to performance. Removing either dictionary component also raises mean errors, supporting the complementary roles of history-derived and fixed smooth directions. Removing the forecast or history embedding increases MSE on all six datasets by 0.29%–1.44% and 1.39%–4.31%, respectively, supporting their joint use for memory addressing. Global-only and independent per-channel Slow Memory also raise MSE on all six datasets, supporting a shared mapping with channel-specific corrections. Appendix Table 6 reports dataset-level results.

Mean across datasets

Design	Configuration	MSE	MAE
	Full	**0.3162**	**0.3435**
Correction dictionary	w/o dictionary	0.3443	0.3573
	w/o historical directions	0.3383	0.3512
	w/o DCT directions	0.3203	0.3465
Forecast-state addressing	w/o forecast embedding	0.3192	0.3460
	w/o history embedding	0.3240	0.3475
Memory sharing	w/o local memory	0.3198	0.3457
	w/o shared memory	0.3225	0.3463

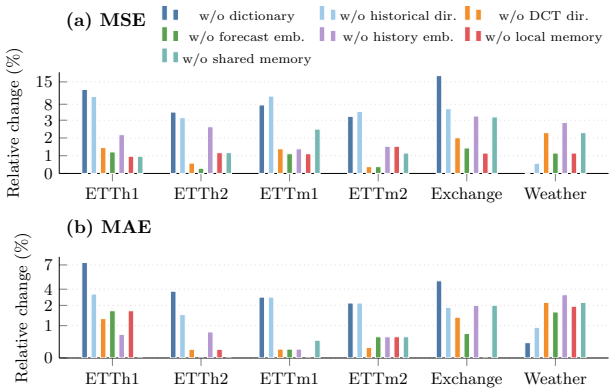


Figure 3: Ablation of TS-CorM design choices. Left: average MSE and MAE across six datasets. Right: per-dataset relative changes after removing individual components. Positive values indicate degradation relative to the full configuration. Results are averaged over forecasting horizons. The vertical axes are compressed above 3% for MSE and 1.5% for MAE.

4.5 Effectiveness of Fast Memory Updates

Figure 4 compares TS-CorM with and without online Fast Memory on ChronosBolt. Electricity and Traffic are excluded due to the computational cost of their large channel counts. With the backbone, representations, and Slow Memory frozen, Fast Memory improves MSE in 21/24 settings and both metrics in 16/24, reducing average MSE by 0.565% and MAE by 0.061%. ETTh1 and ETTm1 improve on both metrics at every horizon, with the largest MSE reduction reaching 1.561% on ETTh1 at $H = 720$. These results suggest that Fast Memory can use delayed feedback to track evolving correction needs, improving upon the offline variant without backpropagation. The stronger aggregate MSE gain is consistent with the squared-error objectives used for feedback encoding and memory writing.

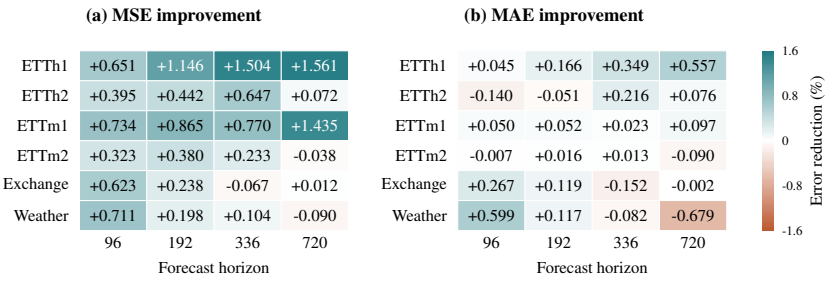


Figure 4: Online Fast Memory gains across 24 ChronosBolt settings covering six datasets and four forecast horizons. Cells show relative error reductions (%). Positive values (teal) indicate improvement; negative values (orange) indicate degradation. Full metrics appear in Table 7.

4.6 Efficiency Analysis

We evaluate inference efficiency with batch size 1 and context length 512. As shown in Table 3, TS-CorM adds only 2.343 ms (2.76%) to ChronosBolt inference, increasing latency from 84.880

Table 3: Inference latency comparison (ms/query) for TS-Memory and TS-CorM on ChronosBolt. The unadapted backbone is included as a reference.

Horizon	End-to-end inference			Adapter-only inference	
	ChronosBolt	+ TS-CorM	+ TS-Memory	TS-CorM	TS-Memory
96	32.535	34.812	35.059	2.277	2.524
192	48.391	50.726	50.972	2.335	2.581
336	96.386	98.751	98.959	2.365	2.573
720	191.820	194.237	194.619	2.417	2.799
Overall	84.880	87.223	87.490	2.343	2.610

to 87.223 ms. Its adapter-only latency[1] is 10.23% lower than TS-Memory (2.343 vs. 2.610 ms) and remains stable across forecast horizons. Most of this overhead comes from constructing the correction dictionary, while memory readout itself is lightweight.

Table 4 compares fitted adaptation parameters, excluding the frozen backbone and online state. With 1536-dimensional keys and 30-dimensional values, the shared and channel-specific Slow Memories contain 92.16K parameters for a single channel, compared with 5.47M for TS-Memory Base. TS-CorM therefore uses 98.32% fewer adaptation parameters while achieving lower forecasting error, demonstrating efficient adaptation in both computation and model size.

Table 4: Adaptation parameter counts, excluding the frozen backbone and online state.

Method	Shared	Per channel
TS-CorM (offline)	0.04608M	**0.04608M**
TS-Memory Base	5.470M	5.470M

4.7 Hyperparameter Sensitivity

We examine the number of DCT directions and the local and global ridge penalties, reporting averaged performance over six datasets at $H = 96$ (Figure 5). Increasing DCT directions steadily lowers MSE, while MAE is minimized at 12 and increases thereafter; expanding from 12 to 48 directions reduces MSE by only 0.03% while increasing MAE by 0.11% and incurring additional computation, suggesting that 12 directions already capture most useful smooth correction structure. For the local ridge penalty, increasing λ_L from 2 to 20 reduces MAE by 0.46% at the cost of only a 0.38% increase in MSE, while stronger regularization degrades both metrics. The global penalty shows the same trade-off: $\lambda_G = 1$ reduces MAE by 0.46% relative to 0.1 with only a 0.41% MSE increase, whereas larger values again worsen both errors. This pattern indicates that moderate regularization balances aggressive correction fitting against stable generalization.

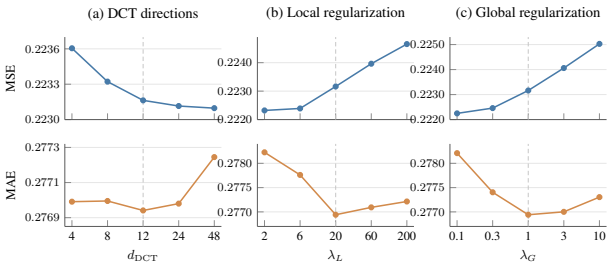


Figure 5: Hyperparameter sensitivity at $H = 96$, averaged over six datasets. The three panels vary the number of DCT directions, λ_L, and λ_G, respectively. Dashed lines mark the adopted values.

5 Conclusion

We introduced TS-CorM, a structured fast–slow corrective memory for adapting frozen time-series foundation models. TS-CorM represents forecast corrections with compact coefficients, addresses them using joint history–forecast states, and organizes their associations in explicit memory matrices. Slow Memory consolidates offline domain knowledge through shared and channel-specific mappings, while Fast Memory incorporates newly observed feedback through recursive online updates without backpropagation. Extensive experiments across multiple TSFM backbones and forecasting benchmarks show consistent improvements over frozen models and strong adaptation baselines, while requiring substantially fewer adaptation parameters and lower inference latency. These results suggest that explicit corrective memory provides a simple and efficient path for both offline and continual adaptation of pretrained time-series forecasters. Future work may explore richer memory representations, online update mechanisms, and broader forms of feedback for continual TSFM adaptation.

[1] Adapter-only latency is estimated by subtracting ChronosBolt latency from the corresponding end-to-end latency.

AI USE STATEMENT

AI Use Statement. Generative AI tools, including OpenAI Codex, were used to assist with translation, academic writing and editing, literature search and summarization, terminology refinement, presentation and interpretation of experimental results, and LaTeX preparation. The tools were not used to originate the research ideas, formulate the methodology, or design the experiments; all scientific concepts, methodological decisions, and experimental procedures were developed by the authors. AI assistance in experimental interpretation was limited to helping summarize and articulate observed results rather than determining scientific conclusions. All AI-assisted text, analyses, and references were reviewed and verified by the authors for correctness, consistency with the implemented methods and reported results, and potential citation or attribution issues. The authors take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

We have taken several measures to ensure the reproducibility of our results. All code and datasets are publicly available through an anonymous repository to facilitate independent replication and verification. The paper provides detailed descriptions of the experimental procedures, model configurations, and hardware settings, and all datasets used in our experiments are publicly accessible. We believe these resources and implementation details provide sufficient support for reproducing our results and facilitating future research.

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. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2403.07815.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting. *arXiv preprint arXiv:2510.15821*, 2025. URL https://arxiv.org/abs/2510.15821.

Andreas Auer, Patrick Podest, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. TiRex: Zero-shot forecasting across long and short horizons with enhanced in-context learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://arxiv.org/abs/2505.23719.

Abdelhakim Benechehab, Vasilii Feofanov, Giuseppe Paolo, Albert Thomas, Maurizio Filippone, and Balázs Kégl. AdaPTS: Adapting univariate foundation models to probabilistic multivariate time series forecasting. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 3731–3748, 2025. URL https://proceedings.mlr.press/v267/benechehab25a.html.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 10148–10167, 2024. URL https://proceedings.mlr.press/v235/das24c.html.

Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam H. Nguyen, Wesley M. Gifford, Chandra Reddy, and Jayant Kalagnanam. Tiny time mixers (TTMs): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2401.03955.

Azul Garza, Cristian Challu, and Max Mergenthaler-Canseco. TimeGPT-1. *arXiv preprint arXiv:2310.03589*, 2023. URL https://arxiv.org/abs/2310.03589.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A family of open time-series foundation models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 16115–16152, 2024. URL https://proceedings.mlr.press/v235/goswami24a.html.

Sungwon Han, Seungeon Lee, Meeyoung Cha, Sercan O Arik, and Jinsung Yoon. Retrieval augmented time series forecasting. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 21774–21797, 2025. URL https://proceedings.mlr.press/v267/han25d.html.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Ayush Jain and Rajat Sen. TimesFM-3: A zero-shot foundation model for multivariate forecasting. Google Research Blog, August 2026. URL https://www.research.google/blog/timesfm-3-a-zero-shot-foundation-model-for-multivariate-forecasting/.

Chenghao Liu, Taha Aksu, Juncheng Liu, Xu Liu, Hanshu Yan, Quang Pham, Silvio Savarese, Doyen Sahoo, Caiming Xiong, and Junnan Li. Moirai 2.0: When less is more for time series forecasting. *arXiv preprint arXiv:2511.11698*, 2025a. URL https://arxiv.org/abs/2511.11698.

Xu Liu, Juncheng Liu, Gerald Woo, Taha Aksu, Yuxuan Liang, Roger Zimmermann, Chenghao Liu, Junnan Li, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Moirai-MoE: Empowering time series foundation models with sparse mixture of experts. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 38940–38962, 2025b. URL https://proceedings.mlr.press/v267/liu25an.html.

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 *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/2ea18fdc667e0ef2ad82b2b4d65147ad-Paper-Conference.pdf.

Yong Liu, Guo Qin, Zhiyuan Shi, Zhi Chen, Caiyin Yang, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Sundial: A family of highly capable time series foundation models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 39295–39317, 2025c. URL https://proceedings.mlr.press/v267/liu25be.html.

Sisuo Lyu, Siru Zhong, Tiegang Chen, Weilin Ruan, Qingxiang Liu, Taiqiang Lv, Qingsong Wen, Raymond Chi-Wing Wong, and Yuxuan Liang. TS-Memory: Plug-and-play memory for time series foundation models. *arXiv preprint arXiv:2602.11550*, 2026. URL https://arxiv.org/abs/2602.11550.

Kanghui Ning, Zijie Pan, Yu Liu, Yushan Jiang, James Yiming Zhang, Kashif Rasul, Anderson Schneider, Lintao Ma, Yuriy Nevmyvaka, and Dongjin Song. TS-RAG: Retrieval-augmented generation based time series foundation models are stronger zero-shot forecaster. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://arxiv.org/abs/2503.07649.

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1ecqn4YwB.

Patrick Podest, Marco Pichler, Elias Bürger, Levente Zólyomi, Bernhard Voggenberger, Wilhelm Berghammer, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. TiRex-2: Generalizing TiRex to multivariate data and streaming. *arXiv preprint arXiv:2607.01204*, 2026. URL https://arxiv.org/abs/2607.01204.

Zhongzheng Qiao, Chenghao Liu, Yiming Zhang, Ming Jin, Quang Pham, Qingsong Wen, P. N. Suganthan, Xudong Jiang, and Savitha Ramasamy. Multi-scale finetuning for encoder-based time series foundation models. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://arxiv.org/abs/2506.14087.

Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. Lag-Llama: Towards foundation models for probabilistic time series forecasting. *arXiv preprint arXiv:2310.08278*, 2023. URL https://arxiv.org/abs/2310.08278.

Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-MoE: Billion-scale time series foundation models with mixture of experts. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2409.16040.

Kutay Tire, Ege Onur Taga, Muhammed Emrullah Ildiz, and Samet Oymak. Retrieval augmented time series forecasting. In *Proceedings of the 29th International Conference on Artificial Intelligence and Statistics*, volume 300, pp. 4294–4302, 2026. URL https://proceedings.mlr.press/v300/tire26a.html.

Ke Alexander Wang, Jiaxin Shi, and Emily B. Fox. Test-time regression: a unifying framework for designing sequence models with associative memory. *arXiv preprint arXiv:2501.12352*, 2025. URL https://arxiv.org/abs/2501.12352.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 53140–53164, 2024. URL https://proceedings.mlr.press/v235/woo24a.html.

Xiaoyun Yu, Li Fan, Xiangfei Qiu, Nanqing Dong, Yonggui Huang, Honggang Qi, Geguang Pu, Wanli Ouyang, Xi Chen, and Jilin Hu. MEMTS: Internalizing domain knowledge via parameterized memory for retrieval-free domain adaptation of time series foundation models. *arXiv preprint arXiv:2602.13783*, 2026. URL https://arxiv.org/abs/2602.13783.

Huanyu Zhang, Chang Xu, Yi-Fan Zhang, Zhang Zhang, Liang Wang, and Jiang Bian. TimeRAF: Retrieval-augmented foundation model for zero-shot time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 2025. doi: 10.1109/TKDE.2025.3579137. URL https://doi.org/10.1109/TKDE.2025.3579137.

Lifan Zhao, Yanyan Shen, Zhaoyang Liu, Xue Wang, and Jiaji Deng. Less is more: Unlocking specialization of time series foundation models via structured pruning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://arxiv.org/abs/2505.23195.

A Implementation and evaluation protocol

This appendix provides additional details of our experiments. Vectors are columns throughout. Memory keys are forecast states, $k_i = z_i$, and target values are oracle coefficients, $v_i^\star = \theta_i^\star$. A readout v directly specifies a coefficient vector, with normalized correction $F_t v$.

A.1 Embedding-based forecast-state construction

The forecast state in Equation (4) concatenates two 768-dimensional REG-token embeddings, giving 1536 coordinates. The history branch encodes the historical context with the frozen Chronos encoder and takes the last REG token from its final output. The forecast branch feeds the FM forecast into the same encoder and extracts the corresponding last REG token. The two vectors are concatenated in history–forecast order. No observed target is used as input to either branch. The forecast state is used directly as the memory key in Equation (5), without an appended intercept or an additional feature projection.

A.2 Dataset details and splits

Following the dataset descriptions in Liu et al. (2024), the four ETT benchmarks contain seven transformer-related variables, including oil temperature: ETTh1/2 are sampled hourly and ETTm1/2 every 15 minutes. Exchange contains daily exchange rates for eight countries. Weather records 21 meteorological variables at 10-minute intervals. Electricity (ECL) contains hourly electricity consumption from 321 clients, and Traffic contains hourly road occupancy measured by 862 sensors in the San Francisco Bay Area. The four ETT subsets are evaluated separately, giving eight benchmark datasets in total.

Table 5 gives the original chronological split boundaries in the source series. Indices are zero-based, and every interval is left-closed and right-open: $[a, b)$ includes index a and excludes b.

Table 5: Original train, validation, and test intervals (zero-based, half-open).

Dataset	**Train**	**Validation**	**Test**
ETTh1, ETTh2	$[0, 8640)$	$[8640, 11520)$	$[11520, 14400)$
ETTm1, ETTm2	$[0, 34560)$	$[34560, 46080)$	$[46080, 57600)$
Exchange	$[0, 5311)$	$[5311, 6071)$	$[6071, 7588)$
Weather	$[0, 36887)$	$[36887, 42157)$	$[42157, 52696)$
Electricity	$[0, 18412)$	$[18412, 21044)$	$[21044, 26304)$
Traffic	$[0, 12280)$	$[12280, 14036)$	$[14036, 17544)$

The ETT datasets use fixed 12/4/4-month train/validation/test intervals, with 30 days per month. Observations beyond the listed test endpoint are not scored. For the other datasets, a series of length L uses $\lfloor 0.7L \rfloor$ training observations and $\lfloor 0.2L \rfloor$ test observations; the intervening observations form the validation interval, giving an approximately 70/10/20 split. These original boundaries are distinct from the subsequent Slow Memory refit on training and validation data described below.

B Detailed Ablation and Online Adaptation Results

Tables 6 and 7 provide the complete numerical results underlying Figures 3 and 4, respectively.

Table 6: Dataset-level ablations, averaged over horizons 96, 192, 336, and 720. Each cell reports MSE/MAE. Red: the best. Blue: the second best.

Configuration	ETTh1	ETTh2	ETTm1	ETTm2	Exchange	Weather
Full	**0.412** / **0.411**	**0.342** / **0.373**	**0.360** / **0.364**	**0.261** / **0.305**	**0.348** / **0.397**	**0.174** / **0.211**
w/o dictionary (direct residual)	0.464 / 0.441	0.361 / 0.387	0.388 / 0.375	0.272 / 0.312	0.407 / 0.417	**0.174** / 0.212
w/o historical directions	0.455 / 0.425	0.355 / 0.378	0.398 / 0.375	0.276 / 0.312	0.371 / 0.404	0.175 / 0.213
w/o DCT directions	0.418 / 0.416	0.344 / 0.374	0.365 / 0.365	0.262 / 0.306	0.355 / 0.402	0.178 / 0.216
w/o forecast embedding	0.417 / 0.417	0.343 / 0.373	0.364 / 0.365	0.262 / 0.307	0.353 / 0.400	0.176 / 0.214
w/o history embedding	0.421 / 0.414	0.351 / 0.376	0.365 / 0.365	0.265 / 0.307	0.363 / 0.405	0.179 / 0.218
w/o local memory	0.416 / 0.417	0.346 / 0.374	0.364 / 0.364	0.265 / 0.307	0.352 / **0.397**	0.176 / 0.215
w/o shared memory	0.416 / 0.411	0.346 / 0.373	0.369 / 0.366	0.264 / 0.307	0.362 / 0.405	0.178 / 0.216

Table 7: TS-CorM (offline) versus TS-CorM (online) on ChronosBolt. Positive Δ denotes error reduction relative to Slow Memory. **Red**: the best. Blue: the second best.

Dataset	Horizon	TS-CorM (offline)		TS-CorM (online)		Improvement (%)	
		MSE	MAE	MSE	MAE	Δ MSE	Δ MAE
ETTh1	96	0.359	0.372	**0.357**	**0.372**	+0.651	+0.045
	192	0.411	0.403	**0.406**	**0.402**	+1.146	+0.166
	336	0.440	0.422	**0.433**	**0.420**	+1.504	+0.349
	720	0.445	0.442	**0.438**	**0.440**	+1.561	+0.557
ETTh2	96	0.278	**0.323**	**0.277**	0.323	+0.395	-0.140
	192	0.340	**0.366**	**0.338**	0.366	+0.442	-0.051
	336	0.369	0.393	**0.367**	**0.392**	+0.647	+0.216
	720	0.381	0.411	**0.381**	**0.411**	+0.072	+0.076
ETTm1	96	0.306	0.325	**0.304**	**0.325**	+0.734	+0.050
	192	0.347	0.353	**0.344**	**0.353**	+0.865	+0.052
	336	0.385	0.376	**0.382**	**0.376**	+0.770	+0.023
	720	0.449	0.413	**0.443**	**0.412**	+1.435	+0.097
ETTm2	96	0.168	**0.241**	**0.168**	0.241	+0.323	-0.007
	192	0.233	0.286	**0.232**	**0.286**	+0.380	+0.016
	336	0.292	0.326	**0.291**	**0.326**	+0.233	+0.013
	720	**0.376**	**0.381**	0.376	0.382	-0.038	-0.090
Exchange	96	0.093	0.213	**0.092**	**0.212**	+0.623	+0.267
	192	0.183	0.307	**0.183**	**0.306**	+0.238	+0.119
	336	**0.332**	**0.419**	0.332	0.419	-0.067	-0.152
	720	0.829	**0.686**	**0.829**	0.686	+0.012	-0.002
Weather	96	0.158	0.197	**0.157**	**0.196**	+0.711	+0.599
	192	0.198	0.237	**0.197**	**0.236**	+0.198	+0.117
	336	0.251	**0.276**	**0.250**	0.276	+0.104	-0.082
	720	**0.320**	**0.324**	0.321	0.326	-0.090	-0.679
Average	–	0.331	0.354	**0.329**	**0.354**	+0.565	+0.061

C Formal definition of the memory framework

The four operations define a common interface for TSFM adaptation:

$$\begin{aligned} M_0 &= \mathcal{C}(D_{\text{off}}; f_\theta), \\ M_t &= \mathcal{W}(M_{t-1}, \mathcal{I}_t; f_\theta), \\ \hat{v}_t &= \mathcal{R}(x_t, M_t; f_\theta), \\ \widehat{y}_t &= \mathcal{E}(x_t, \hat{v}_t; f_\theta). \end{aligned} \tag{12}$$

Here, f_θ is the pretrained TSFM, D_{off} is the domain data available for initialization, and x_t is the input at forecast origin t. Construction $\mathcal{C}$ initializes M_0. For $t \geq 1$, writing $\mathcal{W}$ updates memory using information $\mathcal{I}_t$ available before the forecast is issued. Reading $\mathcal{R}$ produces a method-dependent readout $\widehat{v}_t$, which execution $\mathcal{E}$ uses with x_t to generate $\widehat{y}_t$. Memory may consist of examples, adapted parameters, or learned representations. For methods without online updates, writing is the identity on the memory state. These functional roles need not correspond to separate implementation modules, and every operation uses only information available at the forecast origin.
Table 8 summarizes the inputs and outputs of these operations.

Table 8: Inputs and outputs of the memory operations.

Operation	Inputs	Output
Construct $\mathcal{C}$	Offline domain data D_{off}	Initial memory M_0
Write $\mathcal{W}$	Previous memory M_{t-1}, new information $\mathcal{I}_t$	Updated memory M_t
Read $\mathcal{R}$	Forecasting input x_t, memory M_t	Memory readout $\widehat{v}_t$
Execute $\mathcal{E}$	Forecasting input x_t, readout $\widehat{v}_t$	Forecast $\widehat{y}_t$