Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Memory Mechanisms in Multivariate Time Series: A Survey of Architectures, Retrieval, and Evolution

Sibo Qi
Macao Polytechnic University
Peng Chen
Xihua University
Wuman Luo
luowuman@mpu.edu.mo

Macao Polytechnic University

Research Article

Keywords: multivariate time-series forecasting, memory mechanisms, retrieval-augmented forecasting, continual learning, concept drift, foundation models

Posted Date: July 7th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10171809/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Memory Mechanisms in Multivariate Time Series: A Survey of Architectures, Retrieval, and Evolution

Sibo Qi [1], Peng Chen [2*], Wuman Luo [1*]

[1]Faculty of Applied Sciences, Macao Polytechnic University, Macao, China.

[2]School of Computer and Software Engineering, Xihua University, Chengdu, China.

*Corresponding author(s). E-mail(s): chenpeng@mail.xhu.edu.cn; luowuman@mpu.edu.mo;

Contributing authors: sibo.qi@mpu.edu.mo;

Abstract

Multivariate time-series (MTS) forecasting models must retain useful temporal and cross-variable experience while adapting to non-stationarity, concept drift, and evolving relations. This survey studies that requirement through *memory mechanisms*: identifiable model or system components that preserve information beyond the instantaneous input transformation and use it to condition, adapt, or verify future forecasts. We formalize memory mechanisms for MTS forecasting as the interaction between a retained memory state, an access function, and an update function. This definition motivates a three-part organization. Architecture-level mechanisms describe where retained information is represented; retrieval mechanisms describe how stored experience is read; and evolution mechanisms describe how memory is refreshed, consolidated, or forgotten under drift. We use this framework to review recent MTS forecasting literature, separate exposed memory mechanisms from ordinary attention or static context modeling, and synthesize open challenges in latency, temporal matching, robustness, evaluation, hallucination, and storage theory. The resulting roadmap links each challenge to concrete design directions for memory mechanisms with auditable read, update, deletion, and verification behavior.

Keywords: multivariate time-series forecasting, memory mechanisms, retrieval-augmented forecasting, continual learning, concept drift, foundation models

1 Introduction

Multivariate time-series (MTS) forecasting underpins operational decisions in energy, traffic, finance, climate, healthcare, manufacturing, and cloud systems. The forecasting target is rarely a single smooth trajectory: real deployments contain correlated variables, regime changes, sensor shifts, missing observations, changing sampling patterns, and external events. These conditions make *memory mechanisms* a recurring design concern. A useful forecasting model benefits from retaining historical dynamics and cross-variable relations, while also deciding when old regimes should be reused, updated, or forgotten.

Most deep MTS forecasting has treated memory mechanisms implicitly: recurrent and state-space designs retain history in learned states or parameterized dynamics Kraus et al. (2025); Ma et al. (2025); Li et al. (2025); Transformer and patching models hold context in tokens, attention patterns, and shared projections Nie et al. (2022); Liu et al. (2025a); Cirstea et al. (2022); Lu et al. (2024); and graph, frequency, and foundation-model designs encode structured relations or pretrained temporal priors Yi et al. (2023); Woo et al. (2024); Darlow et al. (2024). These implicit memory mechanisms are computationally convenient and have enabled competitive benchmark performance, but they have three structural limits:

1. **Capacity entanglement.** Memory capacity is entangled with model size and training data—expanding memory often requires retraining a larger model.
2. **Inspectability.** Stored experience is difficult to inspect or edit after training, making it hard to determine what the model retains about specific regimes or variables.
3. **Post-deployment rigidity.** Adaptation to new regimes often depends on fine-tuning, retraining, or ad hoc normalization rather than a principled memory update policy.

A second line of work makes memory mechanisms more explicit. Retrieval-augmented forecasters read historical segments or external knowledge at inference time Ning et al. (2025); Zhang et al. (2024a,b); Xiao et al. (2025); continual and adaptive methods preserve replay buffers, slow-fast states, or evolving knowledge bases for non-stationary streams Missel and Wang (2025); Pham et al. (2022); He and Sick (2021); and foundation-model forecasters import broad pretrained priors and cross-modal context into time-series modeling Woo et al. (2024); Darlow et al. (2024); Liu et al. (2025b). The common movement is not a single architecture, but a shift from forecasting models that primarily *contain* retained information toward systems that expose how retained information is represented, read, or updated.

This survey therefore uses *memory mechanisms* as its organizing concept. Rather than ask only which backbone, scale, retrieval pipeline, or adaptation algorithm a method uses, we ask where retained information resides, how it is accessed, and whether it changes after deployment. These three questions motivate the reader-facing organization around architecture-level, retrieval, and evolution mechanisms. This view reveals a recurring imbalance: architecture-level mechanisms are common in MTS forecasting, while explicit retrieval and post-deployment evolution mechanisms are less consistently specified.

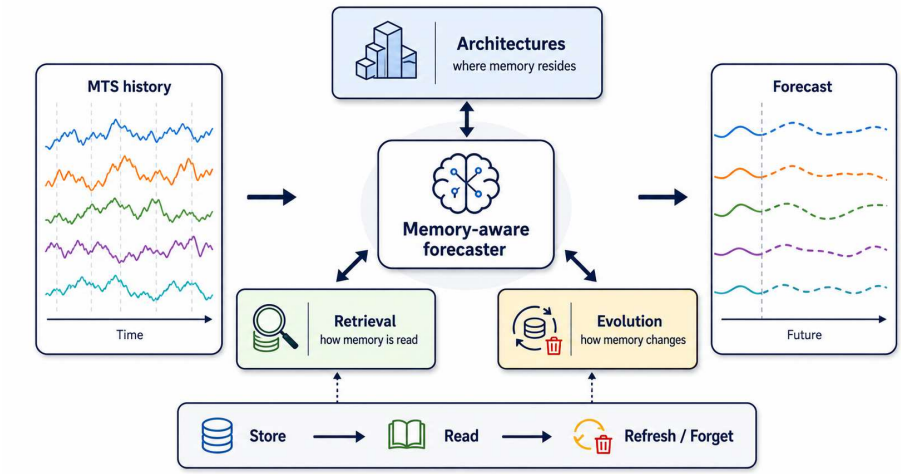


Fig. 1: Overview of the memory-mechanism view used in this survey. Multivariate histories can be retained inside forecasting architectures, read through retrieval mechanisms, and updated through evolution mechanisms, yielding forecasts whose memory assumptions are easier to inspect and maintain.

Section 2 gives the formal definition and review protocol before the method survey begins. Figure 1 summarizes the reader-facing view: an MTS forecaster can use retained information through architectural representation, explicit retrieval, and post-deployment evolution. The figure also previews why the same term "memory" should not be used loosely. A method may retain information without exposing a retrieval mechanism; it may retrieve examples without specifying how memory is updated; and it may adapt online without giving a controllable deletion or verification policy.

Our contributions are fourfold.

1. **Definition of memory mechanisms.** We define a memory mechanism for MTS forecasting as an identifiable component that retains information beyond the instantaneous input transformation and uses it to condition, adapt, or verify forecasts.
2. **Mechanism taxonomy and boundary rules.** We justify a three-part organization—architecture-level, retrieval, and evolution mechanisms—by linking it to representation, access, and update functions in the formal definition. The resulting boundary rules distinguish exposed memory mechanisms from ordinary attention, patching, context modeling, and static training.
3. **Review protocol and evidence map.** We state the screening criteria, evidence levels, and classification rules in the main text, while leaving detailed query strings and long evidence tables to the appendix.
4. **Challenges-to-roadmap synthesis.** We connect open challenges to concrete research directions for latency-aware retrieval, temporal matching, memory validation, memory-specific evaluation, hallucination control, and storage/update theory.

The remainder of the paper is organized as follows. Section 2 defines memory mechanisms, describes the review protocol, justifies the architecture–retrieval–evolution taxonomy, and positions the survey against adjacent reviews. Sections 3–5 survey architecture-level, retrieval, and evolution mechanisms. Section 6 connects open challenges to a research roadmap and states scope boundaries. Section 7 concludes.

2 Memory Mechanisms and Review Protocol

This section establishes the terminology used throughout the survey. We use *memory mechanism* as the umbrella term and avoid treating phrases such as memory-rich, memory-augmented, memory-aware, or memory-governed as separate technical categories. In this survey, a memory mechanism is an identifiable model or system component that retains information beyond the instantaneous transformation of the current input window and uses that retained information to condition, adapt, or verify multivariate time-series forecasts. The retained information may be encoded in parameters, hidden states, attention contexts, structured operators, external stores, replay buffers, adapters, or evolving statistics, but it must play a functional role in the forecast rather than merely describe a model metaphorically.

For an input window $\mathbf{X}_{t-L+1:t} \in \mathbb{R}^{L\times C}$ with C variables and forecast horizon H, this view can be summarized as:

$$\begin{aligned} \hat{\mathbf{Y}}_{t+1:t+H} &= F_\theta(\mathbf{X}_{t-L+1:t}, \mathbf{z}_t), \\ \mathbf{z}_t &= \rho_\phi(q(\mathbf{X}_{t-L+1:t}), \mathcal{M}_t), \\ \mathcal{M}_{t+1} &= U_\psi(\mathcal{M}_t, \mathbf{X}_{t-L+1:t}, \hat{\mathbf{Y}}_{t+1:t+H}, \mathbf{e}_t). \end{aligned} \tag{1}$$

Here $\mathcal{M}_t$ denotes the retained memory state or store, $q(\cdot)$ constructs a query or conditioning representation from the current window, ρ_ϕ accesses retained information and returns the memory-conditioned signal $\mathbf{z}_t$, and U_ψ updates, refreshes, consolidates, or forgets memory using newly available evidence $\mathbf{e}_t$ such as observations, errors, drift indicators, or metadata. In an architecture-only forecaster, $\mathcal{M}_t$ may be embedded in parameters, recurrent states, patch embeddings, or pretrained context, and ρ_ϕ may be an implicit forward computation. In a retrieval-augmented forecaster, ρ_ϕ becomes an explicit read from a persistent store. In an adaptive or continual forecaster, U_ψ becomes an explicit post-training memory-update policy. Figure 1 gives the corresponding schematic view for MTS forecasting.

2.1 Why Architectures, Retrieval, and Evolution?

The three survey pillars are therefore not arbitrary topic labels; they are projections of Equation 1. **Architecture-level memory mechanisms** ask what $\mathcal{M}_t$ is and where retained information resides inside the forecasting system: learned parameters, recurrent or state-space states, attention contexts, patch representations, graph/frequency operators, adapters, or foundation-model priors. **Retrieval memory mechanisms** ask how retained information is accessed: whether the model constructs a query and reads from a persistent store, index, knowledge base, exemplar set, or memory bank.

Evolution memory mechanisms ask how memory changes after training: whether the system defines a write, refresh, consolidation, replay, drift-adaptation, or forgetting policy through U_ψ.

This operation-level view follows a long-standing convention in differentiable-memory models. Neural Turing Machines and End-to-End Memory Networks do not define memory only by the surrounding network backbone; they separate a retained state or store from operations such as addressing, reading, writing, allocation, and response construction Graves et al. (2016); Sukhbaatar et al. (2015). This precedent supports treating a memory mechanism as the coupling between where information is retained and how it is accessed or updated.

Building on this precedent, our Architecture/Retrieval/Evolution taxonomy adapts the operation-level lens to MTS forecasting. Architecture-level mechanisms capture where forecasting-relevant temporal information is represented, retrieval mechanisms capture explicit access to stored cases, knowledge, contexts, or exemplars, and evolution mechanisms capture post-training update, consolidation, drift adaptation, or forgetting. The three pillars therefore serve as forecasting-specific projections of Equation 1, rather than as imported labels or model-family names.

This organization is useful because many competitive MTS forecasters retain information without implementing all three functions explicitly. PatchTST, Timer-XL, and temporal query architectures store long-range information through patching, attention, or learned context, but they do not read from an external memory store Nie et al. (2022); Liu et al. (2025a); Lin et al. (2025). Similarly, LLM-empowered or foundation-model forecasters often import prior knowledge through pretrained parameters, prompts, adapters, cross-modal alignment, or distillation rather than through a formal retrieval interface Liu et al. (2025b); Benechehab et al. (2025); Liu et al. (2025d); Woo et al. (2024); Darlow et al. (2024); Liu et al. (2025c, 2024). The taxonomy therefore avoids labeling every contextual or adaptive architecture as Retrieval or Evolution. Instead, it asks a sequence of inspectable questions: what is stored, how it is read, how it is updated, when it is forgotten, and how the cost or risk of these operations affects forecasting.

Table 1: Compact taxonomy of memory-mechanism and framing-evidence categories used in this survey. The first five rows name concrete forms that instantiate architectural memory, retrieval memory, and memory evolution; the final row separates contextual survey and benchmark evidence from memory mechanisms.

Category	What is retained	Reader boundary
Implicit parametric/state memory	Learned parameters, hidden states, temporal context windows, attention patterns, latent states, or recurrent and state-space dynamics.	Architectural memory unless the method reads from a persistent external index, exemplar store, knowledge base, or memory bank.
Foundation/LLM context memory	Pretrained forecasting or language-model priors, prompts, adapters, alignment modules, or distilled representations.	Architectural memory unless retrieved evidence is explicitly queried and fused into the forecast.
Structured/ prototype memory	Correlation matrices, graph or frequency operators, probabilistic dependency structures, prototypes, or other relational summaries.	Architectural memory unless the structure becomes an external index or an explicitly updated store.
Retrieval/exemplar memory	Historical segments, examples, documents, knowledge items, adapters, or prototypes selected by a query.	Retrieval memory when a persistent store is read and the retrieved content changes the forecast.
Replay/ consolidation memory	Replay buffers, retained tasks, slow-fast states, representative regimes, or compact consolidated summaries.	Evolution memory when the method defines write, refresh, consolidation, or forgetting behavior after deployment.
Survey/benchmark support	Surveys, datasets, benchmarks, and evaluation context used to interpret the field.	Framing evidence rather than a new memory mechanism.

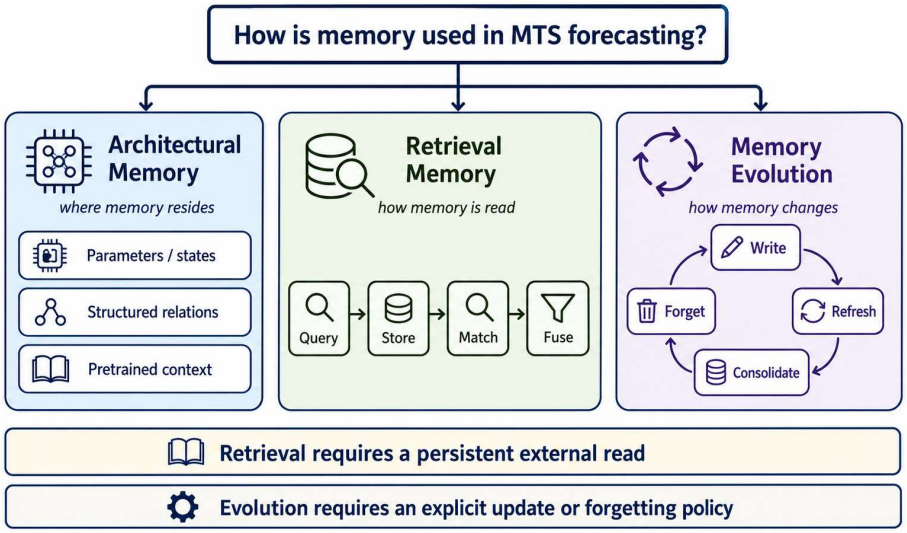


Fig. 2: Reader-facing taxonomy of memory mechanisms in multivariate time-series forecasting. The diagram separates architectural memory, retrieval memory, and memory evolution as parallel operation views, with boundary rules for external reads and explicit update policies.

Table 1 defines five memory-mechanism forms and one framing-evidence category used throughout the survey. Figure 2 provides the reader-facing map behind this table: architectural mechanisms specify where retained information resides, retrieval mechanisms specify how it is read, and evolution mechanisms specify how it changes after deployment. The boundary rules in the figure emphasize that Retrieval requires a persistent external read, while Evolution requires an explicit update or forgetting policy. The detailed paper-level evidence tables and aggregate distribution figures are provided in Appendix B and Appendix C.

2.2 Review Protocol

Because the classification of memory mechanisms depends on implementation details, the screening method is part of the main argument rather than only an appendix item. We used a representative, multi-source review protocol rather than claiming an exhaustive census of all MTS forecasting papers. Candidate papers were collected from recent AI and data-mining conference sources, public scholarly indexes, OpenReview, OpenAlex, CrossRef, ACL Anthology, CVF Open Access, and targeted journal expansion. Search terms combined MTS forecasting with memory-related concepts such as retrieval, external memory, state-space memory, foundation-model context, continual learning, concept drift, replay, adaptation, and forgetting. Classic memory-system papers and adjacent surveys were added as background anchors.

Screening followed three steps. First, candidates were deduplicated by DOI, arXiv identifier, and normalized title. Second, papers were filtered for relevance to

MTS forecasting or for direct conceptual support to memory-system design. Third, mechanism-bearing papers were read at the abstract, method, and when necessary full-text level to decide whether they expose architecture-level, retrieval, or evolution mechanisms. We use three evidence levels in the survey: a closely read core set for method-level mechanism claims and boundary decisions; a broader trend set for aggregate field observations; and background support papers for classical memory architectures, adjacent surveys, benchmarks, and conceptual framing.

The inclusion criteria are deliberately conservative. A paper enters the mechanism discussion when it either proposes an MTS forecasting method with an identifiable memory mechanism, evaluates a memory-relevant MTS forecasting behavior, or provides a directly transferable memory-system primitive such as differentiable reading, replay, or consolidation. Papers are excluded from mechanism claims when memory is only a metaphor, when attention or context is built solely from the current input without a persistent store, when adaptation is only ordinary training without a post-training update policy, or when the work belongs mainly to anomaly detection, classification, imputation, or generation without clear relevance to forecasting memory design. Survey and benchmark papers are used as framing evidence, not as new memory-mechanism instances.

2.3 Scope Boundaries

This survey focuses on multivariate time-series forecasting. Univariate-only methods, classification, anomaly detection, imputation, control, and generation are referenced only when they clarify forecasting memory design. Non-deep statistical forecasting methods are outside the main scope. The primary search window is 2023–2026, with earlier classical memory-system and MTS papers included as conceptual anchors. The resulting taxonomy should be read as a mechanism-level map: it compares store, read, update, refresh, and forget behavior across heterogeneous forecasters, but it does not provide a new quantitative benchmark or a performance leaderboard.

2.4 Positioning Against Adjacent Surveys

After defining memory mechanisms and the review protocol, we situate the survey within adjacent literature. The goal is not to resurvey each neighboring area, but to clarify why memory mechanisms provide a useful organizing axis for MTS forecasting.

Multivariate time-series forecasting surveys. Existing MTS surveys provide broad architectural and benchmark taxonomies organized around model families, forecasting settings, and dataset heterogeneity Benidis et al. (2022); Lim and Zohren (2021); Chen et al. (2023); Song et al. (2024); Shao et al. (2024); Kim et al. (2024); Hajirahimi and Khashei (2023). Representative long-sequence forecasting models such as Informer and Autoformer illustrate the kind of architectural progress these surveys organize: efficient attention, decomposition, and scalable sequence modeling Zhou et al. (2021); Wu et al. (2021). These works are closest to the present survey in application domain, but their primary unit of analysis is usually the forecasting backbone or benchmark setting. A memory-mechanism view asks a different question: whether a method only compresses history into parameters or hidden states, whether it reads

from a persistent external store, and whether it specifies write, refresh, consolidation, or forgetting behavior after deployment.

Time-series foundation model surveys. The rapid emergence of large-scale pretrained forecasters has prompted a growing body of work on time-series foundation models (TSFMs) and LLM-empowered forecasting Wen et al. (2023); Zhang et al. (2024c); Jin et al. (2024a,b); Woo et al. (2024); Darlow et al. (2024); Liu et al. (2025a). These efforts are valuable for understanding pretraining data, scaling, zero-shot transfer, prompt design, and general-purpose architectures. In that literature, memory is often discussed through context length, parameter capacity, prompts, and pretrained representations. Our survey treats those mechanisms as one part of a broader memory taxonomy and asks whether TSFMs expose memory as an inspectable interface: what is retained, how it is accessed, and whether it can be updated, consolidated, or forgotten after deployment.

Retrieval-augmented generation surveys. The RAG literature has produced mature taxonomies for retrieval over text and document stores, covering index construction, query reformulation, fusion strategies, and update policies Gao et al. (2023); Lewis et al. (2020). These taxonomies provide useful design language, but MTS retrieval changes the matching problem: temporal similarity is multidimensional and phase-dependent, queries are continuous-valued, and channel structure can matter as much as temporal proximity. Retrieval-augmented MTS methods such as TS-RAG, TimeRAF, GTR, nearest-neighbor forecasting, and retrieval-augmented LLM forecasting illustrate the promise of external memory while also exposing temporal-specific risks Ning et al. (2025); Zhang et al. (2024a); Cao et al. (2026); Zhang et al. (2024b); Tire et al. (2024); Xiao et al. (2025). Our Retrieval pillar adapts the RAG vocabulary to temporal stores, continuous queries, phase/channel matching, fusion choices, and stale-index diagnostics.

Continual learning and memory evolution. Continual and online learning surveys provide mechanisms for experience replay, regularization-based consolidation, and dynamic architecture expansion Wang et al. (2024); De Lange et al. (2022); Parisi et al. (2019). Concept-drift surveys further clarify how non-stationary data streams challenge static predictors Gama et al. (2014); Suárez-Cetrulo et al. (2023); Hinder et al. (2024); Song et al. (2023). In MTS forecasting, these ideas appear in slow-fast adaptation, online forecasting with replay, adaptive regression, and evolving knowledge bases Missel and Wang (2025); Pham et al. (2022); He and Sick (2021); Julian et al. (2025). The temporal setting adds structure to the general continual-learning problem: drift may occur at multiple time scales, channel correlations can change independently, and forgetting must distinguish reusable seasonal patterns from obsolete regimes. Our Evolution pillar therefore frames memory maintenance as a forecasting-specific problem of write, refresh, consolidation, and deletion.

Classical memory mechanisms. Classical memory architectures provide the vocabulary behind the operational view used in this paper. Neural Turing Machines, Differentiable Neural Computers, End-to-End Memory Networks, and modern Hopfield Networks made read, write, addressing, and associative retrieval explicit neural operations Graves et al. (2014, 2016); Sukhbaatar et al. (2015); Ramsauer et al. (2021);

LSTM introduced gated state retention as a foundational sequence-modeling primitive Hochreiter and Schmidhuber (1997). More recent work on memory-augmented neural networks and Memorizing Transformers extends these ideas to larger-scale sequence modeling Khosla et al. (2023); Wu et al. (2022); Lewis et al. (2020). We use these mechanisms as conceptual anchors rather than as evidence that current MTS forecasters implement a full external-memory interface.

Table 2 summarizes the closest-survey boundary. The distinction is not that adjacent surveys are incomplete; rather, they organize neighboring areas around different units of analysis.

Table 2: Positioning against adjacent survey families. Existing surveys provide the application, retrieval, and continual-learning foundations; this survey uses memory mechanisms as the cross-cutting comparison axis.

Adjacent survey family	Existing emphasis	Distinction in this survey
MTS, TSFM, and LLM forecasting surveys Benidis et al. (2022); Lim and Zohren (2021); Song et al. (2024); Shao et al. (2024); Kim et al. (2024); Wen et al. (2023); Woo et al. (2024); Liu et al. (2025a)	Model families, benchmarks, scaling, transfer, context design, and foundation-model capability.	Compares store, read, write, refresh, and forget operations across backbones.
RAG surveys Gao et al. (2023); Lewis et al. (2020)	Text or document stores, query reformulation, retrieval pipelines, fusion, and update policy.	Adapts retrieval to temporal stores, continuous queries, phase/channel matching, and stale-index diagnostics.
Continual-learning, drift, and memory-augmented surveys Wang et al. (2024); De Lange et al. (2022); Parisi et al. (2019); Gama et al. (2014); Suárez-Cetrulo et al. (2023); Hinder et al. (2024); Khosla et al. (2023); Wu et al. (2022)	Replay, regularization, expansion, drift detection, and read-write memory architectures.	Frames MTS memory maintenance as write, refresh, consolidation, forgetting, and adoption of explicit memory interfaces.

Positioning of this survey. The present survey connects these neighboring communities through a three-pillar framework: Architecture-level mechanisms explain where retained information resides, Retrieval explains how external memory is read,

and Evolution explains how memory is written, refreshed, consolidated, or forgotten. This framing is especially important for MTS forecasting because ordinary attention, prompt context, normalization, or hidden-state recurrence can otherwise be over-labeled as retrieval or memory evolution. The result should be read as a mechanism-level map rather than a performance leaderboard or exhaustive census; its main purpose is to make memory assumptions comparable across otherwise heterogeneous forecasting methods.

Summary. The paper uses one umbrella term—memory mechanism—and grounds it in the retained state $\mathcal{M}_t$, access function ρ_ϕ, and update function U_ψ in Equation 1. Architectures, Retrieval, and Evolution correspond to where memory resides, how it is read, and how it changes. The review protocol is included in the main text because these boundary decisions determine which papers support mechanism-level claims.

3 Architecture-Level Memory Mechanisms

Architecture-level memory mechanisms specify the retained state $\mathcal{M}_t$ in Equation 1: what information a forecaster carries inside its own computation before any explicit external read or post-deployment update interface is introduced. This is the default form of memory in MTS forecasting. The model compresses past observations, cross-variable relations, seasonal patterns, and pretrained experience into parameters, hidden states, attention contexts, patch tokens, graph structures, or adapters. If Retrieval asks how a system reads from a separate store and Evolution asks how memory is updated after deployment, the architectural question is more basic: where is forecasting-relevant information represented?

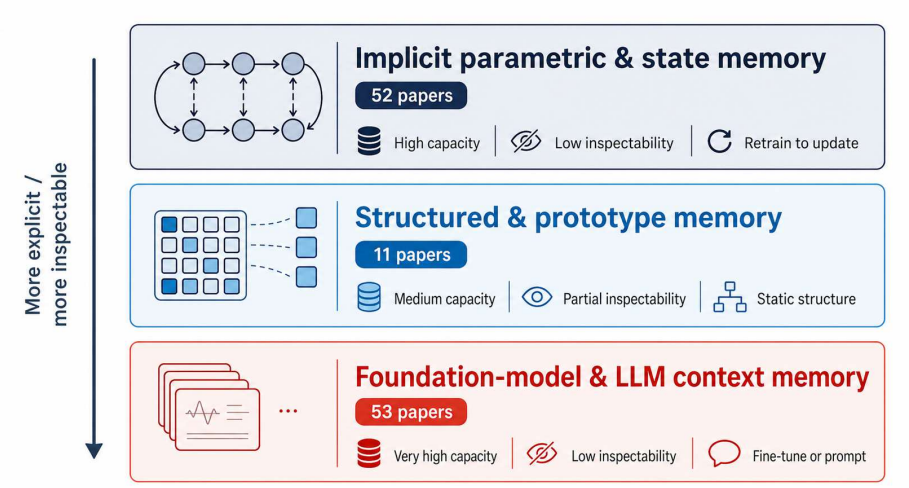


Fig. 3: Architecture-level memory mechanisms in MTS forecasting. Three tiers span implicit parametric memory, structured memory, and foundation-model context memory, trading off capacity, inspectability, and ease of update.

This section reviews architecture-level memory mechanisms by identifying the form in which retained information is represented: temporal state and token memory (Section 3.1), structured relational memory (Section 3.5), and pretrained context memory (Section 3.6). The categories are not mutually exclusive model families; they are ways in which a forecasting architecture instantiates $\mathcal{M}_t$. Figure 3 gives the corresponding hierarchy.

3.1 Implicit Parametric and State Memory

Implicit architectural memory stores information in parameters or latent states that are updated during training and then reused at inference time. A recurrent or state-space forecaster can be viewed as maintaining a compressed state

$$h_t = \Phi_\theta(h_{t-1}, x_t), \qquad \hat{y}_{t+1:t+H} = G_\theta(h_t),$$

where h_t is not an inspectable record of past observations but a learned summary optimized for prediction. This design is attractive because it avoids an external index and keeps inference efficient. Its limitation is that the retained information is entangled with the model computation: once a seasonal pattern, regime, or cross-variable relation has been absorbed into h_t or θ, it is difficult to identify, edit, or delete it without changing the model itself.

3.2 Temporal State Memory

Recurrent, continuous-time, and state-space designs differ in how they compress temporal history. xLSTM-Mixer adapts scalar memory cells from xLSTM to mix temporal and variate information through exponential gating, giving the model a compact recurrent store without quadratic attention cost Kraus et al. (2025). Neural Lad treats the latent state as a continuous-time dynamical system governed by neural ODEs, which is useful for irregular or continuous temporal evolution but still produces an opaque hidden trajectory Li et al. (2023). Selective state-space and hyper-state designs push this idea toward long-horizon MTS forecasting: FACTS propagates learned memory nodes through a factorized state-space graph Li et al. (2025), while TimePro uses variable- and time-aware hyper-states to represent multi-delay effects among variables Ma et al. (2025). WARP goes further by letting the recurrent state take the form of an auxiliary network's weights, enabling gradient-free test-time adaptation but spreading memory across many changing parameters Nzoyem et al. (2026). Across these methods, the common mechanism is compression: history is retained as a state, not as a retrievable case.

3.3 Token and Attention Memory

Transformer and patching architectures represent memory through tokens, attention contexts, and learned projections rather than through a single recurrent state. PatchTST shows how segmenting a time series into patches can preserve local semantics, reduce attention cost, and enable longer lookback windows while storing memory in patch embeddings and shared parameters Nie et al. (2022). Timer-XL extends

the next-token prediction paradigm to multivariate forecasting, using long-context TimeAttention and pretrained parameters to carry temporal priors over variable-length contexts Liu et al. (2025a). TQNet uses periodically shifted learnable queries to capture recurring inter-variable patterns, whereas Triformer uses variable-specific attention and projection matrices to preserve per-variate dynamics Lin et al. (2025); Cirstea et al. (2022). Other designs make temporal context more explicit through learned time embeddings or adaptive local smoothing, as in T-Rep and ARM Fraikin et al. (2024); Lu et al. (2024). These methods may use the language of queries or context, but the memory remains internal: the current input attends to learned or current-window representations rather than to a persistent external store.

3.4 Architectural Trade-Off

Implicit parametric and state memory offers high capacity, mature implementation patterns, and efficient inference. It is therefore a natural baseline for MTS forecasting, especially when the deployment setting emphasizes throughput or when the training distribution is relatively stable. The cost is weak inspectability. A practitioner can often observe that a model benefits from longer contexts, recurrent states, or patch tokens, but cannot easily determine which past regimes are retained or whether a stale regime should be removed. This trade-off motivates the retrieval and evolution mechanisms studied later: they expose read and update operations that architecture-only memory usually hides.

3.5 Structured and Prototype-Based Architectural Memory

MTS forecasting often depends not only on temporal continuation but also on relations among variables. Structured architectural memory makes part of this relational information explicit by storing it in matrices, graphs, frequency operators, or covariance structures. Compared with generic hidden states, these representations are easier to interpret because entries or nodes can often be tied to variables, frequencies, or dependency patterns. Compared with retrieval memory, however, they remain inside the model and are usually not queried, refreshed, or deleted as independent memory records.

Correlation-oriented designs illustrate this intermediate position. CoRA learns cross-variable correlations through a lightweight adapter, decomposing the correlation matrix into time-varying and time-invariant low-rank components and using contrastive learning to distinguish positive and negative variable relations Cheng et al. (2026). The resulting adapter stores relational memory in model parameters: more structured than a generic hidden vector, but not an external memory bank. Probabilistic dependency models provide a related form of memory. Correlated-Error Probabilistic Forecasting uses low-rank-plus-diagonal residual covariance and latent temporal processes to capture how forecast errors co-vary across variables Zheng and Sun (2024). Here, the memory is the learned dependency structure that calibrates predictions through variable-pair relationships.

Graph and frequency architectures store relational memory in a different form. FourierGNN constructs a hypervariate graph in which time-series values are nodes

and Fourier-domain operators capture joint temporal and cross-variable dependencies Yi et al. (2023). This design makes the dependency representation more explicit than ordinary attention because the graph/frequency structure defines how information moves across variables and time. Yet it is still architectural memory: the dependency pattern is learned as part of the model and cannot be independently retrieved like a historical exemplar.

The boundary is therefore functional rather than visual. A matrix, graph, adapter, or prototype is not retrieval memory simply because it looks like a memory structure. It becomes retrieval memory only when a forecasting query reads from a persistent store whose contents can exist apart from the model parameters. Structured architectural memory is best understood as a bridge: it exposes some semantics of what is remembered while retaining the efficiency and opacity of end-to-end architectures.

3.6 Foundation-Model and LLM-Based Context Memory

Pretrained models introduce a third architectural form: memory inherited from large-scale pretraining. A time-series foundation model or LLM-based forecaster does not necessarily store the target system's historical cases in an external database. Instead, it carries broad temporal or linguistic priors in its parameters and adapts those priors to a target MTS task through patching, prompts, adapters, alignment modules, or distillation. This changes the memory question from "what did this model learn from one dataset?" to "which pretrained priors are useful for this target forecasting context, and how are they aligned with multivariate numerical data?"

Time-series foundation models provide the most direct version of pretrained architectural memory. Moirai is pretrained on large-scale heterogeneous time-series data and uses patching and channel-independent processing to support zero-shot forecasting across different frequencies and variate counts Woo et al. (2024). DAM learns continuous-time basis functions from randomly sampled histories, allowing pretrained temporal components to be composed for unseen datasets Darlow et al. (2024). In both cases, the memory is broad and reusable, but it is static after pretraining unless additional adaptation or update mechanisms are introduced.

LLM-based forecasters import a different kind of memory. TimeCMA combines a time-series branch with an LLM-empowered branch and aligns the two modalities through similarity matching Liu et al. (2025b). CALF addresses the distribution mismatch between temporal tokens and textual tokens through cross-modal matching, feature regularization, and output consistency Liu et al. (2025c). AutoTimes projects time series into the language-token embedding space and uses the frozen LLM's autoregressive dynamics as an architectural prior for forecasting Liu et al. (2024). LLM Reprogramming similarly uses prompting, grouped-query attention, and LoRA-style adaptation to combine domain descriptions with numerical sequences Srinivas et al. (2024). These methods can enrich forecasting with pretrained knowledge or textual context, but the memory remains parameterized unless an explicit retrieval store is added.

Distillation and adapter-based methods compress or redirect pretrained memory for MTS use. TimeKD transfers privileged future-aware representations from a calibrated language-model teacher into a lighter student through correlation and feature

distillation Liu et al. (2025d). AdaPTS adapts univariate time-series foundation models to probabilistic multivariate forecasting through learned feature-space adapters and uncertainty modeling Benechehab et al. (2025). These approaches highlight a practical advantage of pretrained architectural memory: it can be reused without training a full model from scratch. They also reveal the main risk. Text-based LLMs, univariate TSFMs, and heterogeneous pretrained forecasters may encode priors that do not align with the target system's sampling frequency, variable semantics, non-stationarity, or cross-variable structure.

3.7 Comparative Summary

The architecture-level pillar answers the storage question for memory-mechanism analysis. Temporal state and token memories compress history into hidden states, patch embeddings, attention contexts, or learned projections. They are efficient and widely usable, but difficult to inspect or edit. Structured relational memories encode cross-variable dependencies in matrices, graphs, frequency operators, or covariance structures. They improve semantic visibility, but usually remain tied to model parameters. Pretrained context memories carry broad priors from TSFMs or LLMs, enabling transfer and low-data adaptation while introducing alignment and inference-cost concerns.

These differences suggest a simple reading guide for the rest of the survey. If a method only stores information in parameters, states, tokens, adapters, or structured operators, it belongs primarily to the architectural pillar. If it reads from a persistent external store, it enters the retrieval pillar. If it specifies how stored information is written, refreshed, consolidated, or forgotten after deployment, it enters the evolution pillar. Architectural memory is therefore not a weaker form of memory; it is the substrate on which explicit retrieval and evolution mechanisms can be built.

4 Retrieval Memory Mechanisms

Retrieval memory mechanisms expose the access function ρ_ϕ in Equation 1. A retrieval-augmented MTS forecaster does not only compress the recent window into hidden states or attention contexts; it constructs a query, reads from a persistent store, and fuses selected evidence into the forecast. A generic retrieval interface can be written as:

$$\hat{y}_{t+1:t+H} = f_\theta\big(x_{1:t},\ \mathcal{R}\big(q(x_{1:t}), \mathcal{M}\big)\big), \tag{2}$$

where $q : \mathbb{R}^{T\times C} \to \mathbb{R}^d$ maps the observed window to a retrieval key, $\mathcal{M} = \{(e_i, s_i)\}_{i=1}^N$ is a memory store with embeddings e_i and associated evidence s_i, $\mathcal{R}(q, \mathcal{M})$ selects entries according to a matching rule, and f_θ fuses the retrieved evidence with the current context.

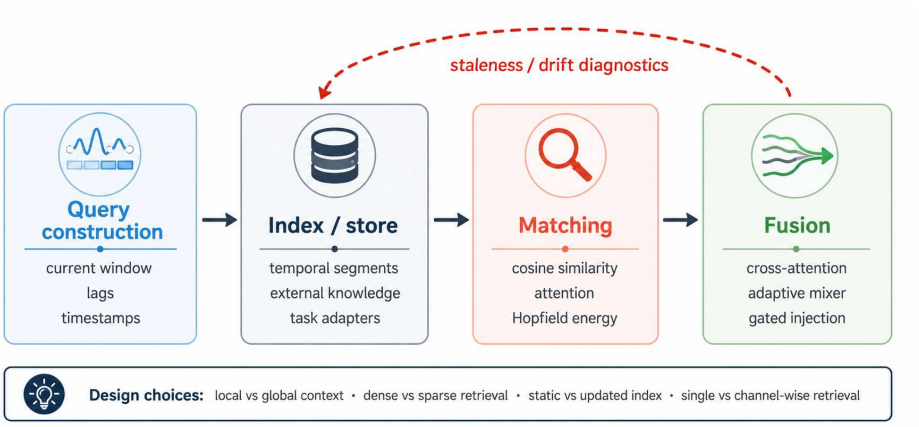


Fig. 4: The retrieval-augmented MTS forecasting pipeline. Four components—query construction, index/store, matching, and fusion—define the retrieval interface. Staleness and drift diagnostics close the feedback loop.

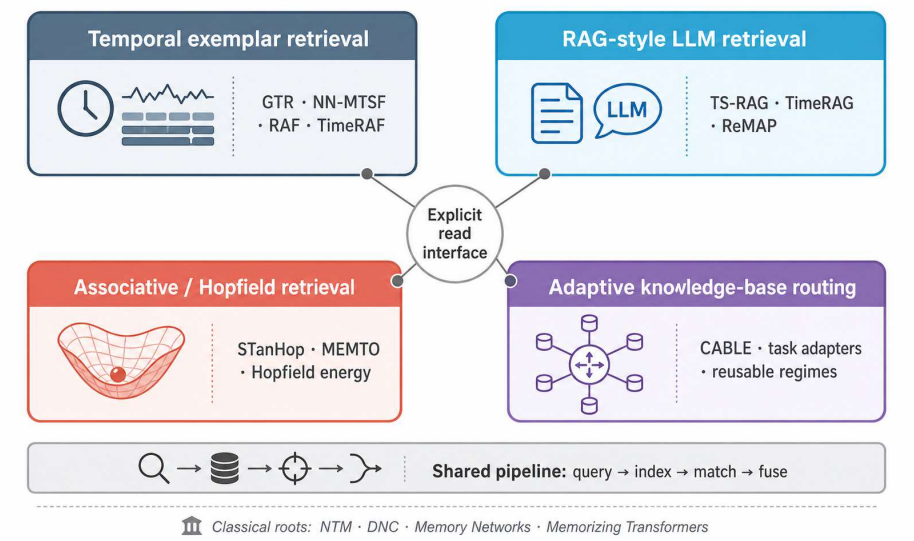


Fig. 5: Taxonomy of retrieval memory mechanisms for MTS forecasting. Four patterns span temporal exemplar retrieval, RAG-style LLM retrieval, associative (Hopfield) retrieval, and adaptive knowledge-base routing, all building on classical differentiable memory foundations.

This section reviews retrieval by the interface choices in Equation 2: what is stored, how the query is represented, how similarity is computed, how retrieved evidence is fused, and how the store is kept reliable under drift. Figure 4 makes these interface choices concrete, while Figure 5 gives a reader map for the retrieval mechanisms discussed below. This organization makes the distinction from architecture-level context

explicit. A long attention window, prompt, or learned query vector is not retrieval unless it reads from a persistent store whose contents can exist apart from the current forward pass.

4.1 Retrieval Interface Foundations

Classical differentiable memory systems provide the vocabulary for the retrieval interface. Neural Turing Machines introduced content-addressed read and write heads over an external memory matrix, while Differentiable Neural Computers added temporal linkage so that a controller could follow the order in which memory entries were written Graves et al. (2014, 2016). End-To-End Memory Networks introduced iterative memory hops, where each read refines the next query Sukhbaatar et al. (2015). Memorizing Transformers showed how nearest-neighbor lookup can extend a sequence model beyond its fixed context window by retrieving cached key-value pairs Wu et al. (2022). Modern Hopfield Networks further connect attention to associative retrieval, where stored patterns are recovered through energy-based dynamics Ramsauer et al. (2021).

For MTS forecasting, these foundations are useful less as architectures to copy than as design axes. The retrieved item may be a raw temporal segment, a learned representation, text or metadata, a model adapter, or an associative prototype. The matching rule may rely on Euclidean distance, cosine similarity, learned cross-attention, spectral similarity, or energy minimization. The fusion step may average retrieved cases, gate them into a foundation model, condition a diffusion process, or route to an adapter. The following subsections use these axes to organize recent MTS retrieval mechanisms.

4.2 Temporal Exemplar Stores

The most direct retrieval store for MTS forecasting is a collection of historical time-series segments. In this setting, the current window is encoded as a query, similar past windows are retrieved, and their subsequent trajectories or representations provide evidence for the forecast. The basic case-based operation selects k neighbors from the store:

$$\mathcal{R}_k(q, \mathcal{M}) = \left\{ (s_i, w_i) \;\middle|\; e_i \in \mathcal{M},\; w_i = \frac{\exp(\mathrm{sim}(q, e_i)/\tau)}{\sum_{j=1}^{k} \exp(\mathrm{sim}(q, e_j)/\tau)} \right\}_{i=1}^{k}, \tag{3}$$

with fusion via $\hat{y}_{t+1:t+H} = f_\theta(x_{1:t}, \sum_{i=1}^{k} w_i \cdot s_i)$. This formulation makes retrieval quality visible: embedding design, normalization, temporal alignment, distance metric, and temperature τ can all change which histories are treated as relevant evidence.

Recent exemplar-retrieval methods instantiate this interface at different levels of complexity. Global Temporal Retrieval builds a full-history embedding store and uses cross-attention to align retrieved distant regimes with the local input Cao et al. (2026). Nearest-neighbor MTS forecasting exposes the same principle in a simpler case-based form, using historical subsequences directly as memory for prediction Zhang et al. (2024b). Retrieval-augmented time-series forecasting and TimeRAF combine raw segments with foundation-model representations, so the store contains both concrete historical cases and learned temporal features Tire et al. (2024); Zhang et al. (2024a). Retrieval-augmented diffusion uses retrieved segments as conditional priors in a probabilistic denoising process, showing that retrieval can support uncertainty modeling

rather than only point forecasts Liu et al. (2024). Cross-RAG and CRAFT highlight two further design choices: whether retrieval can work in zero-shot settings with a frozen index, and whether each channel should maintain its own retrieval store when variables have different periodicities or spectra Lee et al. (2026); Kang et al. (2026).

The shared advantage of temporal exemplar retrieval is traceability: a forecast can be linked to particular past segments. The shared risk is that similarity does not guarantee relevance. Under concept drift, a nearest neighbor in embedding space may belong to an obsolete regime; under channel heterogeneity, a single shared index may retrieve evidence that is useful for one variable but misleading for another. Exemplar retrieval therefore raises memory-system questions that ordinary architecture comparisons do not: how large should the store be, how often should it be refreshed, and how should stale or low-utility entries be pruned?

4.3 RAG-Style and Heterogeneous Retrieval

A second retrieval pattern widens the store beyond raw temporal segments. RAG-style forecasting systems may retrieve text, metadata, event descriptions, financial documents, task descriptions, adapters, or parameter mappings. The motivation is that some forecast-relevant information is not present in the recent numerical window: holiday calendars, market news, domain events, maintenance logs, and task-level context can all change the meaning of a temporal pattern.

TS-RAG represents this direction by maintaining a time-series knowledge base with both temporal segments and text or metadata annotations, then using an adaptive retrieval mixer to weigh retrieved evidence for foundation-model forecasting Ning et al. (2025). TimeRAG adapts this idea to LLM-based forecasting by retrieving both historical sequences and textual context from a dual-modality store and injecting them through a structured prompt Yang et al. (2024). Financial RAG-LLM forecasting illustrates a domain-specific version in which retrieved financial documents can provide predictive context beyond price history Xiao et al. (2025). ReMAP expands the notion of retrievable evidence further: instead of retrieving data records, it retrieves prior model adaptations and maps them to the current forecasting context Dey and Nair (2024).

The main design problem in this family is alignment. Continuous multichannel queries must be embedded into a space compatible with text, metadata, documents, or parameter states, and the fusion step must decide how much these heterogeneous items should influence a numerical forecast. Temporal validity also becomes stricter: a retrieved news item or event description must be available before the prediction time, and stale documents from a different regime may need to be downweighted. RAG-style retrieval therefore provides richer context but also increases the burden on query construction, source freshness, and evidence grounding.

4.4 Associative Retrieval and Knowledge Routing

Associative retrieval provides a more structured alternative to ordinary nearest-neighbor search. STanHop uses sparse tandem Hopfield layers to retrieve temporal patterns from a stored memory set and then refine them through sparsification,

addressing the capacity-precision trade-off that arises when many stored patterns compete for attention Wu et al. (2024). MEMTO, although developed for multivariate time-series anomaly detection, is useful for this taxonomy because its memory slots store prototype representations that are read by similarity and updated through learned gates Song et al. (2023). These designs show how retrieval and evolution can meet inside one module: reading selects relevant memories, while writing updates or sharpens the stored prototypes.

4.5 Adaptive knowledge-base routing

Retrieval can also select model components rather than data examples. CABLE maintains an evolving knowledge base and routes new examples toward relevant stored adapters or representations through gradient-based similarity Julian et al. (2025). Although CABLE is not a core MTS forecaster, the routing pattern is relevant: a forecasting system could store regime-specific adapters, prototypes, or task memories, then select the component most compatible with the current regime. This view bridges the retrieval pillar with memory evolution because the same store must be expanded, consolidated, or pruned as new regimes appear.

Associative retrieval and routing shift attention from "which past segment is closest?" to "which stored pattern, prototype, or adapter should govern the current forecast?" This can improve structure and interpretability, but it also introduces capacity and maintenance questions. Hopfield-style retrieval depends on the separation of stored patterns, while routing-based systems depend on the quality and freshness of the stored components.

4.6 Boundary Rules

Several influential MTS methods remain non-retrieval baselines even though they use terms such as query, attention, context, or knowledge. PatchTST, Timer-XL, TQNet, TimeCMA, TimeKD, and AutoTimes model context internally and do not query an external store Nie et al. (2022); Liu et al. (2025b,d,a); Lin et al. (2025); Liu et al. (2024). Their attention mechanisms, prompt contexts, and cross-modal alignment modules operate within learned parameters and the current input. Formally, a forecasting method is classified as Retrieval iff it satisfies three conditions:

$$\begin{aligned}
&\text{(i)} \quad \mathcal{M} \cap \Theta = \emptyset, \\
&\qquad \text{the memory store is external to model parameters } \Theta, \\
&\text{(ii)} \quad \frac{\partial \hat{y}}{\partial \mathcal{R}} \not\equiv \mathbf{0}, \\
&\qquad \text{the forecast functionally depends on retrieved content,} \\
&\text{(iii)} \quad \mathcal{M} \text{ persists across } \geq 2 \text{ inference steps}, \\
&\qquad \text{the store is not constructed from current input alone.}
\end{aligned} \tag{4}$$

Condition (i) excludes memories that are simply learned weights, adapters, or prompts. Condition (ii) excludes methods that compute internal query scores without an external read. Condition (iii) excludes temporary structures built only from the

current input, such as ordinary self-attention over $x_{1:t}$. Keeping this boundary explicit makes index construction, retrieval latency, stale-memory risk, retrieval faithfulness, and fusion robustness visible as engineering problems specific to external memory.

4.7 Summary

Retrieval memory mechanisms in MTS forecasting can be read as a set of interface choices. Temporal exemplar retrieval stores historical segments and makes forecasts traceable, but depends strongly on embedding quality, channel granularity, and index freshness. RAG-style retrieval widens the store to text, metadata, documents, or parameter mappings, enabling context-aware forecasting while introducing a temporal modality gap. Associative retrieval and knowledge-base routing use stored prototypes, Hopfield patterns, or adapters to select reusable structure rather than only similar cases. Across these designs, the retrieval pillar is distinct from architecture-level memory because it exposes an external read operation. Once that read operation exists, the central questions become how the store is built, how relevance is measured, how evidence is fused, and how the memory remains reliable as the time series evolves.

Summary. Retrieval-augmented MTS forecasting should be identified by its interface, not by terminology. A method enters the Retrieval pillar when a persistent store is queried and the forecast depends on the retrieved content. The main design axes are store content, query representation, matching rule, fusion mechanism, and memory freshness. These axes separate external memory systems from ordinary attention, prompts, and context windows.

5 Evolution Memory Mechanisms

Evolution memory mechanisms expose the update function U_ψ in Equation 1. They govern how a deployed forecasting system writes, updates, consolidates, and forgets retained information after the training stage. Architecture-level mechanisms determine what a model can represent, and retrieval mechanisms determine how stored information is read. Evolution asks whether the retained information remains useful as the data-generating process changes. In MTS forecasting this question is central: regimes recur, channels drift at different rates, sensor relations change, and old seasonal patterns may be either valuable historical evidence or harmful stale memory.

Memory evolution can be expressed as a state transition on the memory store $\mathcal{M}$:

$$\mathcal{M}_{t+1} = \Phi\big(\mathcal{M}_t,\ x_t,\ y_t,\ \mathcal{L}(\hat{y}_t, y_t)\big), \tag{5}$$

which can be decomposed into three recurring operations:

$$\mathcal{M}_{t+1} = \mathcal{M}_t\ \cup\ \mathcal{W}(x_t, y_t)\ \setminus\ \mathcal{F}(\mathcal{M}_t)\ \cup\ \mathcal{C}(\mathcal{M}_t, x_{t-k:t}), \tag{6}$$

where $\mathcal{W}$ writes new information, $\mathcal{F}$ removes stale, poisoned, or low-utility entries, and $\mathcal{C}$ consolidates repeated patterns into compact representations. Each operation at time t is constrained by causality: it should depend only on observations and feedback available up to that time. We therefore treat a method as an evolution

memory mechanism when it specifies a write, refresh, consolidate, adapt, or forget operation beyond ordinary training-time gradient updates.

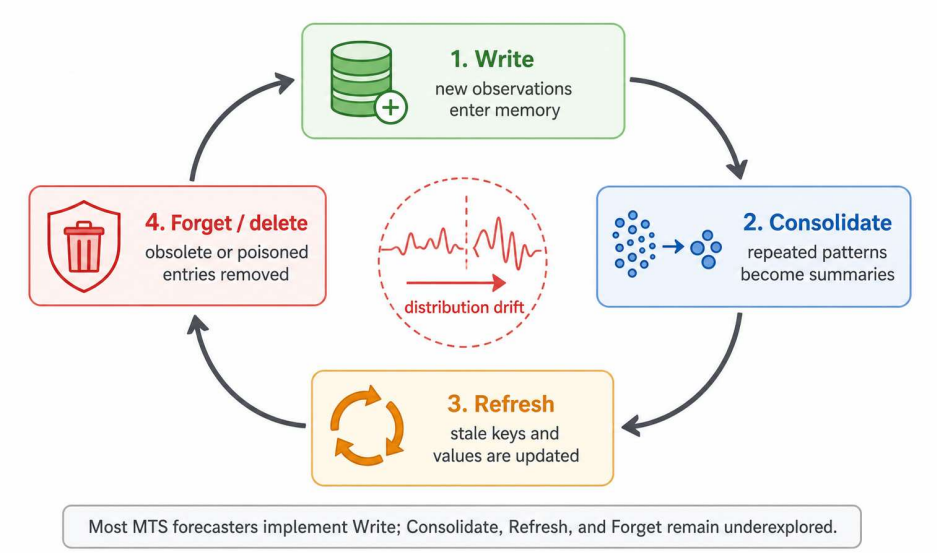


Fig. 6: The memory evolution lifecycle under distribution drift. Many current MTS forecasters emphasize the Write phase during training; Consolidation, Refresh, and Forget remain comparatively underexplored.

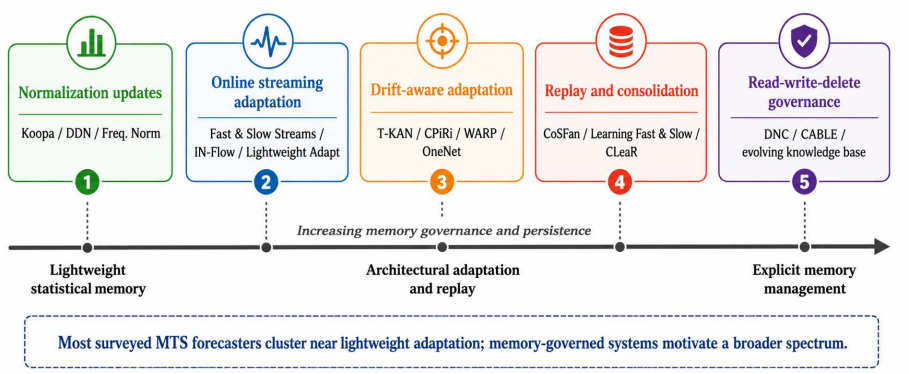


Fig. 7: Spectrum of evolution memory mechanisms, from lightweight normalization-based adaptation (left) to explicit read-write-delete memory management (right). The surveyed MTS forecasters are concentrated near lightweight adaptation, while more explicit memory mechanisms motivate designs spanning a broader spectrum.

The rest of this section follows the lifecycle implied by Equation 6. We first use classical differentiable memory to define the read-write-delete interface. We then review three design choices that recur in MTS forecasting: what enters memory, what part of

the system is modified, and how old knowledge is retained or removed. Figure 6 and Figure 7 visualize this lifecycle and the corresponding mechanism spectrum.

5.1 Classical Foundations: The Read-Write-Delete Paradigm

The Differentiable Neural Computer (DNC) provides a reference point for explicit memory evolution because it separates a neural controller from an external memory matrix with differentiable read, write, and free-allocation operations Graves et al. (2016). This design is useful here not as a direct forecasting architecture, but as an interface model: a system can retrieve relevant entries, write newly verified observations, and release locations that no longer carry useful information. For MTS forecasting, the same interface suggests a natural deployment loop. A forecaster reads comparable regimes, writes ground-truth feedback after it arrives, refreshes representations when repeated patterns accumulate, and deletes or quarantines stale intervals without forcing every update into the base model weights.

Current MTS work rarely implements this full interface end to end. Instead, different methods instantiate partial evolution operations. Replay buffers write and reuse selected experiences; normalization modules refresh statistics; online adapters modify a small parameter subset; drift-aware models change routing or operators; and foundation-model fine-tuning raises the problem of retaining pretrained temporal knowledge. The value of the read-write-delete view is that it makes these partial mechanisms comparable.

5.2 Writing and Consolidating Experience

The most direct evolution operation is writing new experience into a persistent reference set. Replay-based continual learning methods use this idea to balance plasticity with stability. CoSFan maintains task-aware replay buffers with slow and fast adaptation pathways, so transient changes can be tracked without immediately overwriting more stable knowledge Missel and Wang (2025). Learning Fast and Slow brings a similar two-timescale idea to online time-series forecasting, alternating rapid updates with slower consolidation to reduce catastrophic interference Pham et al. (2022). CLeaR approaches the same stability problem through adaptive continual regularization, using learned per-parameter retention strengths rather than an explicit replay-only mechanism He and Sick (2021). Proactive adaptation adds a trigger question: instead of waiting for error to increase after drift, it attempts to detect precursor signals and update before degradation becomes severe Zhao and Shen (2025).

These methods differ in implementation, but they share the same memory trade-off. The current distribution must be learned quickly enough to remain useful, while reference knowledge must be retained strongly enough to avoid overfitting to short-lived noise. This stability-plasticity dilemma can be written as:

$$\min_{\theta,\mathcal{M}} \underbrace{\mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{cur}}}[\mathcal{L}(f_\theta(x;\mathcal{M}),y)]}_{\text{plasticity}} + \lambda \cdot \underbrace{\mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{ref}}}[\mathcal{L}(f_\theta(x;\mathcal{M}),y)]}_{\text{stability}}, \tag{7}$$

where $\mathcal{D}_{\text{cur}}$ is the current data distribution, $\mathcal{D}_{\text{ref}}$ represents consolidated knowledge such as a replay buffer, frozen backbone, or regularization reference, and λ controls the adaptation-retention trade-off. In this view, replay buffers, slow pathways, and elastic regularization are different ways of defining $\mathcal{D}_{\text{ref}}$ and choosing how strongly it should constrain future updates.

5.3 Refreshing Memory Under Drift

Evolution does not always require an explicit buffer. A second family of mechanisms refreshes internal states, operators, statistics, or routing choices when the input distribution changes. T-KAN and MT-KAN model drift through learnable functional components based on Kolmogorov-Arnold network layers, while CPiRi focuses on robustness to channel permutation and changing cross-channel relations Xu et al. (2024, 2026). WARP makes the recurrent state itself adaptive by representing it in weight space, supporting gradient-free test-time adjustment Nzoyem et al. (2026). At the distribution level, drift can be handled by regularizing latent drift processes or dynamically weighting predictors, as in concept-drift generalization work and OneNet's online ensembling Zhao et al. (2026); Zhang et al. (2023). These methods treat evolution as modification of the forecasting machinery rather than insertion of new records into an external store.

Lightweight statistical refresh is especially common because it can be deployed without retraining the full model. Koopman-based decomposition separates time-invariant and time-varying dynamics, while dynamic normalization methods update statistics in the time domain, frequency domain, or both Liu et al. (2023); Dai et al. (2024); Liu et al. (2024, 2023). From the memory perspective, normalization statistics are compact state variables: they are read during the forward pass, updated as new observations arrive, and reset or replaced when they become stale. This makes them a practical but limited form of memory evolution. They are efficient and often easy to insert into existing forecasters, but they usually do not explain which historical regimes are retained or forgotten.

Generative and multiscale methods broaden the refresh operation by localizing the part of the representation that should change. Non-stationary diffusion and dynamic decomposition with diffusion reconstruction adapt the generative process to shifted or unstable components Ye et al. (2025); Wang et al. (2023). TimeStacker exposes non-stationary structure at multiple temporal resolutions, and patch-specific experts route local segments to specialized predictors when shift is concentrated in particular patches Liu et al. (2025e); Sun et al. (2025). These approaches suggest that evolution can be spatially and temporally selective: a system may refresh only a channel subset, frequency band, latent component, patch, or expert rather than treating the entire model as one undifferentiated memory.

5.4 Online and Streaming Adaptation

Streaming settings add two constraints to memory evolution. First, updates must respect temporal causality: statistics, adapters, or replay entries used at time t should not depend on future observations. Second, the update cost must be small enough for

the deployment rate. Fast and Slow Streams makes this causality issue explicit by separating rapid online adaptation from slower consolidation and by highlighting leakage risks in online forecasting evaluation Lau et al. (2025). Lightweight online adaptation for time-series foundation models updates small adapter modules while keeping the backbone frozen, reducing the risk and cost of modifying all pretrained parameters Lee et al. (2025). IN-Flow combines instance normalization with normalizing flows, using simple statistical alignment for first-order shifts and invertible transformations for richer distributional changes Fan et al. (2025).

These streaming methods can be read through the trigger axis of memory evolution. Some update continuously, some update per batch or per step, and others update only when drift indicators or precursor signals appear. Faster triggers reduce the time spent in a mismatched regime, but they increase exposure to transient noise. Slower triggers are more stable but can leave the model stale after abrupt shifts. For MTS, the problem is more complicated because each channel may drift at a different rate and because cross-variable relations can change even when marginal distributions look stable.

5.5 Forgetting and Foundation-Model Retention

Forgetting is not only a failure mode; in an explicit memory mechanism it can also be a deliberate operation. Obsolete regimes, corrupted intervals, low-utility entries, and privacy-sensitive records may need to be removed or downweighted. The difficulty is deciding what should be forgotten without erasing reusable temporal structure. Time-series foundation models make this tension visible. Focused studies of catastrophic forgetting report that fine-tuning on new temporal domains can degrade previously learned capabilities, with periodic and trend knowledge affected differently from other signal properties Karaouli et al. (2025). Rehearsal strategies can mitigate this degradation by interleaving representative old examples during adaptation, but they often assume access to suitable reference data.

This retention problem links foundation-model adaptation back to replay and consolidation. A pretrained forecaster carries broad temporal priors in its parameters; domain adaptation writes new information into that parametric memory; retention mechanisms decide how strongly previous priors should constrain the update. Data-free retention, adapter-based updates, and memory stores of representative regimes are therefore natural directions for MTS foundation models, especially when pretraining data are unavailable or proprietary.

5.6 Synthesis: Toward Dynamic Memory Management

The surveyed mechanisms can be compared along three practical axes. The first is the *adaptation trigger*: continuous updates, per-step streaming updates, per-batch statistic updates, precursor-based updates, or fine-tuning events. The second is the *modification scope*: external memory entries, normalization statistics, adapters, operators, ensemble weights, experts, or full model parameters. The third is the *retention mechanism*: replay buffers, frozen backbones, regularization anchors, time-invariant subspaces, free-allocation, or similarity-based routing. Table 3 summarizes these axes

at the design-family level, so that the comparison emphasizes recurring mechanisms rather than a method inventory.

Several methods point toward a more integrated memory manager. Conditionally whitened generative models and PhysDiff show how generative modeling can incorporate distribution-shift handling through latent whitening or physically guided reconstruction Yang et al. (2026); Li et al. (2025). CABLE illustrates how retrieval and evolution can meet in an adaptive knowledge base, where stored representations or adapters are selected for new cases and revised as tasks evolve Julian et al. (2025). A future MTS foundation model could combine these ideas into an explicit interface: read comparable regimes, write verified observations, refresh embeddings or statistics, consolidate repeated regimes, and delete stale or poisoned entries. The core design question is how these components should coordinate rather than merely coexist.

Table 3: Synthesis of memory-evolution design families. The table groups representative methods by the trigger, modified state, and retention principle that define their memory behavior.

Evolution family	Typical trigger	What changes	Retention principle and examples
Replay and consolidation	Task, stream, or drift update	Replay buffers, task memories, slow–fast states, or regularization weights.	Rehearse or anchor useful old regimes; examples include CoSFan, Learning Fast and Slow, CLeaR, and proactive adaptation.
Statistic refresh	Per-step, per-batch, or inference-time update	Normalization statistics, frequency-domain summaries, or lightweight distribution states.	Track distribution shift at low cost; examples include adaptive normalization, DDN, frequency-adaptive normalization, and IN-Flow.
State/operator adaptation	Continuous or drift-aware update	Recurrent states, Koopman operators, KAN components, or weight-space states.	Adapt the forecasting machinery without replacing the whole memory; examples include Koopa, T-KAN/MT-KAN, CPiRi, and WARP.
Routing and ensemble update	Per-step, per-patch, or similarity trigger	Expert weights, adapter selection, patch experts, or knowledge-base routing.	Isolate regimes and route queries to compatible memories; examples include OneNet, Patch Experts, and CABLE.
Generative refresh	Shift-aware generation or reconstruction	Diffusion components, decomposed latent factors, or multiscale representations.	Refresh unstable components selectively; examples include non-stationary diffusion, Drift Doesn't Matter, TimeStacker, and PhysDiff.
Foundation-model retention	Fine-tuning or online adapter update	Adapters, partial parameters, replay examples, or frozen backbones.	Preserve pretrained temporal priors while adapting domains; examples include lightweight online adaptation and studies of time-series foundation-model forgetting.

Evaluation has not yet caught up with this design space. Standard forecasting metrics such as MAE, MSE, and probabilistic calibration do not reveal whether a memory update was useful, whether a stale entry caused negative transfer, whether adaptation interfered with unrelated regimes, or whether a memory store can be

poisoned. Memory consolidation, the $\mathcal{C}$ operation in Equation 6, can be framed as a rate-distortion trade-off:

$$\mathcal{C}^* = \arg\min_{\mathcal{C}:\mathcal{M}\to\mathcal{M}'} \underbrace{\mathbb{E}_{x\sim\mathcal{D}}\big[\mathcal{L}(f_\theta(x;\mathcal{M}'),y) - \mathcal{L}(f_\theta(x;\mathcal{M}),y)\big]}_{\text{information loss}} + \nu \cdot \underbrace{|\mathcal{M}'|}_{\text{compactness}}, \tag{8}$$

where $\mathcal{C}$ compresses $\mathcal{M}$ into a smaller store $\mathcal{M}'$, $|\mathcal{M}'|$ measures memory footprint, and ν controls the pressure toward compactness. This objective makes explicit that refreshing memory is not only an accuracy problem; it also involves latency, storage cost, and the risk of removing rare but important regimes. The evaluation protocol in Table 4 follows from this view by testing no-memory ablations, stale memory, corrupted memory, delayed updates, memory-size scaling, recurring regimes, and channel-structure shifts.

Table 4: Memory-specific stress tests for memory-augmented MTS forecasting. The protocol focuses on failure modes that standard accuracy-only benchmarks often leave implicit.

Stress setting	Perturbation setup	Primary readouts
No-memory ablation	Disable retrieval, replay, external context, or memory writes while keeping the backbone fixed.	Error delta; calibration delta; per-horizon degradation.
Stale-memory test	Freeze the index before a drift point or query only pre-drift memory.	Stale-hit rate; forecast degradation; recovery time after refresh.
Corrupted-memory test	Inject noisy, shifted, mislabeled, or adversarial intervals into the memory store.	Poisoned-hit rate; error delta; quarantine or deletion precision.
Delayed-update test	Allow observations to enter memory only after fixed delays.	Recovery lag; update cost; stability–plasticity trade-off.
Memory-size scaling	Vary memory or index size while holding the base model constant.	Accuracy–latency curve; footprint; retrieval diversity.
Regime recurrence	Reintroduce seasonal, event, or operating regimes across train–validation–test splits.	Reuse quality; adaptation speed; forgetting rate.
Channel-structure shift	Permute, drop, mask, or reweight channel subsets.	Cross-channel error; relation recovery; missing-channel robustness.

6 Challenges and Research Roadmap

The preceding sections separate memory mechanisms into representation, access, and update functions. This section uses the same structure to connect open challenges with

future research directions. We use *challenge* to denote a limitation or failure mode exposed by current memory mechanisms, and *roadmap* to denote the corresponding design response. The relationship is therefore diagnostic rather than sequential: each challenge identifies what current systems do not yet handle well, while the roadmap points to mechanism-level interfaces that could make the limitation measurable, controllable, or easier to audit.

A practical future MTS forecaster may expose four memory-interface operations:

$$\begin{aligned} \textbf{Read:} \quad & \mathcal{R}(q(\mathbf{X}_{t-L+1:t}), \mathcal{M}_t) \to \{(s_i, w_i)\}_{i=1}^{k}, \\ \textbf{Write:} \quad & \mathcal{W}(\mathbf{X}_{t-L+1:t}, \mathbf{e}_t) \to \mathcal{M}_t \cup \{\text{new entry}\}, \\ \textbf{Refresh:} \quad & \mathcal{C}(\mathcal{M}_t) \to \mathcal{M}'_t \quad \text{s.t. } |\mathcal{M}'_t| \leq |\mathcal{M}_t|, \\ \textbf{Delete:} \quad & \mathcal{F}(\mathcal{M}_t) \to \mathcal{M}_t \setminus \{m \in \mathcal{M}_t \mid \text{util}(m) < \delta\}. \end{aligned} \tag{9}$$

The read interface retrieves relevant regimes, external events, domain descriptions, adapters, or prototypes. The write interface decides what newly verified experience should enter memory. The refresh interface consolidates repeated patterns, updates embeddings, and re-indexes stale entries. The delete interface removes obsolete, poisoned, low-confidence, or privacy-sensitive memory records. The six challenges below can be read as requirements for making these operations useful in MTS forecasting rather than merely naming them.

Table 5: Challenge-to-roadmap map for memory mechanisms in MTS forecasting. Challenges describe current limitations; roadmap directions describe mechanism-level responses.

Challenge	Why it matters for memory mechanisms	Roadmap direction
Latency–utility trade-off	Retrieval, LLM reasoning, and large foundation models may add cost without proportional forecast gain.	Report retrieval/reasoning latency, memory-size scaling, and no-memory ablations; design efficient read interfaces.
Temporal modality gap	Continuous multichannel queries do not match text-style RAG assumptions about symbolic query-document similarity.	Learn phase-, scale-, channel-, and regime-aware embeddings; couple retrieval with in-context demonstrations.
Memory poisoning and stale evidence	Stores can contain obsolete, corrupted, or adversarial records that are selected by similarity but harmful for forecasting.	Add validation, freshness scoring, quarantine, deletion, and robustness tests for retrieved or replayed memory.
Evaluation mismatch	Standard MAE/MSE or benchmark rank rarely isolates read, write, refresh, or forget behavior.	Benchmark retrieval precision, stale-memory degradation, drift recovery, update latency, and forgetting safety.
Hallucination in TSFMs	Foundation-model forecasts or explanations may be fluent but temporally unsupported under shift or rare regimes.	Check consistency between generated forecasts, retrieved evidence, uncertainty estimates, and temporal dynamics.
Theory of what to store	Current methods store parameters, states, cases, prompts, adapters, or buffers without shared storage principles.	Connect non-stationary multivariate process structure to memory capacity, consolidation, deletion, and interpretability.

6.1 Latency-Utility Trade-Off

Retrieval and LLM-based reasoning can expand the effective context of a forecaster, but embedding, search, ranking, prompt construction, and fusion may be too expensive for online control or high-frequency forecasting. This risk is visible in RAG-style time-series systems and LLM-empowered forecasters Ning et al. (2025); Liu et al. (2025b); Zhang et al. (2024a); Tire et al. (2024); Xiao et al. (2025); Srinivas et al. (2024); Liu et al. (2024). Recent ablations on LLM-based forecasting also show that removing or replacing the LLM component can reduce inference cost while preserving comparable forecasting accuracy Tan et al. (2024). Retrieval-augmented forecasting is easier to justify over a no-retrieval baseline when:

$$\mathbb{E}[\mathcal{L}(f_\theta(x; \mathcal{R}(q(x), \mathcal{M})), y)] + \gamma \cdot \mathrm{Lat}(\mathcal{R}) \; < \; \mathbb{E}[\mathcal{L}(f_\theta(x; \emptyset), y)], \tag{10}$$

where $\mathcal{L}(f_\theta(x; \emptyset), y)$ is the loss of the context-only forecaster, $\mathrm{Lat}(\mathcal{R})$ is end-to-end retrieval latency, and γ converts latency to task utility cost. The roadmap response is straightforward: memory-mechanism papers should report latency, memory footprint, and no-memory ablations together with forecast error, especially when the proposed benefit comes from retrieval or large-model reasoning.

6.2 Temporal Modality Gap

Text RAG often assumes that queries and documents share a symbolic modality. MTS retrieval is different: the query is high-dimensional, continuous, multichannel, phase-sensitive, and often non-stationary. A retrieval index that works for one dataset may fail under normalization changes, channel permutations, seasonal shifts, or regime changes. GTR, nearest-neighbor forecasting, TS-RAG, TimeRAF, and retrieval-augmented LLM forecasting show the value of explicit retrieval while also making embedding design, channel alignment, and stale-index handling recurring research questions Cao et al. (2026); Ning et al. (2025); Zhang et al. (2024a,b); Xiao et al. (2025).

The roadmap direction is regime-aware and channel-aware retrieval. Rather than treating temporal similarity as a single distance, future systems could learn embeddings that encode phase, scale, seasonality, variable subsets, and lagged relations. In-context learning for time-series foundation models strengthens this direction: retrieved examples can become demonstrations, and the model's ICL capability can determine how effectively retrieved content guides the forecast Faw et al. (2025); Yao et al. (2025). This couples the read interface with the forecasting model rather than treating retrieval as a detachable pre-processing step.

6.3 Memory Poisoning and Stale Evidence

Memory mechanisms can fail when retained evidence is obsolete, corrupted, or adversarial. BackTime Lin et al. (2024) studies backdoor attacks against MTS forecasting, where an attacker injects trigger patterns into training data and later causes target forecast errors. In a retrieval-augmented system, related risks arise when poisoned or obsolete records propagate through the retrieval-and-fusion pipeline. Continual-learning methods such as CoSFan, Learning Fast and Slow, CABLE, and CLeaR provide adaptation mechanisms Missel and Wang (2025); Pham et al. (2022); Julian et al. (2025); He and Sick (2021), but they do not yet provide a general defense against adversarial memory injection or stale-memory reuse.

The roadmap response is to make validation and deletion first-class memory-interface operations. A forecaster should be able to score retrieved entries for freshness and consistency, quarantine suspicious records, and test whether removing a memory entry improves robustness. This is especially important for external stores, replay buffers, and evolving knowledge bases, where stored items may persist long after the regime that produced them has disappeared.

6.4 Evaluation Mismatch

Current MTS papers usually report point forecasting error, probabilistic calibration, or benchmark rank. These metrics do not isolate memory behavior. They do not show whether a retrieved case was relevant, whether an update reduced drift damage, whether a replay buffer protected useful old regimes, or whether a stale entry caused negative transfer. Benchmark efforts such as TSFM-Bench Li et al. (2025) and Context is Key Williams et al. (2025) advance evaluation across foundation-model and context-aware forecasting settings, while broader MTS surveys summarize standard benchmark practice Song et al. (2024); Shao et al. (2024). Memory-specific evaluation remains less standardized.

The roadmap is to test memory mechanisms directly. Useful stress tests include no-memory ablations, stale-memory ablations, corrupted-memory perturbations, delayed updates, memory-size scaling, recurring regimes, channel-structure shifts, and retrieval-attribution checks. The evaluation protocol in Table 4 follows this logic. Such tests would make it harder to confuse architectural capacity with memory-specific capability.

6.5 Hallucination in Time-Series Foundation Models

As foundation models are increasingly used for forecasting, they inherit a version of the hallucination problem: forecasts or explanations may appear confident while diverging from the temporal dynamics supported by the input context. Zou et al. Zou et al. (2025) define hallucinations for time-series foundation models by checking whether generated forecasts exhibit dynamics inconsistent with the input context. In memory mechanisms, hallucination can arise when a generator overrides retrieved evidence with pretrained or language-level priors, or when textual explanations describe spurious temporal causes.

The roadmap response is evidence-grounded generation. Forecasts should be checked against retrieved temporal evidence, uncertainty estimates, and known signal constraints. LLM-empowered forecasters and agentic forecasting pipelines make this especially important because they may combine numerical prediction, retrieval, tool use, and natural-language explanation Cheng et al. (2025); Liu et al. (2025b,c, 2024). Such an explanation is useful only if the system can trace what it read, why it read it, and how strongly it affected the output.

6.6 Toward a Theory of What to Store

Across the surveyed literature, there is not yet a shared theory of what should be stored. Parameters, hidden states, retrieved cases, correlation matrices, prompts, adapters, replay buffers, and normalized statistics all function as memory, but they differ in capacity, stability, interpretability, cost, and update behavior. State-space models and Mamba-style architectures illustrate the architectural side of this question by offering efficient long-range state propagation Wang et al. (2024); Ahamed and Cheng (2024); Behrouz et al. (2024); Liang et al. (2024). Time-series foundation models raise the same question at scale: pretrained temporal priors may support transfer,

but targeted update and deletion remain difficult Ansari et al. (2024); Rasul et al. (2023); Goswami et al. (2024); Gao et al. (2024); Shi et al. (2025); He et al. (2025).

The roadmap is to connect statistical structure with memory design. Recurrent regimes may be stored as prototypes or adapters; rare but high-impact intervals may need high-fidelity exemplars; fast channel drift may require lightweight statistics; stable seasonal structure may belong in architecture-level representations; and domain events may require heterogeneous retrieval. This direction also motivates memory observability: systems should report what they stored, when it was refreshed, why it was deleted, and how these operations changed forecast behavior.

Summary. Open challenges and roadmap directions are two sides of the same memory-mechanism analysis. Challenges describe where current representations, reads, and updates are brittle; roadmap directions describe how future systems could expose read, write, refresh, delete, validation, and verification interfaces. This relationship is what turns the final part of the survey from a list of open problems into a research agenda for auditable memory mechanisms in MTS forecasting.

7 Conclusion

Memory mechanisms have become an increasingly important but unevenly exposed ingredient in multivariate time-series forecasting. This survey has used *memory mechanism* as a unified term for components that retain information beyond the instantaneous input transformation and use that retained information to condition, adapt, or verify forecasts. The formal definition in Equation 1 turns this idea into three operational questions: where forecasting-relevant information is represented, how it is accessed, and how it changes when new evidence arrives. On this basis, the survey organizes the literature around architecture-level, retrieval, and evolution memory mechanisms.

The resulting view highlights an asymmetric but informative landscape. Architecture-level memory is already widespread in contemporary MTS models through parameters, hidden states, attention contexts, state-space dynamics, adapters, structured operators, and pretrained representations. Explicit retrieval and evolution mechanisms, by contrast, are less consistently exposed as interfaces that can be inspected, refreshed, consolidated, or forgotten. This observation should not be read as a ranking of methods. It is instead a mechanism map: many strong forecasting systems already exploit retained information, but much of that memory behavior remains embedded inside model computation rather than articulated as a controllable system-level design choice.

Looking forward, progress in memory-based MTS forecasting will depend on making memory mechanisms more auditable and operational. Future systems need clearer read interfaces for retrieving regimes, events, correlations, examples, or knowledge; stronger update and deletion policies for drift, stale records, and corrupted evidence; and verification procedures that test whether memory improves forecasts for the right reasons. Evaluation should therefore move beyond aggregate forecasting error and directly stress retrieval quality, update stability, forgetting safety, robustness, and

explanation faithfulness. In this sense, the next stage is not merely to add more memory, but to design memory mechanisms whose read, update, delete, and verification behavior can be observed, controlled, and trusted.

Declarations

- **Funding** This research is supported by the National Key Research and Development Program of China under Grant No. 2024YFF0908200, the National Natural Science Foundation of China under Grant No. 62376043, the Sichuan Provincial Natural Science Foundation under Grant No. 2024NSFTD0008, and the Science and Technology Program of Chengdu under Grant No. 2025-GH02-00020-HZ.

- **Competing interests** The authors have no relevant financial or non-financial interests to disclose.
- **Data availability** No new experimental datasets were generated in this survey. The manuscript synthesizes information from published literature; the reviewed evidence categories are documented in the manuscript tables and appendices.
- **Code availability** Not applicable.
- **Ethics approval** Not applicable.
- **Consent to participate** Not applicable.
- **Consent for publication** Not applicable.
- **Author contributions** Sibo Qi contributed to the conceptualization, literature screening, taxonomy design, evidence synthesis, figure preparation, and manuscript drafting. Peng Chen contributed to conceptual supervision, methodology refinement, interpretation of the survey framework, and critical revision of the manuscript. Wuman Luo contributed to supervision, research framing, manuscript organization, and critical revision. All authors reviewed and approved the final manuscript.
- **Use of AI tools** AI-based tools were used to support language polishing, LaTeX formatting, and illustrative figure preparation. The authors reviewed and approved the final manuscript and take full responsibility for its content.

References

Ahamed, M.A., Cheng, Q.: TimeMachine: A Time Series is Worth 4 Mambas for Long-term Forecasting. 27th European Conference on Artificial Intelligence (ECAI 2024); arXiv:2403.09898 (2024). https://doi.org/10.48550/arXiv.2403.09898 . https://arxiv.org/abs/2403.09898

Ansari, A.F., Stella, L., Turkmen, C., Zhang, X., Mercado, P., Shen, H., Shchur, O., Rangapuram, S.S., Arango, S.P., Kapoor, S., *et al.*: Chronos: Learning the language of time series. arXiv preprint arXiv:2403.07815 (2024) https://doi.org/10.48550/arXiv.2403.07815 arXiv:2403.07815

Benechehab, A., Feofanov, V., Paolo, G., Thomas, A., Filippone, M., Kégl, B.: Adapts: Adapting univariate foundation models to probabilistic multivariate time series forecasting. In: Proceedings of the 42nd International Conference on Machine Learning

(2025). https://openreview.net/forum?id=yeICCRy3lE

Benidis, K., Rangapuram, S.S., Flunkert, V., Wang, Y., Maddix, D., Turkmen, C., Gasthaus, J., Bohlke-Schneider, M., Salinas, D., Stella, L., Aubet, F.-X., Callot, L., Januschowski, T.: Deep learning for time series forecasting: Tutorial and literature survey. ACM Computing Surveys **55**(6), 1–36 (2022) https://doi.org/10.1145/3533382

Behrouz, A., Santacatterina, M., Zabih, R.: Chimera: Effectively modeling multivariate time series with 2-dimensional state space models. arXiv preprint arXiv:2406.04320 (2024) https://doi.org/10.48550/arXiv.2406.04320 arXiv:2406.04320

Cheng, H., *et al.*: Position: Beyond model-centric prediction – agentic time series forecasting. arXiv preprint arXiv:2602.01776 (2025) https://doi.org/10.48550/arXiv.2602.01776 arXiv:2602.01776

Cao, F., Dai, L., Han, J., Xiong, H.: Enhancing multivariate time series forecasting with global temporal retrieval. In: International Conference on Learning Representations (2026). https://openreview.net/forum?id=QUJBPSfyui

Cirstea, R.-G., Guo, C., Yang, B., Kieu, T., Dong, X., Pan, S.: Triformer: Triangular, variable-specific attentions for long sequence multivariate time series forecasting. In: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (2022). https://doi.org/10.24963/ijcai.2022/277 . https://doi.org/10.24963/ijcai.2022/277

Chen, Z., Ma, M., Li, T., Wang, H., Li, C.: Long sequence time-series forecasting with deep learning: A survey. Information Fusion **97**, 101819 (2023) https://doi.org/10.1016/j.inffus.2023.101819

Cheng, H., Wu, X., Shu, Y., Rao, Z., Pan, L., Yang, B., Guo, C.: Cora: Boosting time series foundation models for multivariate forecasting through correlation-aware adapter. In: International Conference on Learning Representations (2026). https://openreview.net/forum?id=JRlNrcTllN

Darlow, L.N., Deng, Q., Hassan, A., Asenov, M., Singh, R., Joosen, A., Barker, A., Storkey, A.: Dam: Towards a foundation model for forecasting. In: International Conference on Learning Representations (2024). https://openreview.net/forum?id=4NhMhElWqP

De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T.: A continual learning survey: Defying forgetting in classification tasks. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(7), 3366–3385 (2022) https://doi.org/10.1109/TPAMI.2021.3057446

Dey, S., Nair, S.R.: ReMAP: Neural model reprogramming with network inversion

and retrieval-augmented mapping for adaptive motion forecasting. In: Advances in Neural Information Processing Systems, vol. 37 (2024). https://doi.org/10.52202/079017-0794

Dai, T., Wu, B., Liu, P., Li, N., Yuerong, X., Xia, S.-T., Zhu, Z.: Ddn: Dual-domain dynamic normalization for non-stationary time series forecasting. In: Advances in Neural Information Processing Systems, vol. 37 (2024). https://doi.org/10.52202/079017-3444

Fraikin, A.F., Bennetot, A., Allassonniere, S.: T-rep: Representation learning for time series using time-embeddings. In: International Conference on Learning Representations (2024). https://openreview.net/forum?id=3y2TfP966N

Faw, M., Sen, R., Zhou, Y., Das, A.: In-context fine-tuning for time-series foundation models. In: Proceedings of the 42nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 267, pp. 16355–16374. PMLR, Online (2025). https://proceedings.mlr.press/v267/faw25b.html

Fan, W., Zheng, S., Wang, P., Xie, R., Yi, K., Zhang, Q., Bian, J., Fu, Y.: In-flow: Instance normalization flow for non-stationary time series forecasting. In: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 295–306. ACM, New York, NY, USA (2025). https://doi.org/10.1145/3690624.3709260 . https://doi.org/10.1145/3690624.3709260

Gao, S., Koker, T., Queen, O., Hartvigsen, T., Tsiligkaridis, T., Zitnik, M.: Units: A unified multi-task time series model. In: Advances in Neural Information Processing Systems (2024). https://openreview.net/forum?id=nBOdYBptWW

Goswami, M., Szafer, K., Choudhry, A., Cai, Y., Li, S., Dubrawski, A.: Moment: A family of open time-series foundation models. arXiv preprint arXiv:2402.03885 (2024) https://doi.org/10.48550/arXiv.2402.03885 arXiv:2402.03885

Graves, A., Wayne, G., Danihelka, I.: Neural Turing machines. arXiv preprint arXiv:1410.5401 (2014) https://doi.org/10.48550/arXiv.1410.5401 arXiv:1410.5401

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S.G., Grefenstette, E., Ramalho, T., Agapiou, J., *et al.*: Hybrid computing using a neural network with dynamic external memory. Nature **538**(7626), 471–476 (2016) https://doi.org/10.1038/nature20101

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., Wang, H.: Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv preprint arXiv:2312.10997 (2023). https://doi.org/10.48550/arXiv.2312.10997 . https://arxiv.org/abs/2312.10997

Gama, J., Żliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Computing Surveys **46**(4), 1–37 (2014) https://doi.org/10.

1145/2523813

Hajirahimi, Z., Khashei, M.: Hybridization of hybrid structures for time series forecasting: a review. Artificial Intelligence Review **56**(2), 1201–1261 (2023) https://doi.org/10.1007/s10462-022-10199-0

Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997) https://doi.org/10.1162/neco.1997.9.8.1735

He, Y., Sick, B.: Clear: An adaptive continual learning framework for regression tasks. AI Perspectives (2021) https://doi.org/10.1186/s42467-021-00009-8

Hinder, F., Vaquet, V., Hammer, B.: One or two things we know about concept drift—a survey on monitoring in evolving environments. Part A: detecting concept drift. Frontiers in Artificial Intelligence **7**, 1330257 (2024) https://doi.org/10.3389/frai.2024.1330257

He, H., Yi, K., Ma, Y., Zhang, Q., Niu, Z., Pang, G.: Sempo: Lightweight foundation models for time series forecasting. In: Advances in Neural Information Processing Systems (2025). https://openreview.net/forum?id=YngHXbJM8g

Julian, J., Koh, Y.S., Bifet, A.: Building adaptive knowledge bases for evolving continual learning models. npj Artificial Intelligence (2025) https://doi.org/10.1038/s44387-025-00028-4

Jin, M., Koh, H.Y., Wen, Q., Zambon, D., Alippi, C., Webb, G.I., King, I., Pan, S.: A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **46**(12), 10466–10485 (2024) https://doi.org/10.1109/TPAMI.2024.3443141

Jin, G., Liang, Y., Fang, Y., Shao, Z., Huang, J., Zhang, J., Zheng, Y.: Spatio-temporal graph neural networks for predictive learning in urban computing: A survey. IEEE Transactions on Knowledge and Data Engineering **36**(10), 5388–5408 (2024) https://doi.org/10.1109/TKDE.2023.3333824

Karaouli, E., Coquenet, D., Fromont, E., Mermillod, M., Reyboz, M.: Are time series foundation models susceptible to catastrophic forgetting? arXiv preprint arXiv:2510.00809 (2025) https://doi.org/10.48550/arXiv.2510.00809 arXiv:2510.00809

Kraus, M., Divo, F., Dhami, D.S., Kersting, K.: xlstm-mixer: Multivariate time series forecasting by mixing via scalar memories. In: Advances in Neural Information Processing Systems (2025). https://openreview.net/forum?id=JlVn0XRpy0

Kim, J., Kim, H., Kim, H., Lee, D., Yoon, S.: A Comprehensive Survey of Time Series Forecasting: Architectural Diversity and Open Challenges. arXiv preprint

arXiv:2411.05793 (2024). https://doi.org/10.48550/arXiv.2411.05793 . https://arxiv.org/abs/2411.05793

Kang, J., Seo, J., Park, S., Han, S., Bae, S., Choe, H., Lee, S.: CRAFT: Channel-wise retrieval for multivariate time series forecasting. arXiv preprint arXiv:2604.05543 (2026) https://doi.org/10.48550/arXiv.2604.05543 arXiv:2604.05543. Accepted at ICASSP 2026 (Oral)

Khosla, S., Zhu, Z., He, Y.: Survey on Memory-Augmented Neural Networks: Cognitive Insights to AI Applications. arXiv preprint arXiv:2312.06141 (2023). https://doi.org/10.48550/arXiv.2312.06141 . https://arxiv.org/abs/2312.06141

Liu, Y., *et al.*: Frequency adaptive normalization for non-stationary time series forecasting. arXiv preprint arXiv:2409.20371 (2024) https://doi.org/10.48550/arXiv.2409.20371 arXiv:2409.20371

Liu, Z., Cheng, M., Li, Z., Huang, Z., Liu, Q., Xie, Y., Chen, E.: Adaptive normalization for non-stationary time series forecasting: A temporal slice perspective. In: Advances in Neural Information Processing Systems, vol. 36 (2023)

Lin, S., Chen, H., Wu, H., Qiu, C., Lin, W.: Temporal query network for efficient multivariate time series forecasting. In: Proceedings of the 42nd International Conference on Machine Learning (2025). https://openreview.net/forum?id=e24CueVty2

Liu, P., Guo, H., Dai, T., Li, N., Bao, J., Ren, X., Jiang, Y., Xia, S.: Calf: Aligning llms for time series forecasting via cross-modal fine-tuning. In: Proceedings of the AAAI Conference on Artificial Intelligence (2025). https://doi.org/10.1609/aaai.v39i18.34082 . https://doi.org/10.1609/aaai.v39i18.34082

Liu, Y., Guo, Q., Huang, X., Wang, J., Long, M.: AutoTimes: Autoregressive Time Series Forecasters via Large Language Models. arXiv (Cornell University) (2024). https://doi.org/10.48550/arxiv.2402.02370 . https://doi.org/10.48550/arxiv.2402.02370

Lu, J., Han, X., Sun, Y., Yang, S.: Cats: Enhancing multivariate time series forecasting by constructing auxiliary time series as exogenous variables. In: Proceedings of the 41st International Conference on Machine Learning (2024). https://openreview.net/forum?id=1lDAGDe0UR

Lu, J., Han, X., Yang, S.: Arm: Refining multivariate forecasting with adaptive temporal-contextual learning. In: International Conference on Learning Representations (2024). https://openreview.net/forum?id=JWpwDdVbaM

Liang, A., Jiang, X., Sun, Y., Shi, X., Li, K.: Bi-Mamba+: Bidirectional Mamba for Time Series Forecasting. arXiv:2404.15772 (2024). https://doi.org/10.48550/arXiv.2404.15772 . https://arxiv.org/abs/2404.15772

Lin, X., Liu, Z., Fu, D., Qiu, R., Tong, H.: BackTime: Backdoor attacks on multivariate time series forecasting. In: Advances in Neural Information Processing Systems (2024). https://openreview.net/forum?id=KI13lipxTW

Lee, S., Lee, J., Seo, J., Yoo, S., Kim, M., Lim, T.Y., Kang, D., Choi, H., Lee, S., Ahn, W.: Cross-RAG: Zero-shot retrieval-augmented time series forecasting via cross-attention. arXiv preprint arXiv:2603.14709 (2026) https://doi.org/10.48550/arXiv.2603.14709 arXiv:2603.14709

Liu, Y., Li, C., Wang, J., Long, M.: Koopa: Learning non-stationary time series dynamics with Koopman predictors. In: Advances in Neural Information Processing Systems, vol. 36 (2023)

Li, N., Laakom, F., Xu, Y., Wang, W., Schmidhuber, J.: Facts: A factored state-space framework for world modelling. In: International Conference on Learning Representations (2025). https://openreview.net/forum?id=dmCGjPFVhF

Li, T., Li, J., Zhu, Z.: Neural lad: A neural latent dynamics framework for times series modeling. In: Advances in Neural Information Processing Systems (2023). https://openreview.net/forum?id=bISkJSa5Td

Liu, C., Miao, H., Xu, Q., Zhou, S., Long, C., Zhao, Y., Li, Z., Zhao, R.: Efficient multivariate time series forecasting via calibrated language models with privileged knowledge distillation. In: 2025 IEEE 41st International Conference on Data Engineering (ICDE), pp. 3165–3178. IEEE, Piscataway, NJ, USA (2025). https://doi.org/10.1109/icde65448.2025.00237 . https://doi.org/10.1109/icde65448.2025.00237

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., Kiela, D.: Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474 (2020)

Li, Z., Qiu, X., Chen, P., Wang, Y., Cheng, H., Shu, Y., Hu, J., Guo, C., Zhou, A., Wen, Q., Jensen, C.S., Yang, B.: Tsfm-bench: A comprehensive and unified benchmark of foundation models for time series forecasting. In: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 5595–5606 (2025). https://doi.org/10.1145/3711896.3737442 . https://doi.org/10.1145/3711896.3737442

Liu, Y., Qin, G., Huang, X., Wang, J., Long, M.: Timer-xl: Long-context transformers for unified time series forecasting. In: International Conference on Learning Representations (2025). https://openreview.net/forum?id=KMCJXjIDDr

Lau, Y.-y.A., Shao, Z., Yeung, D.-Y.: Fast and slow streams for online time series forecasting without information leakage. In: International Conference on Learning Representations (2025). https://openreview.net/forum?id=I0n3EyogMi

Lee, T.L., Toner, W., Singh, R., Joosen, A., Asenov, M.: Lightweight online adaption for time series foundation model forecasts. In: Proceedings of the 42nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 267, pp. 33736–33764. PMLR, Online (2025). https://proceedings.mlr.press/v267/lee25ag.html

Liu, C., Xu, Q., Hao, M., Yang, S., Zhang, L., Long, C., Li, Z., Zhao, R.: Timecma: Towards llm-empowered multivariate time series forecasting via cross-modality alignment. In: Proceedings of the AAAI Conference on Artificial Intelligence (2025). https://doi.org/10.1609/aaai.v39i18.34067 . https://doi.org/10.1609/aaai.v39i18.34067

Liu, Q., Xu, C., Jiang, W., Wang, K., Ma, L., Li, H.: Timestacker: A novel framework with multilevel observation for capturing nonstationary patterns in time series forecasting. In: Proceedings of the 42nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 267, pp. 39929–39946. PMLR, Online (2025). https://proceedings.mlr.press/v267/liu25ch.html

Liu, J., Yang, L., Li, H., Hong, S.: Retrieval-augmented diffusion models for time series forecasting. In: Advances in Neural Information Processing Systems, vol. 37 (2024). https://doi.org/10.52202/079017-0091

Lim, B., Zohren, S.: Time-series forecasting with deep learning: a survey. Philosophical Transactions of the Royal Society A **379**(2194), 20200209 (2021) https://doi.org/10.1098/rsta.2020.0209

Li, L., Zhang, W., Yuan, S., Guo, H., Chen, W.: Physdiff: A physically-guided diffusion model for multivariate time series anomaly detection. In: Advances in Neural Information Processing Systems (2025). https://openreview.net/forum?id=ElTbpJp7b9

Ma, X., Ni, Z.-L., Xiao, S., Chen, X.: Timepro: Efficient multivariate long-term time series forecasting with variable- and time-aware hyper-state. In: Proceedings of the 42nd International Conference on Machine Learning (2025). https://openreview.net/forum?id=s69Ei2VrIW

Missel, R., Wang, L.: Continual slow-and-fast adaptation of latent neural dynamics (cosfan): Meta-learning what-how and when to adapt. In: International Conference on Learning Representations (2025). https://openreview.net/forum?id=Dl3MsjaIdp

Nzoyem, R.D., Keshtmand, N., Crespo-Fernandez, E., TSAYEM, I., Santos-Rodriguez, R., Barton, D.A.W., Deakin, T.: Weight-space linear recurrent neural networks. In: International Conference on Learning Representations (2026). https://openreview.net/forum?id=zHdKaF3ZM7

Nie, Y., Nguyen, N.H., Sinthong, P., Kalagnanam, J.: A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. arXiv (Cornell University) (2022). https://doi.org/10.48550/arxiv.2211.14730 . https://doi.org/10.

48550/arxiv.2211.14730

Ning, K., Pan, Z., Liu, Y., Jiang, Y., Zhang, J.Y., Rasul, K., Schneider, A., Ma, L., Nevmyvaka, Y., Song, D.: TS-RAG: Retrieval-Augmented Generation based Time Series Foundation Models are Stronger Zero-Shot Forecaster. ArXiv.org (2025). https://doi.org/10.48550/arxiv.2503.07649 . https://doi.org/10.48550/arxiv.2503.07649

Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S.: Continual lifelong learning with neural networks: A review. Neural Networks **113**, 54–71 (2019) https://doi.org/10.1016/j.neunet.2019.01.012

Pham, Q., Liu, C., Sahoo, D., Hoi, S.C.H.: Learning Fast and Slow for Online Time Series Forecasting. arXiv (Cornell University) (2022). https://doi.org/10.48550/arxiv.2202.11672 . https://doi.org/10.48550/arxiv.2202.11672

Rasul, K., Ashok, A., Williams, A.R., Ghonia, H., Bhagwatkar, R., Khorasani, A., Bayazi, M.J.D., Adamopoulos, G., Riachi, R., Hassen, N., *et al.*: Lag-llama: Towards foundation models for probabilistic time series forecasting. arXiv preprint arXiv:2310.08278 (2023) https://doi.org/10.48550/arXiv.2310.08278 arXiv:2310.08278

Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Gruber, L., Holzleitner, M., Adler, T., Kreil, D., Kopp, M.K., Klambauer, G., Brandstetter, J., Hochreiter, S.: Hopfield networks is all you need. In: International Conference on Learning Representations (2021). https://openreview.net/forum?id=tL89RnzIiCd

Suárez-Cetrulo, A.L., Quintana, D., Cervantes, A.: A survey on machine learning for recurring concept drifting data streams. Expert Systems with Applications **213**, 118934 (2023) https://doi.org/10.1016/j.eswa.2022.118934

Song, X., Deng, L., Wang, H., Zhang, Y., He, Y., Cao, W.: Deep learning-based time series forecasting. Artificial Intelligence Review (2024) https://doi.org/10.1007/s10462-024-10989-8

Song, J., Kim, K., Oh, J., Cho, S.: MEMTO: Memory-guided transformer for multivariate time series anomaly detection. In: Advances in Neural Information Processing Systems, vol. 36 (2023)

Song, Y., Lu, J., Lu, H., Zhang, G.: Learning data streams with changing distributions and temporal dependency. IEEE Transactions on Neural Networks and Learning Systems **34**(8), 3952–3965 (2023) https://doi.org/10.1109/TNNLS.2021.3122531

Srinivas, S.S., Ravuru, C., Sannidhi, G., Runkana, V.: Reprogramming Foundational Large Language Models(LLMs) for Enterprise Adoption for Spatio-Temporal Forecasting Applications: Unveiling a New Era in Copilot-Guided Cross-Modal Time Series Representation Learning. arXiv (Cornell University) (2024). https:

//doi.org/10.48550/arxiv.2408.14387 . https://doi.org/10.48550/arxiv.2408.14387

Sukhbaatar, S., Szlam, A., Weston, J., Fergus, R.: End-to-end memory networks. In: Advances in Neural Information Processing Systems, vol. 28, pp. 2440–2448 (2015)

Shi, X., Wang, S., Nie, Y., Li, D., Ye, Z., Wen, Q., Jin, M.: Time-moe: Billion-scale time series foundation models with mixture of experts. In: International Conference on Learning Representations (2025). https://openreview.net/forum?id=e1wDDFmlVu

Shao, Z., Wang, F., Xu, Y., Wei, W., Yu, C., Zhang, Z., Yao, D., Sun, T., Jin, G., Cao, X., Cong, G., Jensen, C.S., Cheng, X.: Exploring progress in multivariate time series forecasting: Comprehensive benchmarking and heterogeneity analysis. IEEE Transactions on Knowledge and Data Engineering (2024) https://doi.org/10.1109/tkde.2024.3484454

Sun, Y., Xie, Z., Eldele, E., Chen, D., Hu, Q., Wu, M.: Learning pattern-specific experts for time series forecasting under patch-level distribution shift. In: Advances in Neural Information Processing Systems (2025). https://openreview.net/forum?id=CtoIG9Iwas

Tan, M., Merrill, M.A., Gupta, V., Althoff, T., Hartvigsen, T.: Are language models actually useful for time series forecasting? In: Advances in Neural Information Processing Systems (2024). https://openreview.net/forum?id=DV15UbHCY1

Tire, K., Taga, E.O., Ildiz, M.E., Oymak, S.: Retrieval Augmented Time Series Forecasting. arXiv (Cornell University) (2024). https://doi.org/10.48550/arxiv.2411.08249 . https://doi.org/10.48550/arxiv.2411.08249

Williams, A.R., Ashok, A., Marcotte, É., Zantedeschi, V., Subramanian, J., Riachi, R., Requeima, J., Lacoste, A., Rish, I., Chapados, N., Drouin, A.: Context is key: A benchmark for forecasting with essential textual information. In: Proceedings of the 42nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 267, pp. 66887–66944. PMLR, Online (2025). https://proceedings.mlr.press/v267/williams25a.html

Wu, D., Hu, J.Y.-C., Li, W., Chen, B.-Y., Liu, H.: STanHop: Sparse tandem Hopfield model for memory-enhanced time series prediction. In: International Conference on Learning Representations (2024). https://openreview.net/forum?id=6iwg437CZs

Wang, Z., Kong, F., Feng, S., Wang, M., Zhao, H., Wang, D., Zhang, Y.: Is mamba effective for time series forecasting? In: arXiv Preprint arXiv:2403.11144 (2024). https://doi.org/10.48550/arXiv.2403.11144 . https://arxiv.org/abs/2403.11144

Woo, G., Liu, C., Kumar, A., Xiong, C., Savarese, S., Sahoo, D.: Unified Training of Universal Time Series Forecasting Transformers. arXiv (Cornell University) (2024). https://doi.org/10.48550/arxiv.2402.02592 . https://doi.org/10.48550/arxiv.2402.

02592

Wu, Y., Rabe, M.N., Hutchins, D., Szegedy, C.: Memorizing transformers. In: International Conference on Learning Representations (2022). https://openreview.net/forum?id=TrjbxzRcnf-

Wu, H., Xu, J., Wang, J., Long, M.: Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In: Advances in Neural Information Processing Systems, vol. 34, pp. 22419–22430 (2021)

Wang, C., Zhuang, Z., Qi, Q., Wang, J., Wang, X., Sun, H., Liao, J.: Drift doesn't matter: Dynamic decomposition with diffusion reconstruction for unstable multivariate time series anomaly detection. In: Advances in Neural Information Processing Systems, vol. 36 (2023)

Wang, L., Zhang, X., Su, H., Zhu, J.: A comprehensive survey of continual learning: Theory, method and application. IEEE Transactions on Pattern Analysis and Machine Intelligence **46**(8), 5362–5383 (2024) https://doi.org/10.1109/TPAMI.2024.3367329

Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., Sun, L.: Transformers in time series: A survey. In: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, pp. 6778–6786 (2023). https://doi.org/10.24963/ijcai.2023/759

Xu, K., Chen, L., Wang, S.: Kolmogorov-Arnold Networks for Time Series: Bridging Predictive Power and Interpretability. arXiv (Cornell University) (2024). https://doi.org/10.48550/arxiv.2406.02496 . https://doi.org/10.48550/arxiv.2406.02496

Xiao, M., Jiang, Z., Qian, L., Chen, Z., He, Y., Xu, Y., Jiang, Y., Li, D., Weng, R.-L., Peng, M., Huang, J., Ananiadou, S., Xie, Q.: Retrieval-augmented Large Language Models for Financial Time Series Forecasting. ArXiv.org (2025). https://doi.org/10.48550/arxiv.2502.05878 . https://doi.org/10.48550/arxiv.2502.05878

Xu, J., Zhang, W., Jing, X., Nie, J., Chen, S., Zhang, S.: Cpiri: Channel permutation-invariant relational interaction for multivariate time series forecasting. In: International Conference on Learning Representations (2026). https://openreview.net/forum?id=tgnXCCjKE3

Yang, Y., Chen, S., Hu, P., Shen, Z., Zhang, Y., Sun, Z., Li, S., Chen, Z., Fukumizu, K.: Conditionally whitened generative models for probabilistic time series forecasting. In: International Conference on Learning Representations (2026). https://doi.org/10.48550/arXiv.2509.20928 . https://arxiv.org/abs/2509.20928

Yao, H., Jin, M., Zhang, J.Y., *et al.*: Estimating time series foundation model transferability via in-context learning. arXiv preprint arXiv:2509.23695 (2025) https://doi.org/10.48550/arXiv.2509.23695 arXiv:2509.23695

Yang, S., Wang, D., Zheng, H., Jin, R.: TimeRAG: Boosting LLM time series forecasting via retrieval-augmented generation. arXiv preprint arXiv:2412.16643 (2024) https://doi.org/10.48550/arXiv.2412.16643 arXiv:2412.16643

Ye, W., Xu, Z., Gui, N.: Non-stationary diffusion for probabilistic time series forecasting. In: Proceedings of the 42nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 267, pp. 72112–72130. PMLR, Online (2025). https://doi.org/10.48550/arXiv.2505.04278 . https://proceedings.mlr.press/v267/ye25i.html

Yi, K., Zhang, Q., Fan, W., He, H., Hu, L., Wang, P., An, N., Cao, L., Niu, Z.: FourierGNN: Rethinking Multivariate Time Series Forecasting from a Pure Graph Perspective. arXiv (Cornell University) (2023). https://doi.org/10.48550/arxiv.2311.06190 . https://doi.org/10.48550/arxiv.2311.06190

Zhang, H., Chang, X., Zhang, Y., Zhang, Z., Wang, L., Bian, J., Tan, T.: TimeRAF: Retrieval-Augmented Foundation model for Zero-shot Time Series Forecasting. arXiv (Cornell University) (2024). https://doi.org/10.48550/arxiv.2412.20810 . https://doi.org/10.48550/arxiv.2412.20810

Zhao, Z., Liu, H., Prakash, B.A.: Tackling time-series forecasting generalization via mitigating concept drift. In: International Conference on Learning Representations (2026). https://openreview.net/forum?id=emkvZ7NanK

Zhang, H., Nie, P., Sun, L., Boulet, B.: Nearest neighbor multivariate time series forecasting. IEEE Transactions on Neural Networks and Learning Systems (2024) https://doi.org/10.1109/tnnls.2024.3490603

Zheng, V.Z., Sun, L.: Multivariate probabilistic time series forecasting with correlated errors. In: Advances in Neural Information Processing Systems (2024). https://openreview.net/forum?id=cAFvxVFaii

Zhao, L., Shen, Y.: Proactive model adaptation against concept drift for online time series forecasting. In: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 2020–2031. ACM, New York, NY, USA (2025). https://doi.org/10.1145/3690624.3709210 . https://doi.org/10.1145/3690624.3709210

Zou, Y., Wang, Z., Klabjan, D., Liu, H.: Investigating hallucinations of time series foundation models through signal subspace analysis. In: Advances in Neural Information Processing Systems (2025). https://openreview.net/forum?id=vgfG8sEVf9

Zhang, Y.-F., Wen, Q., Wang, X., Chen, W., Sun, L., Zhang, Z., Wang, L., Jin, R., Tan, T.: OneNet: Enhancing time series forecasting models under concept drift by online ensembling. In: Advances in Neural Information Processing Systems (2023). https://openreview.net/forum?id=Q25wMXsaeZ

Zhang, K., Wen, Q., Zhang, C., Cai, R., Jin, M., Liu, Y., Zhang, J.Y., Liang, Y., Pang, G., Song, D., Pan, S.: Self-supervised learning for time series analysis: Taxonomy, progress, and prospects. IEEE Transactions on Pattern Analysis and Machine Intelligence **46**(10), 6775–6794 (2024) https://doi.org/10.1109/TPAMI.2024.3387317

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., Zhang, W.: Informer: Beyond efficient transformer for long sequence time-series forecasting. Proceedings of the AAAI Conference on Artificial Intelligence **35**(12), 11106–11115 (2021) https://doi.org/10.1609/aaai.v35i12.17325

A Detailed Search and Taxonomy Protocol

This appendix records the search and classification details behind the compact methodology in Section 2.2. The main paper uses these details for transparency, while the synthesis itself is organized around memory mechanisms.

A.1 Search sources

The search began from a local corpus of approximately 80,000 papers from ICLR, ICML, NeurIPS, CVPR, KDD, ACM MM, CHI, ACL, and EMNLP over 2023–2026. This corpus was supplemented through OpenAlex, OpenReview, CrossRef, ACL Anthology, CVF Open Access, public scholarly records, and targeted journal expansion covering survey and method papers from venues such as ACM Computing Surveys, IEEE TPAMI, Information Fusion, Neural Computation, Frontiers in AI, Expert Systems with Applications, Artificial Intelligence Review, and related journals.

A.2 Pillar-targeted queries

OpenAlex queries were grouped by the three survey pillars. Architecture queries included "memory augmented multivariate time series forecasting", "deep memory networks time series forecasting", "state space model memory time series forecasting", "time series foundation model forecasting memory", "large language model time series forecasting memory", and "agent time series forecasting memory reasoning". Retrieval queries included "retrieval augmented time series forecasting", "nearest neighbor time series forecasting", "similarity search time series forecasting", "in context learning time series forecasting retrieval", and "vector database retrieval temporal representation forecasting". Evolution queries included "non stationary multivariate time series forecasting concept drift", "continual learning time series forecasting catastrophic forgetting", "online time series forecasting distribution shift memory", "experience replay time series forecasting", and "adaptive time series forecasting concept drift deep learning". Support queries covered memory poisoning, agent memory, continual-learning replay, and classical differentiable memory.

A.3 Classic-title queries

Additional queries targeted foundational works on Neural Turing Machines, End-to-End Memory Networks, Differentiable Neural Computers, Modern Hopfield Networks, RAG, REALM, Memorizing Transformers, generative agents, Reflexion, Voyager, and established MTS baselines including Informer, Autoformer, FEDformer, PatchTST, TimesNet, iTransformer, Time-LLM, TEMPO, and Learning Fast and Slow.

A.4 Scoring and screening

Candidates were deduplicated by DOI, arXiv ID, and normalized title. Each candidate was scored using lexical match, semantic embedding similarity, citation impact, and recency:

$$0.65 \times \text{semantic} + 0.25 \times \min(\text{lexical}/15.0, 1.0) + 0.06 \times \text{citation} + 0.04 \times \text{recency}.$$

The lexical component weighted MTS terms, memory terms, pillar-specific terms, and joint matches of forecasting and time-series terminology. A small penalty was applied to theory-support papers so that mechanism-bearing MTS papers were prioritized in the core set.

A.5 Evidence levels

The evidence set is separated into three levels. The PDF-verified core set supports method-level mechanism claims and boundary decisions. The broader trend set supports aggregate distributions and field-level patterns. A background support table provides classical memory architectures, adjacent surveys, and contextual work used for positioning rather than counted MTS evidence.

A.6 Taxonomy labeling

Core papers were labeled along three binary axes: Architecture, Retrieval, and Evolution. Architecture covers retained information in parameters, hidden states, attention contexts, patch representations, state-space dynamics, graph/frequency structures, adapters, or foundation-model priors. Retrieval requires an explicit read from a store, index, knowledge base, exemplar set, or memory bank. Evolution requires an explicit update, consolidation, replay, drift-adaptation, or forgetting policy. Full-text review of uncertain cases reduced over-broad labels caused by terms such as “query”, “attention”, “context”, “adaptive”, or “online” when those terms referred to internal computation rather than an exposed memory interface.

B Evidence-Set Distributions

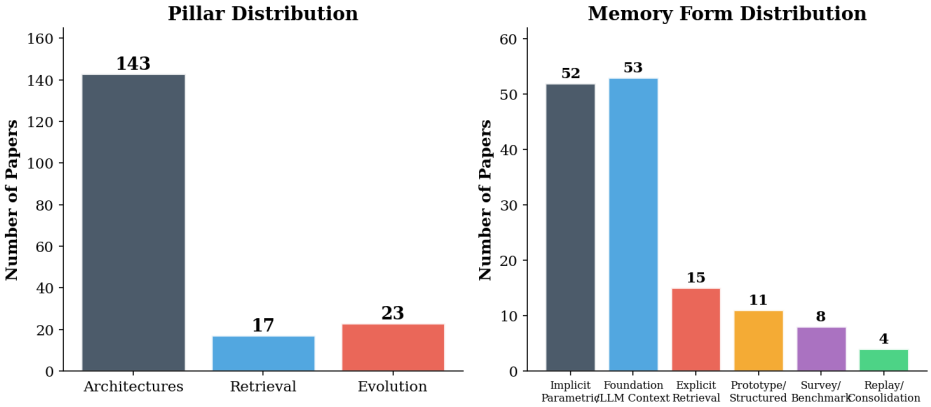


Fig. 8: Aggregate distribution of papers across the three memory pillars and taxonomy categories in the trend-level evidence set.

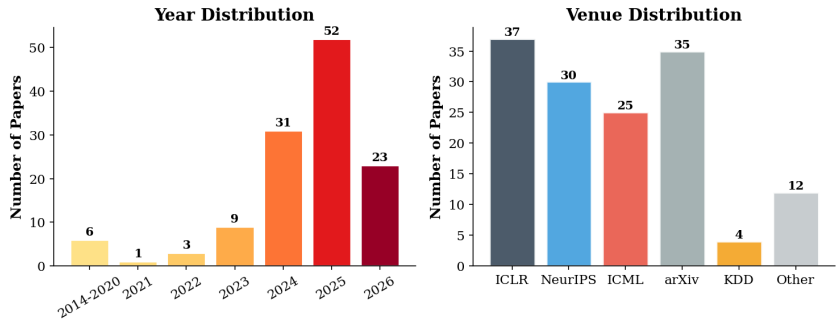


Fig. 9: Distribution of papers by year and publication venue in the trend-level evidence set.

Table 6: Quantitative distribution of the closely reviewed core evidence set across the three survey pillars, taxonomy categories, publication years, and venues.

Pillar Distribution		**Category Distribution**	
Architecture only	27	Implicit Parametric / State Memory	17
Architecture + Retrieval	6	Foundation / LLM Context Memory	8
Architecture + Evolution	6	Explicit Retrieval / Exemplar Memory	6
Architecture + Retrieval + Evolution	1	Replay / Consolidation Memory	4
		Prototype / Structured Matrix Memory	3
		Survey / Benchmark Support	2
Year Distribution		**Venue Distribution**	
2017	1	ICLR	10
2021	1	arXiv	12
2022	3	ICML	4
2023	2	NeurIPS	3
2024	15	AAAI	2
2025	14	Others (9 venues)	9
2026	4		

Summary: **Pillars:** 27 papers rely solely on architectural memory; 7 incorporate explicit retrieval; 7 incorporate memory evolution (1 paper spans all three). **Categories:** Implicit parametric/state memory is the largest group (17 papers, 42.5%), followed by foundation/LLM context memory (8, 20.0%), explicit retrieval (6, 15.0%), replay/consolidation (4, 10.0%), prototype/structured matrix (3, 7.5%), and survey/benchmark framing evidence (2, 5.0%). **Temporal trend:** 82.5% of core papers (33 of 40) appear between 2024 and 2026, consistent with the recent growth of memory-mechanism research in MTS forecasting. **Venue concentration:** ICLR accounts for 25% of core papers (10 of 40); together, the five most frequent sources (ICLR, arXiv, ICML, NeurIPS, AAAI) cover 77.5%.

C Retrieval and Evolution Evidence Boundaries

Table 7: Retrieval and evolution mechanisms in the closely reviewed core evidence set under the operational taxonomy. Arch., Ret., and Evol. denote architectural memory, retrieval memory, and memory evolution.

Method	Pillar	Read interface	Write/evolution policy	Evidence boundary
GTR Cao et al. (2026)	Arch.; Ret.	Global temporal retriever over full-cycle/global history embeddings; retrieved segments are aligned and fused with local input.	No memory evolution: RevIN stabilizes distribution shift, but the paper does not define online update, consolidation, or forgetting.	Retrieval is valid because the paper defines a plug-in retriever and global temporal embedding. Evolution is removed because alignment and normalization are not memory maintenance.
CoSFan Missel and Wang (2025)	Arch.; Evol.	Replay sampling from fixed-memory reservoirs; no semantic/indexed retrieval interface.	Continual slow-fast adaptation with experience replay, task-aware/task-relational strategies, and shift detection.	The paper is primarily a continual meta-learning architecture for non-stationary latent dynamics. It uses replay memory, but not retrieval in the RAG/query-index sense.
TS-RAG Ning et al. (2025)	Arch.; Ret.	Retriever queries a dedicated knowledge base and fuses retrieved knowledge through Adaptive Retrieval Mixer.	No explicit evolution: no online knowledge-base update, forgetting, or consolidation policy is defined.	Retrieval is explicit. Evolution is removed because the knowledge base is used for retrieval and fusion, not dynamically maintained.
Fast/Slow Pham et al. (2022)	Arch.; Evol.	None or implicit.	Continual update with consolidation and forgetting.	The method is treated as Evolution because it explicitly addresses online adaptation and retention, but it does not instantiate an external retrieval index.
T-KAN/MT-KAN Xu et al. (2024)	Arch.; Evol.	None.	Drift-aware modeling: T-KAN is presented as detecting concept drift and explaining nonlinear relationships under changing environments.	Retrieval is removed. Evolution is retained conservatively because the paper explicitly targets concept-drift detection, although it is not online memory maintenance.

Continued on next page

Table 7: Retrieval and evolution mechanisms in the closely reviewed core evidence set under the operational taxonomy (continued). Arch., Ret., and Evol. denote architectural memory, retrieval memory, and memory evolution.

Method	Pillar	Read interface	Write/evolution policy	Evidence boundary
CPiRi Xu et al. (2026)	Arch.; Evol.	None.	Structural/ distributional co-drift robustness: permutation-invariant relational modeling supports channel addition/reordering without retraining.	Evolution is retained because the method directly targets changing channel structure and co-drift, though it is not an online learner.
TimeRAF Zhang et al. (2024a)	Arch.; Ret.	Retrieval-augmented foundation/LLM context.	Static training or offline refresh.	Retrieval is retained because the forecasting process uses retrieved temporal context; evolution is not counted because the memory store is not maintained online.
NN MTSF Zhang et al. (2024b)	Arch.; Ret.	Nearest-neighbor subsequence retrieval.	Static training or offline refresh.	Retrieval is retained because historical subsequences are explicitly selected and used for prediction; no separate write, consolidation, or forgetting policy is defined.
WARP Nzoyem et al. (2026)	Arch.; Evol.	None.	Gradient-free test-time adaptation and in-context adaptation via recurrent weight-space state.	Retrieval is removed. Evolution is retained because the paper explicitly treats model behavior adaptation without gradients and in-context updating.
Retrieval TSF Tire et al. (2024)	Arch.; Ret.	Explicit temporal retrieval module.	Static training or offline refresh.	Retrieval is retained because the method introduces a temporal retrieval component; no online evolution policy is specified.

Continued on next page

Table 7: Retrieval and evolution mechanisms in the closely reviewed core evidence set under the operational taxonomy (continued). Arch., Ret., and Evol. denote architectural memory, retrieval memory, and memory evolution.

Method	Pillar	Read interface	Write/evolution policy	Evidence boundary
Financial RAG Xiao et al. (2025)	Arch.; Ret.	Retrieval-augmented foundation/LLM context.	Static training or offline refresh.	Retrieval is retained because external financial context is selected and fused with the forecasting process; memory evolution remains outside the method scope.
CABLE Julian et al. (2025)	Arch.; Ret.; Evol.	Knowledge-base lookup/routing via gradient similarity between new examples and past tasks/adapters.	Evolving continual-learning knowledge base with dynamic adapter routing and reuse of prior task knowledge.	The paper builds and routes through adaptive knowledge bases for continual learning. It is not MTS-specific, but it is relevant to retrieval and evolution mechanisms.
CLeaR He and Sick (2021)	Arch.; Evol.	None or implicit.	Continual update with consolidation and forgetting.	The method is treated as Evolution because it provides a continual-regression mechanism for balancing plasticity and retention, without an explicit external retrieval interface.

D Citation-Context Anchors for Challenge Claims

The challenge and roadmap discussions in Section 6 combine direct findings from cited work with survey-level synthesis. Table 8 records the page-level anchors used to keep those claims scoped.

Table 8: Citation-context anchors for challenge and evaluation synthesis claims. Page numbers refer to the source PDFs inspected during revision and are used as traceability anchors rather than quoted evidence.

Claim area	Anchored sources	Checked context	Boundary retained in the manuscript
Latency–utility trade-off	LLM-forecasting ablations Tan et al. (2024): p2, p5, p7, p20–p21. Retrieval and LLM-forecasting examples are discussed in the main text.	The checked ablations support the narrower claim that some LLM components can be removed or replaced with lighter modules while preserving comparable forecasting accuracy and reducing inference cost.	The manuscript treats latency as a design trade-off and does not claim a universal latency threshold for retrieval-augmented forecasting.
Temporal modality gap	TimeRAF Zhang et al. (2024a): p2, p7, p9, p16; nearest-neighbor MTS forecasting Zhang et al. (2024b): p1, p5–p6; retrieval-augmented TSF Tire et al. (2024): p2, p7, p9, p23; financial RAG forecasting Xiao et al. (2025): p1, p3–p4, p7.	The checked papers support the presence of explicit time-series retrieval stores, query construction, retrieved candidates, and fusion or conditioning mechanisms.	The broader modality-gap discussion remains a survey synthesis claim about continuous, multichannel temporal queries rather than a direct finding from a single benchmark.
Memory poisoning and corrupted memories	BackTime Lin et al. (2024): p1–p5, p9. Continual and replay-based methods are cited as adaptation mechanisms in Section 6.	BackTime supports the existence of stealthy trigger injection and target-pattern manipulation for MTS forecasting.	The manuscript distinguishes demonstrated MTS backdoor attacks from proposed retrieval-memory contamination tests.
Evaluation mismatch	TSFM-Bench Li et al. (2025): p2–p7, p9; Context is Key Williams et al. (2025): p1, p3, p8–p9, p30, p55; LLM ablations Tan et al. (2024): p2, p5, p7, p20–p21.	The checked sources support standardized foundation-model evaluation, textual-context forecasting evaluation, task-level failure modes, inference-time differences, and ablations of architectural complexity.	Table 4 is framed as a proposed memory-specific protocol, not as an existing community standard.

Claim area	Anchored sources	Checked context	Boundary retained in the manuscript
Hallucination in TSFMs	TSFM hallucination analysis Zou et al. (2025): p1–p3, p6–p8, p10, p27.	The checked source defines time-series foundation-model hallucination through generated dynamics that diverge from the input context and links failures to loss of context information in hidden states.	The manuscript avoids unsupported claims about universal uncertainty calibration failure and focuses on context-divergence and signal-subspace evidence.
Theory of what to store	Cross-section synthesis from the architecture, retrieval, and evolution evidence tables rather than a single source.	The reviewed papers use parameters, hidden states, retrieved cases, correlation structures, prompts, adapters, and replay buffers as memory-like objects with different capacity and update properties.	The manuscript presents this as an open theoretical opportunity, not as a settled result or benchmark finding.