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 in Deep Time-Series Models

Minh Hoang Nguyen
s223669184@deakin.edu.au
Deakin University
Huu Hiep Nguyen
Deakin University
Manh Nguyen
Deakin University
Van Dai Do
Deakin University
Dung Nguyen
Deakin University
Hung Le
Deakin University

Research Article

Keywords: Time series, External memory, Retrieval augmentation, Agentic memory, Foundation models, Survey

Posted Date: November 19th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11495678/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 in Deep Time-Series Models

Minh Hoang Nguyen[1*†], Huu Hiep Nguyen[1†], Manh Nguyen[1†], Van Dai Do[1†], Dung Nguyen[1†], Hung Le[1†]

[1*]Applied Artificial Intelligence Initiative (A^2I^2), Deakin University, Geelong, 3216, Victoria, Australia.

*Corresponding author(s). E-mail(s): s223669184@deakin.edu.au;
Contributing authors: s225250685@deakin.edu.au;
s224852302@deakin.edu.au; v.do@deakin.edu.au;
dung.nguyen@deakin.edu.au; thai.le@deakin.edu.au;
†These authors contributed equally to this work.

Abstract

Deep learning for time series has progressed through successive architectural paradigms, from recurrent networks and transformers to structured state-space models, retrieval-augmented predictors, foundation models, and tool-using agents. These developments are typically studied in isolation, organized by architecture or modeling era. We argue that they can instead be viewed through a common question of *how does a time-series model retain and access information beyond its immediate input?* This question is motivated by a fundamental limitation of conventional time-series modeling: information relevant to a prediction may lie far beyond a feasible input window, while compressing history into a fixed-size state can discard information that may become useful later. We formulate this challenge as a *memory* problem and organize existing time-series methods along a spectrum from internal memory, encoded in parameters and fixed-size states, to external memory that is addressable, retrievable, and increasingly maintained by agents. We then develop a unified taxonomy of memory mechanisms and review three classes of external memory, including explicit modules, retrieval augmentation, and agentic stores, under a common framework for what is retained, how it is written and accessed, and how it persists. A cross-cutting analysis maps these mechanisms to time-series tasks and identifies gaps in both methods and evaluation. We conclude by outlining open problems in building memory systems that can selectively retain, retrieve, revise, and forget information as temporal environments evolve. The result is a framework for studying memory as a first-class dimension of time series modeling, independent

of the underlying backbone. The accompanying paper collection is available at https://github.com/DA2I2-SLM/Time-series-memory.

Keywords: Time series, External memory, Retrieval augmentation, Agentic memory, Foundation models, Survey

1 Introduction

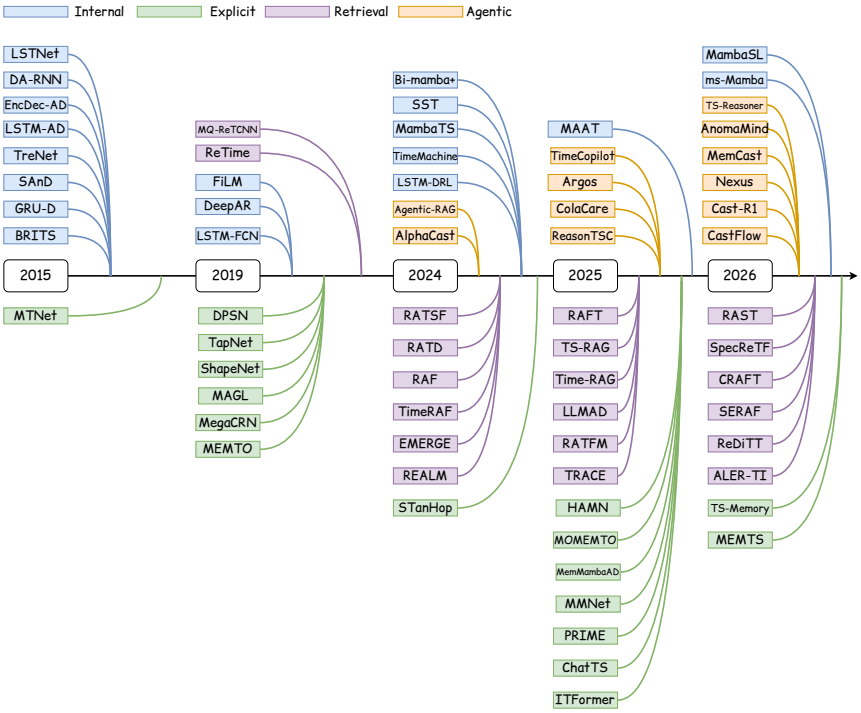


Fig. 1 Evolution of memory mechanisms in time-series modeling from 2015 to 2026. Representative methods are organized along a temporal axis and categorized according to four increasingly explicit memory paradigms.

Time series underpin applications ranging from energy (Tu et al. 2024; Jiang et al. 2024), materials science (Le et al. 2025), and finance (Shi et al. 2026) to healthcare (Xu et al. 2023; Staniek et al. 2024) and industrial monitoring (Alnegheimish et al. 2025), and the central modeling challenge is temporal dependence that reaches beyond any convenient input window. A decade of deep learning has produced a succession of architectural paradigms, including recurrence (Hochreiter and Schmidhuber 1997), attention (Vaswani et al. 2017), structured state (Gu et al. 2022; Gu and Dao 2023), retrieval (Liu et al. 2024; Han et al. 2025), and, more recently, foundation models

and agents (Garza et al. 2023; Tao et al. 2026). These paradigms are typically surveyed apart, as though each represents a distinct stage in the evolution of time series modeling. We argue that they can instead be understood through a common notion of *memory* (the mechanisms by which a model retains information from the past or broader context and accesses that information when producing an output). Recurrence compresses history into a hidden state, state-space models maintain a structured finite-dimensional summary, explicit memory modules retain addressable representations, retrieval systems use collections of observations as external memory, and agentic systems actively construct and maintain persistent stores. These mechanisms occupy different points on a common spectrum rather than forming disconnected architectural families. Figure 1 summarizes this evolution, tracing the progression from internal state-based memory toward increasingly explicit, retrievable, and actively maintained forms of memory. They differ primarily in whether memory is internal or external, whether its capacity is fixed or can grow with accumulated data, how information is written and read, and how long the resulting state persists.

While memory architectures have been extensively studied in Natural Language Processing (NLP) (Zhong et al. 2024; Borgeaud et al. 2022; Zhang et al. 2025), they remain comparatively under-surveyed in the time-series domain. Crucially, memory is not simply a lens that can be seamlessly transferred from language modeling because continuous temporal data impose requirements that make memory a fundamentally distinct modeling problem. Information relevant to a prediction often lies far outside any computationally feasible lookback window. Winter peak energy demand, for example, may depend entirely on what happened during the previous winter and a newly deployed machine may have little to no historical data of its own. Simply extending the context window is not always a practical solution, as longer sequences exponentially increase the cost of attention while burying important and rare observations given large amounts of irrelevant history. Conversely, compressing the past into a fixed-size hidden state creates the opposite problem. It forces the model to discard information that may later become important, especially when the useful signal is weak, rare, or separated from the current observation by a long time gap. Consequently, the specific mechanisms by which temporal memory is retained, retrieved, updated, and discarded must be treated as a distinct and central challenge in time-series modeling.

This motivates our focus on external memory, which we define as information that a model can read from, and in some cases write to, a store outside its learned parameters. We use internal memory as the conceptual baseline for this distinction, connecting recurrence and state-based approaches to this baseline while deferring detailed architectural comparisons to existing surveys (Wen et al. 2023; Hewamalage et al. 2021). Our review therefore focuses on how memory becomes explicit, retrievable, and actively maintained, and how these mechanisms support downstream time-series tasks. We organize the literature around forecasting, classification, anomaly detection, and imputation, while also examining emerging reasoning and decision-making settings in which memory may retain not only historical signals but also evidence, actions, outcomes, and evolving context.

Rather than organizing methods by backbone or retrieval mechanism, we classify them according to what they retain and how that information is written and accessed.

Table 1 Positioning against representative time series and memory surveys. Coverage of the memory spectrum: ✓ substantial, (∼) partial, blank little/none.

Survey	Internal	Explicit	Retrieval	Agentic	Problem view
Lim and Zohren (2021)	✓	(∼)			
Wen et al. (2023)	(∼)				
Jin et al. (2024)	(∼)	(∼)			
Liang et al. (2024)	(∼)		(∼)	(∼)	
Chang et al. (2025)			(∼)	✓	(∼)
This survey	(∼)	✓	✓	✓	✓

This perspective reveals common memory mechanisms across systems that use different terminology. For example, PatchTST (Nie et al. 2023) retains historical patches as separately addressable entries, while PM-MemNet stores representative historical patterns and retrieves those relevant to the current window (Lee et al. 2022). Despite their different terminology, both retain past information as addressable entries and selectively access it based on the current input. Their difference lies primarily in what is stored and how the entries are constructed, rather than in the underlying role of memory.

This perspective also changes how we position existing surveys. Prior surveys have largely been organized around architectures or broader modeling paradigms. Reviews of recurrent, transformer, and graph-based models typically treat memory as an implicit property of the backbone. In contrast, surveys of foundation models and agents focus on broader modeling or system paradigms. These perspectives are valuable, but they cover different parts of the memory spectrum and rarely connect them within a common framework. As summarized in Table 1, existing surveys provide substantial coverage of parametric and architectural approaches, while explicit, retrieval-based, and agentic memory receive limited or fragmented focus. Our survey instead makes memory the organizing principle, providing a unified view of memory. This complements rather than replaces architecture-centric and agent-centric accounts by focusing on what information persists, how it is written and retrieved, and when it is revised or discarded.

1.1 Contributions

Our contributions are as follows.

- **A formal framework for memory in time series.** We define memory in terms of what information is retained, how it is accessed, and how long it persists, and introduce four dimensions for comparing memory mechanisms: representation, capacity, access mechanism, and persistence.
- **A unified taxonomy of memory mechanisms.** We organize existing approaches along a spectrum from implicit parametric memory to explicit, retrieval-based, and agentic memory, together with a problem-by-architecture matrix that highlights common design choices across methods.

- **A systematic review of external memory.** We examine the three major classes of external memory under a common framework covering their mechanisms, subclasses, writing and integration strategies, and limitations. A formal read/write formulation places explicit, retrieval-based, and agentic memory within a common lifecycle.
- **A task-centric view of memory.** We analyze memory mechanisms across forecasting, classification, anomaly detection, imputation, reasoning, and decision-making, highlighting which memory mechanisms are well studied and which remain largely unexplored for each task.
- **A critical analysis of evaluation and resources.** We review existing datasets, benchmarks, and evaluation protocols and argue that current evaluations largely measure downstream task accuracy rather than memory operations themselves. We identify four measurements for evaluating memory retention, retrieval quality, temporal validity, memory management.
- **Open research directions.** We identify seven concrete challenges for building reliable future time-series memory systems, focusing on how memory can remain useful, efficient, and trustworthy as data and environments evolve.

Together, these contributions provide a framework for understanding how memory is represented, operated, and evaluated in deep time-series models.

1.2 Research Questions

To structure the review, we organize our investigation around four research questions:

- **RQ1: Memory mechanisms in time-series models.** How do deep time-series models represent, retain, update, and retrieve temporal information beyond the immediate input window?
- **RQ2: Memory architectures and temporal modeling.** What are the main memory paradigms in deep time-series models, and how do their representations, writing and retrieval mechanisms, and retention timescales influence their ability to capture temporal dependencies?
- **RQ3: Memory for time-series tasks.** How are different memory mechanisms used across forecasting, classification, anomaly detection, imputation, reasoning, and decision-making, and what task-specific benefits and limitations do they exhibit?
- **RQ4: Evaluation and open challenges.** How are memory capabilities evaluated in deep time-series models, and what challenges remain in measuring memory effectiveness, managing retained information, and adapting memory to non-stationary temporal data?

These questions provide a common thread throughout the survey. RQ1 establishes the operations and properties used to characterize memory, while RQ2 guides the taxonomy and architectural analysis in Sections 4 to 8. RQ3 motivates the task-centric analysis in Section 9, and RQ4 informs the discussion of evaluation resources and open challenges in Sections 10 to 11.

1.3 Paper Organisation

The remainder of the survey is organized as follows. Section 2 formalizes memory and introduces the dimensions used throughout the survey. Section 3 discusses why memory is particularly important for time series, while Section 4 presents the proposed taxonomy and problem-by-architecture matrix. Section 5 provides background on internal memory, and Sections 6 to 8 review the three external memory classes. Section 9 provides the task-centric view, followed by the resources and benchmarks in Section 10 and the open challenges in Section 11.

Terminology introduced throughout the survey, including the four memory classes, their subtypes, the four problem settings, and the evaluation measures of Section 10, is collected in a glossary in Supplementary Material A.

2 Preliminaries

2.1 A Generic Memory-Augmented Model

Let $\mathbf{x}_{1:T} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ denote a multivariate time series, where $\mathbf{x}_t \in \mathbb{R}^d$ and d is the number of variables or channels. At an operational step t, standard models receive an observation context $\mathbf{o}_t$, e.g., an input window such as $\mathbf{o}_t = \mathbf{x}_{t-L+1:t}$, while memory-augmented models additionally access information retained from observations outside this immediate context. We denote this retained information by the memory state $\mathcal{M}_t$. A generic memory-augmented model can therefore be expressed as

$$\begin{aligned} \mathcal{M}_t &= \mathcal{W}_\phi(\mathcal{M}_{t-1}, \mathbf{o}_t, Q), \\ \mathbf{r}_t &= \mathcal{R}_\psi(\mathcal{M}_t, \mathbf{o}_t, Q), \\ \hat{\mathbf{y}}_t &= \mathcal{P}_\theta(\mathbf{o}_t, \mathbf{r}_t, Q). \end{aligned} \tag{1}$$

where $\mathcal{W}_\phi$ writes information to memory, $\mathcal{R}_\psi$ reads information from memory, and $\mathcal{P}_\theta$ produces the task output. Here, Q denotes an optional task or query specification. This factorization separates three roles that recur throughout the survey: what is retained, how it is accessed, and how the retrieved information is used.

2.2 What is memory? A working definition

We use *time-series memory* to denote information that is structured and retained such that it remains conditionally accessible across immediate observation windows as well as extended historical horizons. Crucially, the difference between memory and standard parametric knowledge is drawn along the axis of *representation and access* (refer to Definition 1).

Definition 1 (Memory of a model) The *memory* of a time-series model refers to any mechanism that retains historical time-series information and makes it accessible for downstream inference. This may occur through direct access to a finite context window, a dynamically maintained state, or a separately maintained store. Memory mechanisms

therefore differ in how historical information is represented, accessed, updated, and persisted, as well as in their memory capacity. Information implicitly encoded in static model parameters (which remain fixed after training) is not considered memory. In contrast, dynamically updated parameter stores—such as fast weights that adapt during inference to retain recent history—function as accessible information and fall within our definition. We refer to this broad construct throughout the survey as "time-series memory."

Under Definition 1, the trained parameters θ (such as the convolutional filters or projection matrices of a network) do not constitute memory. They encode statistical regularities learned during training rather than retaining specific historical information for subsequent access. For example, parameter optimization updates the model according to:

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}, \tag{2}$$

thereby incorporating knowledge from the training data into the model's parameters. Memory, in contrast, makes historical information available to the predictor through a context window, dynamic state, or dedicated store, with different mechanisms providing different forms of access, persistence, and capacity.

2.3 Task Instantiations

To demonstrate the operational framework, governed by the write operator $\mathcal{W}_\phi$, read operator $\mathcal{R}_\psi$, and prediction operator $\mathcal{P}_\theta$, we show how it explicitly instantiates across downstream tasks. In each case, downstream computation relies on a composite pipeline: a query Q prompts the readout operator $\mathcal{R}_\psi$ to extract a contextual representation $\mathbf{r}_t = \mathcal{R}_\psi(\mathcal{M}_t, \mathbf{o}, Q)$ from memory, which the predictor $\mathcal{P}_\theta$ subsequently processes alongside the current observation. For readability, we define the functional shorthand $f_\theta(\mathbf{o}, \mathcal{M}_t) \equiv \mathcal{P}_\theta(\mathbf{o}, \mathcal{R}_\psi(\mathcal{M}_t, \mathbf{o}, Q), Q)$, collapsing retrieval and prediction into a single mapping.

Forecasting. Given a historical lookback window of length L, the predictor $\mathcal{P}_\theta$ conditions on input observations $\mathbf{x}_{t-L+1:t}$ and memory readout $\mathbf{r}_t$ to project a future horizon of H values:

$$\begin{aligned} \hat{\mathbf{x}}_{t+1:t+H} &= \mathcal{P}_\theta\Big(\mathbf{x}_{t-L+1:t}, \mathcal{R}_\psi(\mathcal{M}_t, \mathbf{x}_{t-L+1:t}, Q), Q\Big) \\ &\equiv f_\theta\big(\mathbf{x}_{t-L+1:t}, \mathcal{M}_t\big). \end{aligned} \tag{3}$$

Classification. Given a fully observed trajectory $\mathbf{x}_{1:T}$, the model queries the aggregated memory $\mathcal{M}$ to map the sequence to a discrete class distribution:

$$\begin{aligned} \hat{y} &= \mathcal{P}_\theta\Big(\mathbf{x}_{1:T}, \mathcal{R}_\psi(\mathcal{M}_T, \mathbf{x}_{1:T}, Q), Q\Big) \\ &\equiv f_\theta(\mathbf{x}_{1:T}, \mathcal{M}_T). \end{aligned} \tag{4}$$

Anomaly Detection. The model evaluates an incoming observation $\mathbf{o}_t$ against context retrieved from historical memory via $\mathcal{R}_\psi$, outputting a step-level or window-level anomaly score s_t:

$$\begin{aligned} s_t &= \mathcal{P}_\theta\Big(\mathbf{o}_t, \mathcal{R}_\psi(\mathcal{M}_t, \mathbf{o}_t, Q), Q\Big) \\ &\equiv f_\theta(\mathbf{o}_t, \mathcal{M}_t). \end{aligned} \tag{5}$$

Imputation. Let Ω represent the indices of observed entries and Ω^c denote missing values. The predictor reconstructs $\hat{\mathbf{x}}_{\Omega^c}$ by leveraging memory representations retrieved conditionally on the observed subset $\mathbf{x}_\Omega$:

$$\begin{aligned} \hat{\mathbf{x}}_{\Omega^c} &= \mathcal{P}_\theta\Big(\mathbf{x}_\Omega, \mathcal{R}_\psi(\mathcal{M}_\Omega, \mathbf{x}_\Omega, Q), Q\Big) \\ &\equiv f_\theta(\mathbf{x}_\Omega, \mathcal{M}_\Omega). \end{aligned} \tag{6}$$

Reasoning. Given a series observation $\mathbf{o}_t$ and a query Q, the model retrieves relevant information from memory and uses it to derive an inference. Depending on the setting, the output may be a textual answer, explanation, reasoning trace, or intermediate inference:

$$\begin{aligned} \mathbf{y} &= \mathcal{P}_\theta\Big(\mathbf{o}_t, \mathcal{R}\psi(\mathcal{M}_t, \mathbf{o}_t, Q), Q\Big) \\ &\equiv f_\theta(\mathbf{o}_t, \mathcal{M}_t), \end{aligned} \tag{7}$$

where $\mathbf{y}$ denotes the task-dependent reasoning output. Unlike forecasting or classification, the objective is not necessarily to predict a numerical value or class, but to interpret the observed series and produce an inference supported by the available evidence.

Decision-Making. Given an observation $\mathbf{o}_t$, the model retrieves relevant historical information and uses it to select an action a_t. The action may subsequently alter the environment and generate new observations, actions, and outcomes that can be incorporated into memory:

$$\begin{aligned} a_t &= \pi_\theta\Big(\mathbf{o}_t, \mathcal{R}\psi(\mathcal{M}_t, \mathbf{o}_t, Q), Q\Big) \\ &\equiv \pi_\theta(\mathbf{o}_t, \mathcal{M}_t), \end{aligned} \tag{8}$$

where a_t denotes the selected action. In contrast to reasoning, whose output is an inference or reasoning artifact, decision-making produces an action that directly determines the agent's subsequent interaction.

Crucially, these distinct objective functions do not necessitate architecturally separate memory mechanisms. Instead, they demonstrate the versatility of a unified memory substrate in supporting disparate downstream computations. We examine how different memory mechanisms instantiate this common formulation across tasks in Section 9.

2.4 Four Axes of Memory

We characterize memory mechanisms along four axes: representation, capacity, persistence, and access. These properties describe the memory state and its lifecycle rather than the particular downstream task.

Representation specifies the form and granularity of retained information. Depending on the mechanism, memory may be represented as:

$$\begin{aligned} &\mathcal{M}_t \in \mathbb{R}^{d_h}, \quad \text{or} \\ &\mathcal{M}_t \in \mathbb{R}^{N_t \times d_m}, \quad \text{or} \\ &\mathcal{M}_t = \{\mathbf{x}_i\}_{i \in \mathcal{I}_t} \end{aligned} \tag{9}$$

Here, $\mathbb{R}^{d_h}$ denotes an implicit fixed-dimensional state, $\mathbb{R}^{N_t \times d_m}$ denotes a collection of explicit vector entries, and $\{\mathbf{x}_i\}_{i \in \mathcal{I}_t}$ denotes retained observations or other raw inputs.

Capacity specifies how retained information scales with the available data horizon. Let N_t denote the number of memory entries retained at time t. We categorize this scaling dynamic into three distinct regimes:

(1) Architecturally Bounded Capacity imposes a strict, static upper limit on the memory footprint:

$$N_t \leq N_{\max}, \tag{10}$$

where $N_{\max}$ is independent of the amount of observed data. Once this limit is reached, additional historical information must be compressed, overwritten, or otherwise discarded.

(2) Data-Scaled Capacity can increase as more observations become available, without a fixed architectural upper limit:

$$N_t \to \infty \quad \text{as} \quad t \to \infty. \tag{11}$$

New observations can therefore be added to memory as the data horizon grows, allowing the memory footprint to expand with the amount of accumulated information. The capacity is consequently determined by the data retained rather than by a fixed architectural limit.

(3) Policy-Bounded Capacity can grow with incoming data in principle, but its effective size is constrained by an explicit retention policy:

$$N_t \leq N_{\text{policy}}(t), \tag{12}$$

where $N_{\text{policy}}(t)$ is determined by factors such as eviction rules, storage budgets, or computational constraints. Unlike architecturally bounded memory, the limit is not inherent to the model architecture; instead, the system decides how much information to retain as memory grows.

Collectively, the three regimes provide an additional memory perspective for understanding the interplay between structural limits, data accumulation, and active state management.

Persistence describes how long retained information remains available. For an entry m written at time t_{write} and removed at t_{discard}, its lifetime is

$$\tau(m) = t_{\text{discard}} - t_{\text{write}}. \tag{13}$$

We distinguish:

(1) Per-sequence persistence: memory is reset when a new sequence or interaction begins.

(2) Per-dataset persistence: memory is constructed from a dataset and remains fixed during deployment.

(3) Online persistence: memory continues to evolve across deployment or interaction steps.

Capacity and persistence are independent. A memory can have bounded capacity but persist throughout deployment, or have data-scaled capacity while aggressively evicting old entries.

Access specifies how information is written to and read from memory. Writing may involve recurrent state updates, gated updates, appending new entries, overwriting existing entries, or controller-selected operations. Reading may use dense attention, similarity search, nearest-neighbor retrieval, key-value lookup, or a learned controller. The access axis captures not only whether memory can be read, but also how selectively information is written, addressed, updated, and forgotten.

Remark 1 These axes provide complementary dimensions for characterizing memory mechanisms. The taxonomy in Section 4 builds on them to organize time-series memory approaches by memory architecture design, studying memory approaches that range from implicit, fixed-capacity state representations to explicit stores with data-scaled or policy-bounded capacity and increasingly active mechanisms for writing, retrieval, and consolidation.

2.5 Internal versus External Memory

Equipped with these axes, we can broadly divide time-series memory into two dominant paradigms. *Internal memory* resides in fixed-size latent states, such as recurrent cell states, where the capacity is architecturally bounded at design time, and the representation is strictly implicit. *External memory*—categorized in this survey into explicit memory banks, retrieval memory indices, and agentic memory systems—is defined by its discrete representation and distinct access mechanisms. While explicit memory relies on a fixed, architecturally bounded capacity, retrieval and agentic mechanisms expand this to data-scaled and policy-bounded capacities, allowing the memory footprint to grow dynamically. This decoupling of storage from model parameters allows external stores to incorporate information never seen in training without changing a single weight. This survey centers on these external memory paradigms, treating internal memory primarily as a point of contrast, though the boundary is increasingly permeable in modern hybrid architectures.

Taken together, these axes and paradigms provide a common language for describing how memory is represented, scaled, and accessed across time-series models. However, characterizing memory mechanisms alone does not explain why different forms of memory are needed in the first place. We therefore first examine the sources of memory demand in time-series problems.

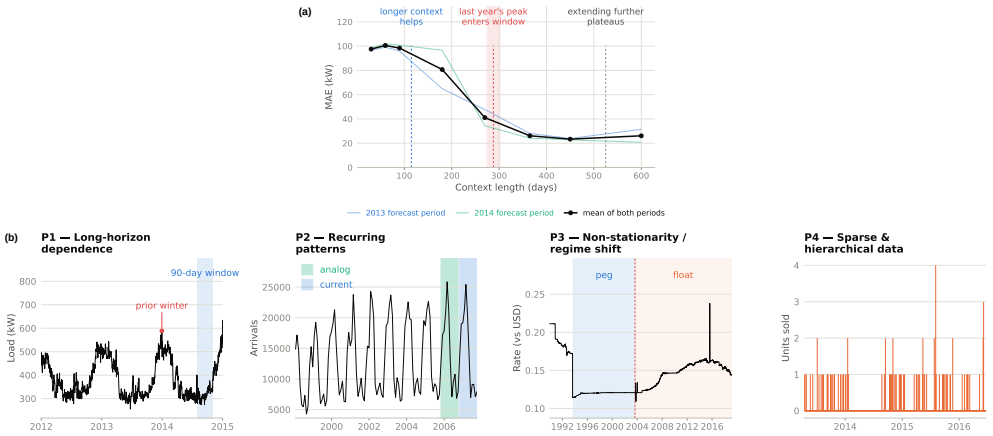


Fig. 2 Why forecasting needs memory. (a) Zero-shot Chronos (Ansari et al. 2024) error when forecasting a real winter peak, as input context length grows (mean of two forecast periods; thin lines show each). Error stays flat below ∼90 days. It drops sharply once the context reaches the previous winter's peak (∼275–302 days back), and it plateaus beyond 450 days. The task needs long-range history, but a fixed-size window alone cannot fully exploit it. **(b)** Four real-world series motivating the problems in Section 3. **P1:** winter-peaking electricity demand (*Electricity*), where a 90-day autumn window misses the previous winter's peak. **P2:** monthly tourist arrivals (*Tourism*) with a strong annual cycle; the current window closely matches the same period one year earlier. **P3:** an exchange rate (*Exchange-Rate*) moving from a long peg to a floating regime, which makes old peg-era episodes misleading. **P4:** an *M5* product-store sales series with 93% zero-sales days, which needs information from related series or higher aggregation levels.

3 Why Time Series Need Memory

Before studying types of memory, we first need to understand why time-series models rely on them. Time-series models need memory when useful predictive information lies outside the immediate input window or is difficult to recover from a long history. Standard lookback windows work well when the past is recent and compact, but they fall short when key signals are distant, scattered across other series, or observed once and needed much later. Returning to the winter electricity example from Section 1, forecasting a winter demand peak depends on operational data from the previous winter, which is long gone from a standard 90-day context window. As a result, the performance of forecasting models stalls until the context window stretches far enough, yet extending it further yields diminishing returns, demonstrating the need for better memory (see Figure 2(a)). Similarly, an anomaly detector deployed on a newly commissioned machine has no historical baseline of its own and must rely on logs from older equipment. In both scenarios, the model cannot simply look at its recent history; it must store, retrieve, or maintain past information over long horizons.

We group these challenges into four primary problem settings (see Figure 2(b)), which define the structure of our taxonomy in Section 4 and guide our analysis of specific mechanisms in Section 6–Section 8. These settings also highlight a fundamental architectural tradeoff: whether to compress historical context into a fixed, *internal*

state (such as updated weights or hidden vectors) or to store it in an *external* memory that the model queries on demand.

3.1 P1: Long-horizon dependence

Time series frequently depend on events far back in their history. Unlike language, where context dependencies appear at arbitrary token distances, time-series dependencies are often periodic, driven by daily, weekly, or seasonal cycles. A retail forecaster, for example, relies on values recorded at exact annual offsets. Other exogenous drivers, such as macroeconomic shifts, climate trends, and regime changes, operate at slower frequencies and continuously influence the target variable over spans far exceeding practical window sizes.

Expanding the history window L to capture these signals quickly becomes impractical. Standard self-attention scales quadratically with L, whereas linear models like DLinear (Zeng et al. 2023) require larger input layers, diluting rare, informative points among irrelevant ones. Recurrent architectures and state-space models address this by compressing past information into a fixed hidden state. However, this internal compression is inherently lossy and often loses faint, distant signals. External memory architectures such as MTNet (Chang et al. 2018), which stores historical blocks and retrieves relevant segments via attention, offer an alternative by maintaining distant context without continuous compression.

3.2 P2: Recurring Patterns

Many time series exhibit recurring behavior. When current conditions resemble a past event, the outcome of that past event provides strong predictive value. An unexpected demand spike is easier to forecast if the model can reference a similar spike from the previous year. Conversely, reconstruction-based anomaly detectors identify outliers by verifying that current inputs do not resemble any historical normal baseline (Gong et al. 2019).

Internal state compression is ill-suited for this task because it blends past episodes, destroying the distinct local dynamics needed for direct matching. Explicit, external memory solves this by storing past states as addressable entries. Models can either store raw historical windows to ground forecasts in historical precedents or maintain a bank of learned memory prototypes to evaluate incoming data. Beyond performance gains, explicit retrieval makes models far easier to audit, as predictions can be traced back to concrete historical examples.

3.3 P3: Non-stationarity and regime shift

Time series often undergo distribution shifts due to evolving market dynamics, sensor drift, or changing consumer habits. When the underlying data-generating process changes, historical data can become uninformative or actively misleading.

This creates a tension for memory design. On one hand, dynamic external stores allow models to adapt to distribution shifts without full retraining; new data points can be written to memory while outdated entries are removed. Fixed-state memory,

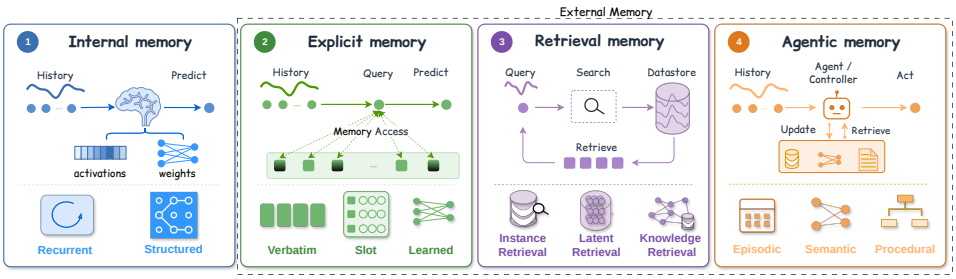


Fig. 3 Taxonomy of memory architectures for time-series models. (1) Internal memory retains historical information in dynamic activations, with learned parameters governing how the state is updated. It includes recurrent and structured state-space mechanisms. External memory includes: (2) explicit memory maintains dedicated addressable entries, (3) retrieval memory accesses an external collection, and (4) agentic memory actively manages what is stored and retrieved. The figure also shows the main subclasses within each memory architecture.

by contrast, remains static after training. On the other hand, standard similarity-based retrieval assumes that past patterns remain valid indicators of future behavior. During a regime shift, retrieving a closely matching historical episode can lead to confident, yet erroneous predictions.

Handling non-stationary data requires active management policies rather than passive storage. Models must explicitly decide when to overwrite information, drop stale records, or ignore strong matches that belong to outdated regimes. This is a core motivation for the agentic memory write and update strategies detailed in Section 8.

3.4 P4: Sparse and hierarchical data

In many real-world applications, individual series lack sufficient local history to train a reliable model. For example, a newly launched product has no sales history, a newly deployed sensor lacks past baselines, and rare events offer few instances to learn from. In these cases, models must refer to related series or higher levels of aggregation. A new product can inherit trends from established items in the same category, while a regional sensor can leverage cross-series patterns across a spatial network. Unlike P1, which addresses temporal dependencies within a single series, P4 addresses spatial and structural dependencies across multiple series and hierarchical levels.

Memory mechanisms address data sparsity by maintaining a shared repository of global features across series. Frameworks like memory-augmented graph networks and hierarchical memory modules store cross-series patterns (Liu et al. 2022), allowing data-scarce nodes to pull context from the broader system. Shared external memory is naturally suited to this task, as a single series' internal state cannot easily transfer learned representations across distinct streams.

3.5 Concerns shared across problems

Two additional practical challenges influence memory selection across all four problem settings:

- **Exogenous covariates:** Target series are frequently governed by exogenous variables such as news events, weather conditions, or macroeconomic indicators that lie outside the target time series itself. Memory architectures must be designed to store and index these external covariates alongside target histories.
- **Auditability:** In safety-critical domains like finance, healthcare, and industrial control, model predictions must be verifiable. External memory systems provide an explicit audit trail by identifying the exact past episodes or prototypes that triggered an output.

4 Taxonomy of Time-series Memory

The problems in Section 3 necessitate memory mechanisms and architectures that preserve useful information from the past in an effective form for prediction. How this information is preserved varies substantially across time-series models. Some models compress the observed history into an internal state, while others keep individual observations or learned patterns explicitly available. External retrieval further allows models to draw on historical information, while more recent approaches actively manage what information should be stored, updated, or discarded.

We organize these approaches into four broad classes: *internal memory*, *explicit memory*, *retrieval memory*, and *agentic memory*. The difference is mainly architectural, based on where historical information is stored and how the model interacts with it. Figure 3 summarizes the four classes and their main subclasses. In the previous section, the problems describe *why* memory is needed, while here the taxonomy describes *how* that memory is implemented.

4.1 Four memory architectures

Internal memory (Section 5). Internal memory retains past information within the model rather than in a separate memory store. We review two common forms. *Recurrent-state memory*, as used in RNNs, LSTMs, and GRUs (Elman 1990; Hochreiter and Schmidhuber 1997; Cho et al. 2014), carries a hidden state forward and updates it with each new observation. *Structured-state memory*, including state-space models (Gu and Dao 2023), summarizes a longer history into a structured, fixed-size representation that preserves useful temporal information over long sequences. In both cases, prediction depends on an internal state that has accumulated information from earlier observations.

The advantage is simplicity since memory is part of the model computation and requires no separate storage or retrieval step. However, historical information must be compressed into a bounded representation, so distant or infrequent events may be weakened or lost. This makes internal memory a useful baseline for P1, but less suitable when particular historical episodes must remain individually accessible.

Explicit memory (Section 6). Explicit memory is the simplest form of external memory. It preserves information as individually addressable entries rather than compressing the entire history into a single state. We distinguish three main mechanisms. *Verbatim memory* keeps past observations directly accessible, as in Transformer

attention (Vaswani et al. 2017). *Slot memory* uses dedicated memory locations that store selected states or representations (Graves et al. 2016; Le et al. 2018). *Learned memory* stores compact patterns such as prototypes or shared representations. In time-series models, these memories make past observations or their representations explicitly addressable, rather than requiring historical information to be compressed into a hidden state. Their main limitation is thus capacity, which is tied to the context length, number of slots, or size of the learned memory.

Retrieval memory (Section 7). Retrieval memory stores a larger collection outside the model and searches it when needed. Unlike explicit memory, the retrieved information can come from long historical archives, other series, or external knowledge. Retrieval memory has three main subclasses. *Instance retrieval memory* stores concrete cases such as past windows or trajectories (Khandelwal et al. 2020). *Latent retrieval memory* stores compact representations such as embeddings. *Knowledge retrieval memory* stores external information such as text, events, or structured domain knowledge (Lewis et al. 2020; Guu et al. 2020). Their main advantage is scale and cross-series access, while the central challenge is retrieving information that is truly relevant rather than merely similar.

Agentic memory (Section 8). Agentic memory allows an agent to store and update information from past interactions. We categorize three main forms based on what is stored. *Episodic memory* records specific past events, such as previous forecasting episodes and their outcomes. *Semantic memory* stores more general knowledge learned from past observations, such as recurring regimes or relationships between variables. *Procedural memory* stores knowledge about how to perform a task, such as which time-series tools or retrieval strategies to use in different situations (Shinn et al. 2023). Many agentic architectures combine two or more of these memory types.

Unlike retrieval memory, agentic memory can be changed based on new observations and outcomes. The agent can add new experiences, update its knowledge, or change how it handles similar situations in the future. This makes agentic memory useful when data or tasks change over time, but it also requires the agent to decide what information to keep, update, or remove.

Table 2 Relationship between the four time-series memory problems and memory architectures. ●● dominant, ● established but less prevalent, and (○) emerging or underexplored.

Problem	Internal	Explicit	Retrieval	Agentic
P1–Long-horizon dependence	●●	●	●●	●
P2–Recurring patterns	●	●●	●●	●
P3–Non-stationarity/drift	(○)	(○)	●	●
P4–Sparse/hierarchical data	●	●●	●	(○)

4.2 Connecting problems and memory architectures

The four architectures are not tied one-to-one to the four problems in Section 3. A long-horizon dependency, for example, can be handled by maintaining a recurrent or state-space representation, explicitly storing historical segments, or retrieving the relevant period from an external archive. Table 2 summarizes these relationships. Internal memory is widely used for long-horizon modeling (P1), particularly when historical information can be compressed into a useful latent state. Explicit and retrieval memory are natural choices for recurring patterns (P2), where retaining distinguishable historical examples enables direct matching. For non-stationarity (P3), retrieval allows the available context to change over time, while agentic memory additionally provides mechanisms for removing or updating stale information. Sparse and hierarchical settings (P4) have commonly used internal sharing and explicit cross-series memory to transfer information between related streams.

The matrix should therefore be read as a map of common architectural choices, not a strict assignment of problems to methods. Several architectures can address the same problem, and their suitability depends on the structure of the data and the type of historical information required.

5 Internal Memory

Definition 2 (Internal memory) Internal memory refers to time-series models that encode historical observations through recurrently updated, fixed-dimensional latent states rather than explicitly storing the observed history. This paradigm enables models to incorporate increasingly long temporal histories by continuously compressing past observations into an internal hidden state. The state is updated according to learned dynamics, allowing it to capture temporal regularities, dependencies, and evolving patterns that are relevant to future predictions.

5.1 Mechanism

Internal memory can be viewed as compressing the observed history of a time series into a fixed-size dimensional representation. Given a time series $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$, where $\mathbf{X} \in \mathbb{R}^{T \times d}$ with d denoting the variate number (number of channels) and T the temporal sequence length, let $\mathbf{x}_t \in \mathbb{R}^d$ denote the input observation at time step t. The compression mechanism can be summarized as:

$$\mathbf{h}_t = u_\theta(\mathbf{h}_{t-1}, \mathbf{x}_t) \in \mathbb{R}^{d_h}, \qquad \widehat{\mathbf{y}}_t = f_\theta(\mathbf{o}_t, \mathbf{h}_t), \tag{14}$$

where $\mathbf{h}_t$ is the d_h-dimensional hidden state, u_θ updates it as new observations arrive, and $\mathbf{o}_t$ represents the local observation context available to the model at time t.

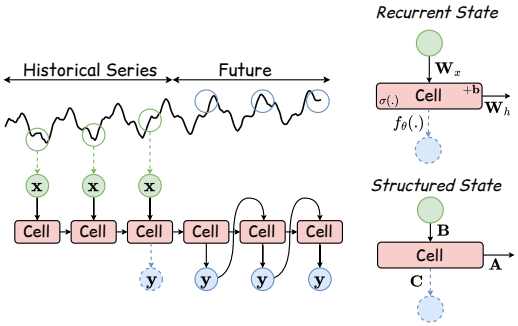


Fig. 4 Illustration of internal memory in time-series forecasting. The figure shows two approaches to compressing historical observations into a fixed-dimensional internal state: recurrent-state memory, where the state is updated through nonlinear recurrent transitions, and structured state memory, where temporal history is encoded through parameterized state-space dynamics.

This is the implicit-memory special case of the generic memory-augmented model in Equation (1): the memory is carried by the hidden state, $\mathcal{M}_t \equiv \mathbf{h}_t$, while memory formation is performed by the recurrent update u_θ. Retrieval is implicit, with $\mathcal{R}_\psi(\mathcal{M}_t, \mathbf{o}_t, Q) \equiv \mathbf{h}_t$, so the predictor evaluates both the local context and internal state via $f_\theta(\mathbf{o}_t, \mathbf{h}_t)$.

Repeated updates make $\mathbf{h}_t$ depend on the entire history, allowing it to retain information beyond the current input window. Its fixed dimension d_h gives internal memory an architecturally bounded capacity (refer to Section 2.4). The compression of history into a fixed-size dimensional representation, however, makes internal memory difficult to inspect, address, or revise at the level of individual observations or past events.

We therefore view internal memory as the starting point of the time-series memory spectrum, where the emphasis is on learning compact summaries of temporal history before moving towards mechanisms that preserve past observations as persistent and addressable information. For readers seeking more details on the recurrent paradigm and its architectural variants in this domain, several dedicated surveys provide comprehensive reviews (Hewamalage et al. 2021; Wen et al. 2023). Figure 4 illustrates internal memory mechanisms through two representative approaches: recurrent-state memory and structured state memory.

5.2 Types of Internal Memory

5.2.1 Recurrent state

RNNs (Elman 1990) established the fundamental formulation of recurrent internal temporal memory by maintaining an evolving hidden state that summarizes previous observations. Unlike models that process each observation independently, recurrent architectures update their internal state according to both the current input and the accumulated historical context, allowing temporal dependencies to be represented through sequential state evolution. In its generic form, a recurrent model instantiates

Equation (14) with local observation context $\mathbf{o}_t = \mathbf{x}_{t-L+1:t}$ through a one-step update,

$$\begin{aligned} \mathbf{h}_t &= \sigma(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t + \mathbf{b}) \,, \\ \widehat{\mathbf{y}}_t &= f_\theta(\mathbf{x}_{t-L+1:t}, \mathbf{h}_t), \end{aligned} \tag{15}$$

where σ is an elementwise nonlinearity, $\mathbf{W}_h \in \mathbb{R}^{d_h \times d_h}$ and $\mathbf{W}_x \in \mathbb{R}^{d_h \times d}$ are learned parameters that govern state transitions and input encoding, respectively, and $\mathbf{b}$ is a bias. The hidden state $\mathbf{h}_t$, rather than these learned parameters, carries information from the observed history.

One of the difficulties is that early recurrent networks can struggle to preserve information over long horizons because repeated state updates caused gradients to vanish or explode. LSTMs (Hochreiter and Schmidhuber 1997) address this limitation through gated memory updates, which control what information is retained, discarded, and exposed at each time step. GRUs (Cho et al. 2014) provide a simpler gated alternative by combining the mechanisms for updating and resetting the recurrent state. These architectures become important building blocks for time series modeling because they provide a natural mechanism for representing dependencies.

A substantial body of time series forecasting work subsequently adapted recurrent memory to better exploit the structure of real-world temporal data. DeepAR (Salinas et al. 2020), for example, uses an autoregressive recurrent state to capture temporal patterns shared across related series. DA-RNN (Qin et al. 2017) enhances recurrent memory with dual-stage attention, allowing its recurrent encoder–decoder states to selectively incorporate informative variables and historical information when generating forecasts. TreNet (Lin et al. 2017) combines convolutional layers with LSTMs, using local temporal features to inform recurrent representations of longer-term dependencies, while LSTNet (Lai et al. 2018) extends this idea with a convolutional-recurrent architecture, a skip-RNN for long-range periodic patterns, and an autoregressive component for highly autocorrelated series.

The same recurrent memory principle extends beyond forecasting. Multivariate LSTM-FCNs (Karim et al. 2019) combine fully convolutional pathways with recurrent blocks for time series classification, demonstrating that the hidden state can encode discriminative temporal patterns. LSTM-based methods have likewise been used for anomaly detection, where the recurrent state provides a compact representation of expected temporal behavior against which deviations can be identified (Malhotra et al. 2015). Encoder–decoder variants further exploit recurrent states to model normal multi-sensor dynamics and detect anomalous sequences (Malhotra et al. 2016). Recurrent memory has also been adapted to incomplete observations. GRU-D (Che et al. 2018) modifies internal cell updates with trainable exponential decay mechanisms for missing intervals, while BRITS (Cao et al. 2018) uses twin bidirectional recurrent states to iteratively impute missing values in multivariate sequences.

In more specialized settings, SAnD (Song et al. 2018) layers self-attention over LSTM-based representations to selectively identify informative variables and temporal patterns in clinical time series, while CloudLSTM (Zhang et al. 2021) extends recurrent modeling to spatiotemporal point-cloud streams by introducing a Dynamic

Point-cloud Convolution operator that extracts local spatial correlations from irregular point coordinates directly within the recurrent cell, demonstrating how recurrent states can encode evolving spatial as well as temporal dependencies. As we can see, combining recurrent memory with complementary paradigms, such as convolutions for local feature extraction, attention for selective routing, or linear components for scale, makes these architectures powerful, proving that the core principle of folding history into a compact hidden state remains versatile.

Interestingly, this recurrent view of memory has not disappeared with the rise of newer sequence architectures. P-sLSTM (Kong et al. 2025) revisits LSTMs in the context of modern long-term forecasting and demonstrates that, with appropriate architectural and training choices, recurrent models can remain highly competitive against more recent methods.

5.2.2 Structured state

State-space models approach the same problem from a different direction. Instead of designing increasingly sophisticated gates for a recurrent state, they ask whether the dynamics of the state itself can provide a more effective compression mechanism. Structured state-space models (SSMs) (Gu et al. 2022) replace explicit gating operations with parameterized state transitions, where a fixed-dimensional hidden state continuously summarizes the input history. A general discrete-time state-space formulation mapping a time series observation $\mathbf{x}_t \in \mathbb{R}^d$ to a state $\mathbf{h}_t \in \mathbb{R}^{d_h}$ can be written as:

$$\mathbf{h}_t = \mathbf{A}\mathbf{h}_{t-1} + \mathbf{B}\mathbf{x}_t, \qquad \hat{\mathbf{y}}_t = \mathbf{C}\mathbf{h}_t, \tag{16}$$

which is a linear special case of Equation (15), obtained by replacing the nonlinear transition σ with the identity map and setting the transition matrix $\mathbf{W}_h = \mathbf{A} \in \mathbb{R}^{d_h \times d_h}$, the input projection $\mathbf{W}_x = \mathbf{B} \in \mathbb{R}^{d_h \times d}$, and $\mathbf{b} = \mathbf{0}$. The readout is likewise specialized to the linear state mapping $\hat{\mathbf{y}}_t = \mathbf{C}\mathbf{h}_t$. Linearity makes the compression map available in closed form,

$$\mathbf{h}_t = \mathbf{A}^t \mathbf{h}_0 + \sum_{k=1}^{t} \mathbf{A}^{t-k} \mathbf{B}\mathbf{x}_k, \tag{17}$$

so that $\mathbf{A}^{t-k}$ determines how the contribution of the observation at time k propagates to the state at time t. This perspective has also been explored specifically for time series forecasting. FiLM (Zhou et al. 2022a) introduces a frequency-enhanced Legendre memory model that combines structured state representations with frequency-domain information, illustrating how the capacity of a compact latent state can be improved by incorporating temporal structure beyond simple sequential recurrence.

More recent state-space models make this compression adaptive to the observed sequence. Mamba (Gu and Dao 2023) introduces selective state spaces with input-dependent state transitions, allowing the model to selectively retain or discard information while maintaining efficient linear-time sequence processing. This idea has quickly been explored in time series forecasting, with TimeMachine (Ahamed and Cheng 2024) investigating Mamba-based components for long-term forecasting and

MambaTS (Cai et al. 2024) adapting selective state-space modelling specifically to long-term temporal dependencies. Architectural extensions have further refined these predictive capabilities; for instance, ms-Mamba (Karadag et al. 2026) utilizes varying sampling rates to capture multi-scale temporal contexts, while Bi-Mamba+ (Liang et al. 2024) incorporates bidirectional processing to selectively integrate features over longer temporal ranges. The utility of selective state compression also extends well beyond forecasting. For time series classification, MambaSL (Jung and Kim 2026) demonstrates that an efficient single-layer Mamba architecture can achieve discriminative performance. Similarly, in anomaly detection, the MAAT (Sellam et al. 2025) integrates Mamba architectures with sparse attention mechanisms to effectively capture both short- and long-term temporal dependencies alongside association discrepancies. SST (Xu et al. 2024) further combines Mamba and Transformer experts across temporal scales, reflecting the broader trend towards combining compressive state representations with mechanisms that can access richer contextual information. Attraos (Hu et al. 2024) further extends structured-state memory with a multi-resolution dynamic memory unit that captures historical dynamical structures across different temporal resolutions. At the same time, empirical analysis such as Wang et al. (2025) suggests that selective state compression is not universally optimal, highlighting the importance of matching the memory mechanism to the structure of the underlying time series.

5.3 Coverage and Limitations

Despite the growing range of mechanisms proposed in the literature, the fundamental limitations of internal memory become clear when evaluated along our four axes. In terms of representation, whether history is accumulated through gated cell states, structured state transitions, or input-selective dynamics, the model ultimately distills the past into an implicit, fixed-dimensional vector. Because this state size is static, the memory's capacity is strictly architecturally bounded (refer to Section 2.4); it cannot grow as the data horizon expands, forcing constant compression and overwriting of old information. Consequently, the persistence of any single observation is highly volatile, typically lasting only until it is diluted by subsequent updates. Finally, the access mechanism is entirely implicit. A model may learn that an important event occurred, yet it cannot directly query or retrieve that event as a discrete stored record, nor selectively revise it after compression. This distinction separates internal memory from the external memory mechanisms considered next.

Remark 2 Internal memory provides an efficient mechanism for temporal modelling by encoding historical information in fixed-dimensional latent states, whose dynamics are governed by learned parameters. Its strength lies in compact representation and computational efficiency; however, its memory is constrained by fixed capacity, limited persistence, and implicit access. These limitations motivate the transition towards external memory mechanisms that support persistent storage, retrieval, and management of historical experiences (Sections 6 to 8).

6 Explicit Memory

Definition 3 (Explicit memory) Explicit memory refers to time-series models that hold information in a store of individually addressable entries (slots, vectors, or records), each written and read as a unit rather than entangled in weights or a dense state. At its core, this paradigm asks: *how can a model keep specific pieces of the past available without folding them into a single shared representation?* The answer is to separate storage from computation: entries are created or revised individually and consulted through content-based addressing, so stored content can be inspected, edited, or expanded without changing the model that reads it.

6.1 Mechanism

Let the store be $\mathcal{M}_t = [\,\mathbf{m}_1, \ldots, \mathbf{m}_{N_t}\,]^\top \in \mathbb{R}^{N_t \times d_m}$ with entry width d_m. For slot and learned stores, N_t is a constant fixed before training. For verbatim stores, it grows as the window fills, but it never exceeds the context length L, and it resets once the model moves to the next sequence. Explicit memory is therefore bounded in both cases, which is different from the retrieval stores of Section 7. An encoder maps the lookback window to a query $\mathbf{q}_t = g_\phi(\mathbf{x}_{t-L+1:t})$, and the *read* resolves it against every entry by normalized similarity,

$$w_{t,i} = \frac{\exp\big(s(\mathbf{q}_t, \mathbf{m}_i)\big)}{\sum_{j=1}^{N_t} \exp\big(s(\mathbf{q}_t, \mathbf{m}_j)\big)}, \quad \mathbf{r}_t = \sum_{i=1}^{N_t} w_{t,i}\, \mathbf{m}_i, \tag{18}$$

with $s(\cdot,\cdot)$ typically cosine similarity or a scaled dot product. Equation (18) operates like an attention mechanism, sharpening the weights toward the most similar memory contents. The prediction conditions on the window to produce the readout, $\hat{\mathbf{y}}_t = \mathcal{P}_\theta(\mathbf{x}_{t-L+1:t}, \mathbf{r}_t)$, where $\hat{\mathbf{y}}_t$ is the task target. The *write* takes one of two forms. Verbatim stores append, encoding a new element as a fresh entry and leaving the rest untouched, while slot stores revise existing entries across time steps:

$$\mathbf{m}_i \leftarrow \mathbf{m}_i + \boldsymbol{\psi}_i \odot \Delta(\mathbf{m}_i, \{\mathbf{q}_t\}), \tag{19}$$

where Δ produces the candidate update for entry i, usually an aggregate of the queries that addressed it weighted by $w_{t,i}$, and the gate $\boldsymbol{\psi}_i$ controls how much of that update is admitted; controller-issued erase and add operations are the special case in which the model emits Δ and $\boldsymbol{\psi}_i$ itself. Explicit memory specializes Equation (1) by pairing append or gated write operations ($\mathcal{W}_\phi$, Equation (19)) with content-addressed read mechanisms ($\mathcal{R}_\psi$, Equation (18)). Unlike internal memory's identity read (Section 5), $\mathcal{R}_\psi$ performs selective retrieval over individually addressable entries $\mathbf{m}_i$, establishing its explicit structure. Persistence, the last axis of Section 2.4, refers to how long the

memory store survives. For example, a verbatim store holds only the entries built from the current input window. When the model moves on to the next sequence, those entries are discarded, and nothing carries over, so this memory is per-sequence.

Memory capacity is designed differently according to the type of storage. In slot and learned stores, while the model parameters governing the write rules are fixed after training, the contents of the slots $\mathbf{m}_i$ act as a dynamically updated latent workspace that evolves via runtime write operations during inference. At inference, the read can therefore only return mixtures of patterns that training placed in the store. In verbatim, the entries are data, not parameters. Writing for these stores involves no changes in parameters: the trained encoder maps the new observation to a vector, and that vector is appended to the store as a fresh entry. Because encoding and appending are forward-pass operations, they apply to any input the model receives, including one it has never seen before. The store can thus hold new content, though how faithfully that content is represented depends on the encoder, which remains fixed. The read weights $w_{t,i}$ record how strongly the current window draws on each entry, which is what makes the read traceable, partially addressing the auditability problem raised in Section 3. The methods in this family share the read of Equation (18); they differ in what $\mathbf{m}_i$ contains and in how the write of Equation (19) is applied (see Figure 5(a) for an illustration).

6.2 Types of Explicit Memory

6.2.1 Verbatim Memory

Verbatim memory gives every observed timestep its own entry in the store. Entries are encoded on arrival, and the encoding is applied elementwise: no entry is merged with another or summarized into a coarser state, and forgetting occurs only when an entry is truncated from the store. A query is therefore scored against each retained element individually rather than against a learned summary of the past. The limiting case is the attention context of a Transformer (Vaswani et al. 2017), where the store is the input window itself, and every position remains addressable until the window advances past it.

Transformer-based forecasters instantiate this design directly. Within this literature, the word memory usually refers to verbatim retention. For example, LogTrans is introduced as breaking the memory bottleneck of self-attention on long series (Li et al. 2019), and the sparse or restructured reads of Informer (Zhou et al. 2021), Autoformer (Wu et al. 2021), and FEDformer (Zhou et al. 2022b) answer the same quadratic burden, thinning how the store is read while leaving its per-position entries intact. PatchTST (Nie et al. 2023) coarsens the unit of storage from a timestep to a patch, shortening the store without consolidating across entries, and this patch-level verbatim store has become the default in time-series foundation models: TimesFM (Das et al. 2024) and Timer (Liu et al. 2024) hold one key-value entry per patch in a decoder-only cache, Chronos (Ansari et al. 2024) quantizes values into a discrete vocabulary whose tokens are attended over individually, and Moirai (Woo et al. 2024) and MOMENT (Goswami et al. 2024) flatten patch sequences inside masked encoders. iTransformer (Liu et al. 2024) marks the boundary of the category:

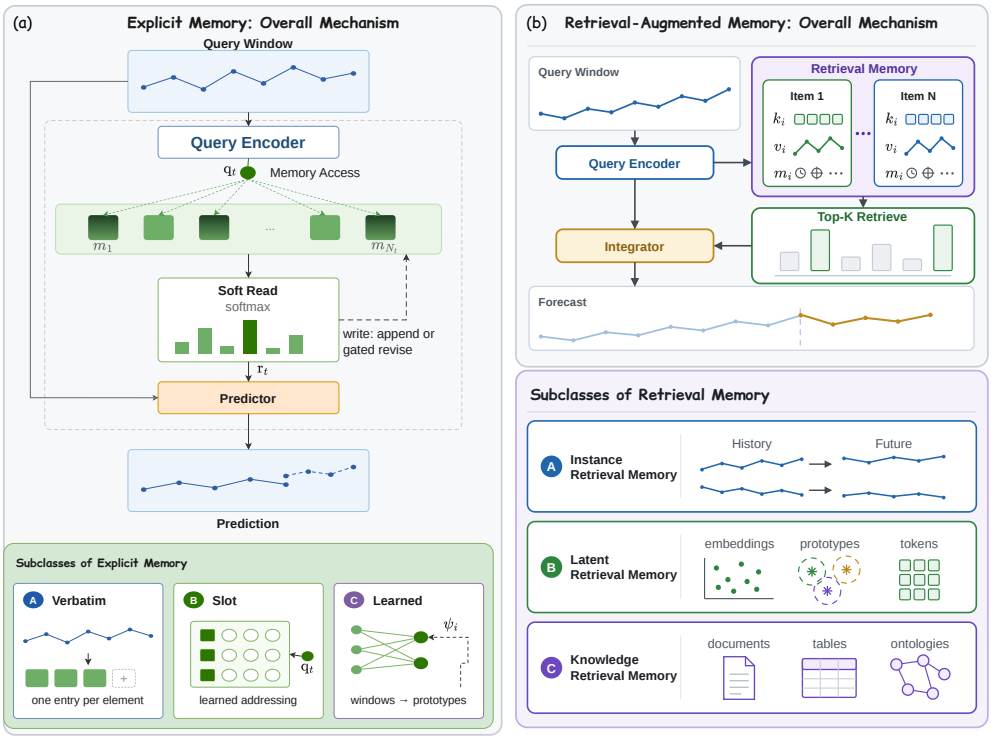


Fig. 5 **(a)** Explicit memory: A query window is encoded as $\mathbf{q}_t$ and read against a store of individually addressable entries $\mathbf{m}_i$. The read operation weights entries by similarity, and the retrieved mixture $\mathbf{r}_t$ is integrated with the query window to produce the prediction. Writing takes one of two forms: appending a new entry, or revising existing entries through the gate ψ_i. The bottom section illustrates the three explicit-memory forms: verbatim, slot, and learned explicit memory. **(b)** Retrieval-augmented memory: A query window is encoded as $\mathbf{q}_t$ and matched against an external memory store whose entries contain a key $\mathbf{k}_i$, a value $\mathbf{v}_i$, and metadata $\mathbf{m}_i$. For the forecasting task, the top-K relevant entries are retrieved and integrated with the query to produce the forecast. The bottom section illustrates the three retrieval-memory forms: instance, latent, and knowledge retrieval memory.

it embeds the full history of each variate into a single token, so its store is verbatim across variates but consolidated over time.

Models that couple time series with language keep the same memory and extend what it already holds. Time-LLM (Jin et al. 2024) reprograms patch embeddings into the token space of a frozen language model, while ChatTS (Xie et al. 2025) and ITFormer (Wang et al. 2025) interleave encoded time-series tokens with text so that a single attention context serves as a joint store over both modalities. Verbatim memory keeps every element separately addressable, but this has a trade-off: the store grows linearly with the context length L, and attention over it grows quadratically. Because these models rely on a finite transformer context, their capacity is inherently architecturally bounded (see Section 2.4) limiting how far back in time they can look.

As a result, much of this literature focuses on making the read operation cheaper, while leaving the contents of the store unchanged.

6.2.2 Slot Memory

Slot memory uses a fixed set of locations that hold model states rather than observations. The model learns how to read from and write to them. The formulation descends from the Neural Turing Machine (NTM) (Graves et al. 2014) and the Differentiable Neural Computer (DNC) (Graves et al. 2016), which pair a controller with an addressable matrix and learn the addressing itself. It also descends from the Variational Memory Encoder–Decoder (Le et al. 2018), which couples the store with a latent variable so that reads inject stochastic structure rather than a single deterministic vector. For temporal data, the design focuses not only on the addressing machinery, but also on what a memory slot is made to hold. PM-MemNet (Lee et al. 2022) recasts forecasting as pattern matching: representative traffic patterns are clustered offline and stored as memory keys, and a query built from the input window retrieves the values that condition the prediction, allowing the model to respond to abrupt regime changes that a purely autoregressive encoder tends to smooth over. MegaCRN (Jiang et al. 2023) instead places a meta-node bank inside a graph convolutional recurrent encoder-decoder and uses the retrieved slots to generate the graph itself, so the memory parameterizes spatial structure rather than merely augmenting the hidden state. In anomaly detection, the mechanism is reversed: MemAE (Gong et al. 2019) restricts the decoder to a sparse combination of memorized normal prototypes, so that the reconstruction error rises precisely for inputs that no slot can express. All of these designs fix the number of slots before training.

One of the central design questions in slot memory is about what to store to ensure reliable retrieval. The clearest example in time series is STanHop (Wu et al. 2024). It builds a forecaster from modern Hopfield layers, a content-addressable associative memory whose retrieval dynamics converge to a stored pattern, with optional external memory modules attachable at each resolution. The accompanying sparse Hopfield extension gives a tighter retrieval-error bound at no cost in capacity. STanHop (Wu et al. 2024) is notable among the methods we review for explicitly evaluating store capacity and the reliability of pattern recovery.

Write operations are less studied than reads in memory-based time-series models. Controller-issued, location-addressed updates in the NTM style have no counterpart on standard time-series benchmarks in the work we surveyed. Related works like STanHop populates its stores in advance, and the remaining methods collapse the write into the fixed or gated content-addressed update of Equation (19). The nearest exceptions sit at the boundary of the field. Titans (Behrouz et al. 2025) learns its write rule at test time and reports gains on forecasting benchmarks, but its store is a parametric fast-weight memory rather than an addressable slot matrix. In short, time-series models have learned how to read from slot memory. They decide where to look, and how much of what they find to change. However, they barely learned to write, lacking mechanisms to tell them what is worth storing, or where to put it.

6.2.3 Learned Memory

This memory stores compact representations learned from data, such as prototypes, centroids, or shared pattern banks. It is by far the dominant form in time series, and its logic inverts verbatim memory. Rather than keeping many concrete episodes and searching among them, the model keeps a few entries that summarize recurring structure, and explains a query as a mixture of them.

MEMTO (Song et al. 2023) is the canonical time-series instance of this adaptation. It retains the reconstruction framing and the prototype bank but makes the update itself learned: a gate determines how intensively each entry absorbs new information, and k-means initialization stabilizes a two-phase procedure where entries are revised incrementally. Detection uses deviation in both input and latent space rather than reconstruction error alone. Recent work carries the gated bank into stronger backbones, with MOMEMTO (Yoon et al. 2025) attaching it to a patch-based foundation model and MemMambaAD (Li et al. 2025) pairing it with a state-space encoder.

A similar prototype-based logic applies to classification, in which the stored items serve as representatives of each class. TapNet (Zhang et al. 2020) learns an embedding together with a set of class prototypes and classifies by attentional distance. The bank is therefore the classifier, and a prediction is traceable to the representative it matched. Shapelet methods reach a similar structure by storing discriminative subsequences (Li et al. 2021; Tang et al. 2020), and memory shapelet learning (Wan et al. 2024) maintains the store online for early classification of streaming series.

Forecasting relies on learned memory mostly when several series share the same structure. MegaCRN (Jiang et al. 2023) attaches a Meta-Node Bank to a graph convolutional recurrent encoder-decoder. Node and time representations query the bank, and the retrieved entries generate the graph used for propagation. The bank therefore acts as an index of operating regimes rather than of episodes. Its reported disentanglement of road links and time slots with different behaviour is the clearest evidence that a bounded learned store can carry some non-stationary structure. MAGL (Liu et al. 2022) records global historical features to mine spatial correlations, and HAMN (Lee et al. 2025) organizes the store by aggregation level, allowing sparse series to draw on their aggregates.

A more recent line of work uses learned memory to adapt frozen foundation models, and is notable because it argues explicitly against retrieval approaches in Section 7. TS-Memory (Lyu et al. 2026) distills an offline nearest-neighbour teacher into a lightweight memory adapter, fused with the backbone at constant overhead so that no datastore is searched. MEMTS (Yu et al. 2026) reaches a similar design from the same motivation. Overall, a bounded memory method is often chosen over an unbounded one not because it holds more, but because of its constant building cost.

Recent papers on memory also focus on imputation. MMNet (Miao et al. 2025) pairs a missing-aware embedding with a memory-enhanced encoder, so a partially observed window is reconstructed from global similarity across the dataset rather than local context alone, and PRIME (Yu et al. 2025) maintains a prototype memory of inter-series structure for irregularly sampled clinical series.

6.3 Memory Writing and Integration

The *read* operator reduces to Equation (18) in almost all of these methods. Yet there are major differences in how each method *writes* with Δ and $\boldsymbol{\psi}_i$ in Equation (19), and how the read result reaches the output. The most common approach is degenerate, in that the write is gradient update at training:

$$\begin{aligned} \Delta(\mathbf{m}_i, \{\mathbf{q}_t\}) &= -\nabla_{\mathbf{m}_i} \mathcal{L}, \\ \boldsymbol{\psi}_i &= \begin{cases} \eta\, \mathbf{1} & \text{at training} \\ \mathbf{0} & \text{at inference} \end{cases} \end{aligned} \tag{20}$$

Here, memory entries are learned by gradient descent on the task loss $\mathcal{L}$ with learning rate η, then frozen. MemAE (Gong et al. 2019), TapNet (Zhang et al. 2020), PM-MemNet (Lee et al. 2022), MegaCRN (Jiang et al. 2023), STanHop's pre-populated modules (Wu et al. 2024), and the foundation-model adapters (Lyu et al. 2026; Yu et al. 2026) all write this way. MNAD (Park et al. 2020) makes the write input-dependent. Let $\mathcal{U}_i$ be the time steps that addressed entry i. Then:

$$\Delta(\mathbf{m}_i, \{\mathbf{q}_t\}) = \sum_{t \in \mathcal{U}_i} w_{t,i}\, \mathbf{q}_t, \qquad \boldsymbol{\psi}_i = \mathbf{1}. \tag{21}$$

MEMTO (Song et al. 2023) is the design in which the gate is learned. Let $\tilde{\mathbf{q}}_i$ be the aggregated queries addressing entry i, its update reads:

$$\begin{aligned} \mathbf{m}_i &\leftarrow (\mathbf{1} - \boldsymbol{\psi}_i) \odot \mathbf{m}_i + \boldsymbol{\psi}_i \odot \tilde{\mathbf{q}}_i, \\ \boldsymbol{\psi}_i &= \sigma\big(\mathbf{W}[\,\mathbf{m}_i; \tilde{\mathbf{q}}_i\,]\big), \end{aligned} \tag{22}$$

which is Equation (19) with $\Delta = \tilde{\mathbf{q}}_i - \mathbf{m}_i$. Here, σ is the logistic function and $\mathbf{W}$ a learned projection, so each entry learns how much of the present it admits. MOMEMTO (Yoon et al. 2025) and MemMambaAD (Li et al. 2025) inherit this gate under stronger backbones, and online shapelet maintenance keeps the write active during deployment (Wan et al. 2024).

The retrieved mixture then either *replaces* the encoding before decoding, which is what forces reconstruction through stored content in the MemAE line, is *fused* with the prediction as in MEMTO (Song et al. 2023) and TS-Memory (Lyu et al. 2026), or *parameterizes* another component as when MegaCRN generates a graph. Replacing is the strongest option among the three. The decoder can only use what is in the store, so a poor reconstruction means no entry matched the input. This mechanism makes the error a useful anomaly score.

Two failure modes are specific to this class. *Slot collapse* occurs when entries converge on similar content, leaving an effective capacity far below N; the compactness and separateness losses of MNAD (Park et al. 2020) and the clustering-based initialization of MEMTO (Song et al. 2023) aim to address this problem. *Addressing diffusion* occurs when the weights $w_{t,i}$ spread across many entries, so that arbitrary inputs can be reconstructed from a mixture. Sparse addressing is a natural solution,

and the sparse Hopfield analysis of STanHop characterizes its effect on retrieval error. Both modes indicate that in explicit memory, the diversity and sparsity of the store are part of the objective, not emergent properties of training.

6.4 Coverage and Limitations

Explicit memory is concentrated in anomaly detection and classification. Anomaly detection is its strongest setting, because a bank of normality yields a detection criterion directly rather than as a by-product. Classification is the second, where prototype and shapelet stores align with matching a series against class representatives. Forecasting adoption is narrower than for other memory types, particularly retrieval memory. In addition, it is mostly cross-series or hierarchical, and imputation has only recently acquired dedicated instances.

The strengths and limitations of explicit memory follow directly from its position along our four memory axes. Because its capacity is architecturally bounded, the memory cost is known in advance, with no index to maintain and no latency that grows with the corpus. However, this means the size N must be fixed before it is known how many distinct regimes exist in the data. Its discrete representation ($\mathcal{M}_t \in \mathbb{R}^{N \times d_m}$) allows for interpretable read access: the memory content can be explicitly audited, and the addressing weights clearly account for each prediction. Yet, write access is notoriously unstable; joint training of reads and writes typically requires clustering initialization and diversity losses to prevent collapse. Furthermore, because persistence is typically per-dataset, memory entries lack temporal context; a prototype cannot be dated or deliberately retrieved when past conditions return. Second, unless the memory is specifically designed to update online, the store remains static after deployment and cannot learn patterns it never saw during training.

Remark 3 Explicit memory makes stored content addressable and auditable at a cost fixed by design, and its central design variable is not the read but the write, that is, whether and how selectively entries are revised. Its constraint is that capacity and content are both decided before deployment, so the store can index the regimes seen in training but cannot grow to accommodate new ones. Removing that constraint requires a store populated from data that expands independently of the model, which is the retrieval memory of Section 7.

7 Retrieval-Augmented Memory

Definition 4 (Retrieval-augmented memory) Retrieval-augmented memory refers to time-series systems that retain information in an external store and selectively recall a query-dependent subset to condition the current prediction. At its core, this paradigm asks: *which information from a potentially large external memory should be recalled for the current query?* Rather than compressing all useful history into a bounded internal representation, retrieval memory preserves individually retrievable records whose

capacity can grow independently of the predictor. Unlike agentic memory, however, the defining operation is selective recall rather than active management of what should be written, revised, or forgotten over time.

This shifts the memory problem from compressing all useful history into a bounded representation to determining what information to preserve externally and which records to recall for a particular query. An important design choice is therefore not only *how* memory is retrieved, but *what* is retained as memory. Time-series systems may retain concrete historical cases, compact latent representations, or external knowledge absent from the numerical series. We use this distinction as the primary organization of this section, categorizing retrieval-augmented memory into *instance retrieval memory*, *latent retrieval memory*, and *knowledge retrieval memory*; these categories primarily describe memory *representation*, while capacity, read/write, and persistence provide complementary characteristics. Figure 5(b) summarizes this perspective. At a high level, a query window is encoded into a query representation $\mathbf{q}_t$ and matched against an external store containing keys $\mathbf{k}_i$, values $\mathbf{v}_i$, and optional metadata $\mathbf{m}_i$. The top-K relevant entries are retrieved, and their values are provided to the predictor along with the query.

Let the external store be

$$\mathcal{M} = \{(\mathbf{k}_i, \mathbf{v}_i, \mathbf{m}_i)\}_{i=1}^{N}, \tag{23}$$

where $\mathbf{k}_i$ is an indexable key, $\mathbf{v}_i$ is the stored memory content, and $\mathbf{m}_i$ denotes optional metadata such as timestamp, series identity, domain, sampling rate, or regime. For a query window $\mathbf{x}_{t-L+1:t}$, a query encoder produces $\mathbf{q}_t = g_\phi(\mathbf{x}_{t-L+1:t})$ and retrieves

$$\begin{aligned} \mathcal{I}_t &= \mathrm{TopK}_i \; s(\mathbf{q}_t, \mathbf{k}_i; \mathbf{m}_i), \\ \widehat{\mathbf{y}} &= f_\theta(\mathbf{x}_{t-L+1:t}, \{\mathbf{v}_i\}_{i\in\mathcal{I}_t}) \, . \end{aligned} \tag{24}$$

Here, TopK implements the read operator $\mathcal{R}$ via nearest-neighbor retrieval, and s is the similarity measurement. We further characterize the retrieval-augmented memory by the *store*, which determines the representation, capacity, and persistence of retained information; the *retriever*, which implements the read operation; and the *integration mechanism*, which determines how recalled information affects the output. These components need not operate over the same representation. A historical episode, for example, may be indexed by a learned or spectral embedding while the information returned to the predictor remains the original episode. Conversely, some systems retain only an encoded object and never return the underlying trajectory.

We therefore classify retrieval memory primarily by the content $\mathbf{v}_i$ that is persistently retained and made available for recall, rather than by the metric or backbone used to access it. Hybrid stores can combine more than one type, for example pairing a trajectory with textual context. The store may be built from the training set,

observations accumulated before deployment, or a cross-series archive. In all cases, temporal validity is essential: keys must be computable from information available at query time, and any retrieved continuation or outcome must already have been observed to avoid leakage.

7.1 Types of Retrieval Memory

7.1.1 Instance Retrieval Memory

This memory retains concrete historical cases such as windows, trajectories, context–future pairs, or examples from related series so that a current query can be grounded in specific precedents. In memory terms, these individually addressable cases form an *explicit* store, whose capacity can scale with the amount of retained data. The store is typically maintained as a per-dataset index and accessed through similarity-based retrieval. This is the main form of retrieval memory in time-series forecasting and naturally targets recurring patterns, long-range dependence, and data-sparse series.

Early retrieval-based forecasting established the idea that historical cases can serve as external predictive evidence rather than merely as training examples. ReTime (Jing et al. 2022) provides one of the earliest general formulations, motivated by the observation that forecasting uncertainty increases when only limited target history is available. It theoretically relates this uncertainty to prediction error and argues that conditioning on relevant reference series can reduce it. Operationally, ReTime separates *relational retrieval*, which identifies related series using structural relations rather than relying only on potentially sparse query values, from *content synthesis*, which combines the target and retrieved references to produce the prediction. Its extension to spatiotemporal forecasting and imputation further established retrieval as a reusable time-series mechanism rather than a forecasting-specific heuristic.

Subsequent work shifted from retrieving related series to retrieving concrete historical analogues from the forecasting corpus. RAFT (Han et al. 2025) is a representative formulation: for a given query, it searches the training history for similar observed patterns and retrieves the corresponding realized futures, so that previously observed continuations become direct evidence for forecasting. This makes retrieval complementary to parametric learning: the predictor need not encode every recurring trajectory in its weights, because relevant precedents can instead be supplied at inference time. Empirically, RAFT shows that this relatively simple augmentation is effective across a broad set of forecasting benchmarks.

Other methods develop the same instance-memory principle through different retrieval and integration mechanisms. RATSF (Wang and Cui 2024) constructs a time-series knowledge base and introduces retrieval-augmented cross-attention to assimilate historical segments, targeting strongly non-stationary service-volume forecasting. RATD (Liu et al. 2024) transfers retrieval into probabilistic generation: an embedding-based retriever selects reference sequences, which then guide the denoising process of a diffusion forecaster rather than being directly fused with a deterministic prediction. These works therefore preserve the same memory object, such as historical examples, while changing how retrieved cases influence the predictor.

The second branch uses instance retrieval to adapt time-series foundation models without modifying the information stored in memory. RAF (Tire et al. 2024) investigates retrieval specifically in the zero-shot TSFM setting and develops strategies to select related series and incorporate them into pretrained forecasters. TimeRAF (Zhang et al. 2025) goes further by learning the retriever end-to-end and introducing channel prompting to integrate retrieved knowledge, while TS-RAG (Ning et al. 2025) uses an Adaptive Retrieval Mixer to weight retrieved contexts and their associated future horizons before prediction. Across these methods, the progression is primarily in *how retrieved instances are selected and fused*.

More specialized extensions broaden the same idea across predictors and domains. TimeRAG (Yang et al. 2025) converts retrieved sequences into context for an LLM-based forecaster, while RAST (Ruan et al. 2026) adapts retrieval to traffic prediction through fine-grained spatiotemporal patterns and separate spatial–temporal query construction. Other work changes the access rule rather than the memory itself: SpecReTF (Nguyen et al. 2026a) introduces frequency-aware retrieval with temporal recency, and Kang et al.'s CRAFT (Kang et al. 2026) extends spectral matching to multivariate channel structure. Instance retrieval has also begun to appear outside forecasting, including historical positive/negative cases for anomaly detection in LLMAD (Liu et al. 2025) and in-domain normal examples for test-time adaptation in RATFM (Maru and Sato 2025).

7.1.2 Latent Retrieval Memory

Latent retrieval memory retains compact, derived representations as first-class memory entries, such as embeddings, latent states, learned tokens, or priors. Although its contents are encoded, latent retrieval memory remains *explicit* when the stored vectors or tokens are individually addressable; such banks can grow with available data, are typically read by embedding similarity, and are commonly retained across queries. This can reduce storage cost, make heterogeneous observations comparable in a shared space, and emphasize task-relevant structure that is difficult to express through a fixed distance over raw values.

ALER-TI (Truong et al. 2026) illustrates why retaining latent objects can be preferable to retrieving raw observations. In imputation, the incomplete query and a complete historical candidate are observed under different missingness patterns, so similarity measured directly in the observation space can be poorly defined. ALER-TI addresses this mismatch through Latent Embedding Alignment, applying post-hoc masking in latent space so that complete historical embeddings can be pre-computed and cached while remaining comparable with corrupted queries. Retrieval therefore operates over a representation specifically constructed to preserve useful historical information under missingness rather than over the raw trajectory itself.

ReDiTT (Lyu et al. 2026) provides a complementary motivation in asynchronous event sequences, where raw observations combine continuous inter-event times, discrete event types, variable sequence lengths, and padding. It encodes historical sequences into latent tokens, maintains these representations in a reference bank, and retrieves structurally similar latent sequences during both training and inference. The

retrieved latent representations are then supplied through cross-attention to a conditional diffusion transformer, providing trajectory-specific guidance that stabilizes long-horizon generation. Here, the latent representation is not just an indexing key. Rather, it is itself the persistent object retrieved and consumed by the predictor, making ReDiTT a clear instance of latent retrieval memory under our definition.

GTR (Cao et al. 2026) provides another example in multivariate time series forecasting, where a global temporal embedding represents long-range periodic patterns across the entire cycle. Given the current input and its position within the global cycle, GTR retrieves the corresponding latent segment and aligns it with the local multivariate sequence. Rather than retrieving a concrete historical trajectory, GTR retrieves a compact representation of global temporal structure, allowing long-range periodic information to be accessed without extending the input context.

SteerCast (Do et al. 2026) adopts a complementary latent-retrieval strategy for decoder-only forecasters, shifting from trajectory imitation to latent guidance. Rather than storing raw future sequences, it constructs a database where each history latent representation is paired with a *steering vector* defined in the model's latent space—capturing the residual difference between the ground-truth continuation and the model's own prediction. At test time, the method retrieves and aggregates these steering vectors to inject a corrective latent signal into the forecaster's hidden states at every step of generation.

The boundary with instance retrieval memory is determined by the role of the stored representation, not simply by the use of an encoder. If a method stores $\mathbf{k}_i = g(\mathbf{x}_i)$ only as a search key but returns the original historical case $(\mathbf{x}_i, \mathbf{y}_i)$, we consider it as *instance retrieval memory with latent retrieval*. We reserve *latent retrieval memory* for systems in which the encoded object itself is persistent and constitutes the principal recalled information. This distinction is particularly important for foundation-model retrieval, where learned encoders are often used for matching even though the retrieved value remains an identifiable historical segment. Latent retrieval memory gains compactness and abstraction, but can also discard rare details that later become predictive, making the representation objective part of the memory-design problem.

7.1.3 Knowledge Retrieval Memory

Knowledge retrieval memory stores information that is not simply another realization of the target series, including text, semantic descriptions, exogenous events, metadata, and structured knowledge. These individually addressable knowledge records form an *explicit* store that can grow with the available corpus, is read through semantic or structured retrieval, and is typically persistent across queries. Knowledge retrieval memory therefore provides context about the process rather than another trajectory to imitate.

TRACE (Chen et al. 2025) represents a shift from retrieving numerically similar trajectories toward retrieving information grounded in the semantic context of a series. It learns a multimodal representation that aligns time-series channels with associated textual descriptions, using fine-grained channel-level alignment and hard-negative mining to make the embedding space semantically discriminative. The

resulting retriever supports both time-series-to-text and text-to-time-series retrieval, allowing downstream models to access contextual evidence that numerical similarity alone may not recover. TRACE therefore contributes not only another retrieval metric, but also a mechanism for defining *semantic relevance* across modalities.

SERAF (Zhou et al. 2026) addresses a related problem from a different direction. Rather than requiring an externally annotated text corpus, it derives textual descriptions of historical segments and performs two parallel retrievals: one according to numerical time-series similarity and another according to semantic similarity between the generated descriptions. Their retrieved futures are then selectively combined, allowing two histories that differ in local shape or scale but share higher-level characteristics such as trend or volatility to inform one another. This makes semantic retrieval complementary to instance-level numerical matching, particularly under non-stationarity where nearest trajectories are not necessarily the most informative precedents.

Other systems instantiate knowledge retrieval in domain-specific forms. Financial forecasting systems retrieve contemporaneous news or textual evidence alongside market histories, while healthcare models such as EMERGE (Zhu et al. 2024) and REALM (Zhu et al. 2024) connect longitudinal observations and clinical text to biomedical knowledge graphs and retrieve relevant entities or relations. These systems differ in knowledge source and downstream task, but share the same memory principle: the recalled value contributes information about the temporal process that is not represented simply as another historical trajectory.

Knowledge retrieval memory thus broadens retrieval from "what happened in a similar historical case?" to "what external information is relevant to the current temporal state?" It can also complement instance retrieval memory: a retrieved trajectory may be accompanied by text or metadata that helps determine whether the analogy is valid. The main difficulty is temporal and semantic alignment. Retrieved knowledge must match the correct entity and variable, but also remain valid at the relevant time and under the current regime; otherwise, semantically related but stale context can be as misleading as an outdated historical example.

7.2 Memory Access and Integration

Across the three memory types, reading is query-dependent, whereas writing typically consists of constructing or updating the external store and persistence is usually per-dataset. Instance memories are commonly searched with fixed similarities such as correlation, Euclidean distance, or dynamic time warping. Learned encoders replace hand-designed metrics with embedding similarity, while time-series-specific retrieval incorporates frequency, phase, recency, channel structure, calendar alignment, stationarity, or regime metadata. These choices define the *read operation* over the store. In particular, a spectral or latent key should not be interpreted as latent retrieval memory when the returned value is still a concrete historical episode.

The central challenge is that *similarity is not equivalent to predictive relevance.* The ideal retriever should identify the memory that most improves downstream inference, not merely the nearest item under a generic distance. This distinction becomes especially important under non-stationarity, where a highly similar example from an

obsolete regime may be less useful than a moderately similar example from the current regime. Temporal metadata, structured similarity, and learned relevance functions provide different ways to address this mismatch.

Retrieved content is then incorporated through several recurring mechanisms. It may be appended to the input, fused through cross-attention or adaptive mixers, used as generative conditioning, or converted into prompt context. Input augmentation keeps retrieved evidence directly inspectable, whereas attention or latent fusion can select useful information more flexibly. Generative conditioning allows retrieved examples or priors to shape an entire predictive distribution, while prompting enables LLM-based systems to reason over retrieved evidence. These are *access and integration strategies*, not additional memory types: the same instance store can support an MLP, diffusion model, TSFM, or LLM depending on how its entries are consumed.

7.3 Coverage and Limitations

Retrieval memory is heavily concentrated in forecasting, where instance retrieval memory dominates and increasingly interacts with foundation models, diffusion forecasters, and multimodal evidence. Outside forecasting, the literature is much thinner: retrieval-based anomaly detection has a strong recent example in LLMAD (Liu et al. 2025), while imputation, classification, reasoning, and decision-making remain relatively sparse. Latent and knowledge memories are particularly recent, suggesting that only a small subset of possible memory-type–task combinations has been explored. Section 9 provides the task-centric comparison across memory architectures.

Across tasks, retrieval is characterized by explicit representations combined with a data-scaled capacity that grows seamlessly as the available corpus expands. It relies on query-dependent reads, limited or predominantly offline writes, and per-dataset persistence, while the external store can be updated without retraining the predictor. Instance retrieval memory additionally provides concrete precedents, latent retrieval memory offers compact reusable representations, and knowledge retrieval memory incorporates information unavailable in the numerical signal. Their corresponding limitations are retrieval and index-maintenance cost, dependence on the relevance function, loss of interpretability or detail in latent representations, and temporal/provenance issues for external knowledge. All three also provide little benefit when the store lacks an informative precedent. Finally, retrieval memory must be distinguished from the neighbouring classes in our taxonomy. Trainable prototype banks remain part of explicit memory (Section 6) when their capacity is architecturally bounded and learned jointly with the model, even if their read operation resembles retrieval. Conversely, the methods surveyed here largely use an unbounded external store with predefined or simply appended contents and focus on query-dependent reads of useful evidence. Once a model assumes active write access—using a controller or policy to decide what should be written, revised, promoted, or forgotten over time—the system shifts from data-scaled to policy-bounded capacity, moving beyond passive retrieval into the agentic memory considered in 8.

Remark 4 Retrieval-augmented memory is typically explicit in representation, unbounded in capacity, read through query-dependent retrieval, and persistent at the dataset level. Its

Table 3 Functional memory types instantiated for time series. Decay refers to how quickly an entry loses validity under regime change.

Type	Stored content	Written	Read at	Decay
Episodic: *"What happened in similar cases before?"*	Context, forecast, and error triples	per instance	prediction, reflection	fast
Semantic: *"What is generally true of this series?"*	Regimes, calendar rules, exogenous facts	once evidence accrues	prediction, scoring	medium
Procedural: *"How should I go about forecasting this?"*	Tool trajectories, model-selection playbooks	once evidence accrues	planning, action	slow

strength lies in scalable access to concrete experiences, latent representations, or contextual knowledge without requiring all useful information to be compressed into model parameters. However, the store remains largely passive: retrieval determines what is recalled, but not what should continue to be remembered. Under non-stationarity, previously useful memories may become stale or misleading, motivating agentic memory mechanisms that actively govern writing, updating, and forgetting (Section 8).

8 Agentic Memory

Sections 6 and 7 described stores that a model *reads*. In both cases, the memory store is essentially passive in one specific sense: the policy governing what is retained, retrieved, or revised is fixed before deployment. An explicit module may well write to its slots, but through a gate whose behaviour was set during training (Section 6.3), and a retrieval index is built once from a corpus and thereafter only queried. Neither decides what is worth keeping in the first place, notices when a stored item has stopped being true, or removes it. Agentic memory delegates those three decisions to a controller and remains an emerging, early-stage paradigm in time series.

For non-stationary time series, these decisions are crucial to ensure the quality of memory contents. Recalling last year's demand response to a promotion is wrong if the price regime has since changed, and the higher the similarity score, the more confidently the agent will be wrong. Static retrieval augmentation inherits this problem (Section 7) because it can retrieve the most similar episode, but similarity alone cannot tell whether that episode remains valid.

Concurrent surveys cover the surrounding agent architecture, including reasoning and agentic systems in time series (Chang et al. 2025) and agent memory in general (Hu et al. 2025). We survey *the store* rather than the whole agent, and we place it

on the same spectrum as the explicit modules of Section 6 and the retrieval indices of Section 7, which makes the continuity across three classes visible and provides a lens complementary to those architecture-level accounts.

8.1 Mechanism

Definition 5 (Agentic memory) An agentic memory is an external store $\mathcal{M}$ over which a controller π governs three policies: a *write* policy determining which artifacts of an interaction are persisted and in what abstracted form, a *read* policy determining what is recalled and when, and a *forget* policy determining what is revised, demoted, or discarded. The store evolves across interactions rather than being fixed per query.

Following the lifecycle formalization now standard in the agent-memory literature (Hu et al. 2025), an agent-–environment interaction at step t emits artifacts ϕ_t that comprise a reasoning trace, the tools invoked, the agent's output, and, when available, feedback on the outcome. This lifecycle directly specializes the generic memory formulation in Section 2, mapping generic observations to richer interaction artifacts ($\mathbf{o}_t \to \phi_t$) and factorizing the unified write operator $\mathcal{W}_\phi$ into distinct formation ($\mathcal{F}$) and evolution ($\mathcal{E}$) transformations. Memory then evolves as

$$\begin{aligned} \mathcal{M}_{t+1} &= \mathcal{E}(\mathcal{F}(\mathcal{M}_t, \phi_t)) \quad \Big[\equiv \mathcal{W}_\phi(\mathcal{M}_t, \phi_t)\Big], \\ r_t &= \mathcal{R}(\mathcal{M}_t, o_t, Q) \qquad \Big[\equiv \mathcal{R}_\psi(\mathcal{M}_t, o_t, Q)\Big]. \end{aligned} \tag{25}$$

where $\mathcal{F}$ forms memory candidates, $\mathcal{E}$ consolidates and prunes them, and $\mathcal{R}$ returns the signal on which the predictor conditions, with o_t denoting the current observation or context derived from the input time series $\mathbf{x}_{1:t}$ and Q the current task specification. The three policies of Definition 5 map onto these operators: the write policy is realized by $\mathcal{F}$, the forget policy by $\mathcal{E}$, and the read policy by $\mathcal{R}$; the write and forget policies thus jointly constitute the generic write operator W_ϕ. All three operators are issued by the controller π (Definition 5) rather than fixed before deployment, which separates this class from Sections 6 and 7. For simplicity, we leave π implicit in the notation.

Retrieval memory (Section 7) can be viewed as the limiting case in which write-time filtering and post-deployment evolution are fixed or absent, so that only $\mathcal{R}$ remains active at inference. Internal memory (Section 5) instead is the case in which $\mathcal{F}$ is instantiated as the model's recurrent state-update function, while $\mathcal{E}$ and $\mathcal{R}$ require no separate memory operations. In the agentic systems surveyed below, these operators are implemented through different mechanisms. For example, MemCast (Tao et al. 2026) uses model-based extraction and reflection to form and refine memory entries, while Argos (Gu et al. 2025) writes rules and revises them against feedback.

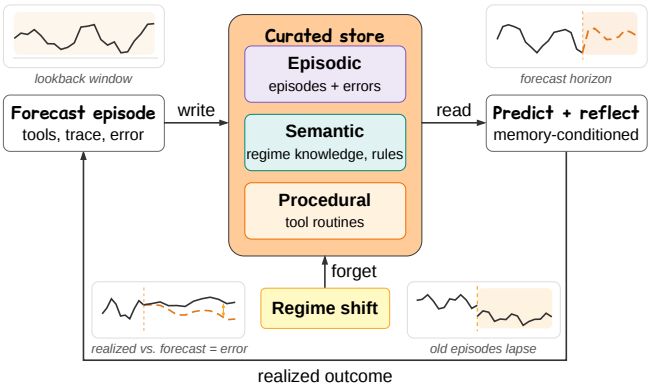


Fig. 6 The agentic memory loop for time series, illustrated on forecasting. A *write* policy distills a forecasting episode into a store partitioned by memory type, a *read* policy recalls entries to condition prediction and reflection, and the realized outcome is written back once the horizon elapses. The loop itself is task-agnostic; we use forecasting to make the *forget* policy concrete because the horizon supplies a natural point at which an entry's validity can be checked. Beyond memory operations in Sections 6 and 7, an agentic *forget* policy can demote entries whose validity has lapsed under regime change, which is the mechanism that non-stationarity necessitates.

For $\mathcal{R}$, CastFlow (Pan et al. 2026) selects by similarity, MemCast reweights by confidence, and Cast-R1 (Tao et al. 2026) folds the retrieval decision into a learned policy.

Overall, agentic memory is *explicit* in representation and, in principle, policy-bounded (refer to Section 2.4) in capacity, with the controller able to determine how much information is retained through its forgetting policy rather than through a fixed architectural limit. Read and write operations are likewise *controller-issued*, allowing the memory contents and access patterns to adapt over time. In the fully online setting, the store persists across interactions and can be updated as new experience accumulates. However, as discussed below, most existing time-series systems do not yet operate in this fully online regime, instead constructing their stores offline and keeping them largely fixed at deployment. Figure 6 illustrates these components.

Classifying memory by function, as the agent literature does (Sumers et al. 2024; Hu et al. 2025), reveals that most modules serve a small set of shared roles within a single store, despite differences in framework design and terminology. Table 3 defines these three functional types by their target questions and stored contents, with Section 8.3 expanding on the remaining columns.

8.2 Type of Agentic Memory

A recent position paper recasts forecasting as an agentic process of perception, planning, action, reflection, and memory (Cheng et al. 2026). The memory machinery itself is inherited from NLP agents, where verbal self-reflection (Shinn et al. 2023), insight distillation (Zhao et al. 2024), workflow abstraction (Wang et al. 2024), and agentic retrieval (Singh et al. 2025) provide the write and read mechanisms that time-series systems now adopt. Instead of treating memory as one module in an agent loop, we

Table 4 Representative agentic memory systems for time series, organized by the store each maintains rather than by the agent architecture that surrounds it, which is what separates this account from agent-centric treatments (Cheng et al. 2026; Chang et al. 2025). *Type*: E, S, P, and W denote episodic, semantic, procedural, and working (within-episode) memory. *Scope*: per-sequence (resets when the interaction ends), per-dataset (distilled offline and frozen at test time), or online. *Forget* indicates an explicit revision, demotion, or validity-decay policy. *Task* names the canonical time-series tasks the store serves, where *All four* denotes forecasting, anomaly detection, classification, and imputation.

System	Type	Memory content	Write policy	Scope	Forget	Task
MemCast (Tao et al. 2026)	E, S, P	patterns, wisdom, laws	offline distillation + reflection	per-dataset	conf. decay	Forecasting
Cast-R1 (Tao et al. 2026)	W	decision-relevant state	RL-learned, multi-turn	per-sequence	none	Forecasting
CastFlow (Pan et al. 2026)	P	tool-use trajectories	sampled successful paths	per-dataset	none	Forecasting
Nexus (Das et al. 2026)	E, S	error signatures, guidelines	backtest calibration	per-dataset	none	Forecasting
AlphaCast (Zhang et al. 2025)	E, S	historical cases, domain knowledge, context	context/retrieval	per-sequence	none	Forecasting
Argos (Gu et al. 2025)	S	anomaly rules	multi-agent generate and repair	per-dataset	rule revision	Anomaly det.
AnomaMind (Tao et al. 2026)	S, P	anomaly patterns, domain knowledge	offline pattern mining	per-dataset	none	Anomaly det.
Agentic-RAG (Ravuru et al. 2024)	P	prompt pool of skills	learned, hierarchical	per-dataset	none	All four

treat the memory itself as the object of study and ask what its write, read, and forget policies commit the system to. This shift allows an agentic memory to be compared with the explicit modules and retrieval indices on shared axes (Section 6 and 7). We organize the time-series agentic systems by the memory type they principally populate (Table 4). Across these types, the same lifecycle operators are realized differently. The following sections will show how they form, evolve, and retrieve memory through

model-based extraction, rule-based revision, similarity-based retrieval, and learned policies.

8.2.1 Episodic and working stores.

MemCast distills the training data into a hierarchical memory whose lowest tier summarizes prediction outcomes into *historical patterns* and conditions inference on them, with a per-entry confidence score updated dynamically as the entry proves useful. This confidence adaptation lets the memory store evolve at inference time while explicitly avoiding test-data leakage, so the historical tier plays the role of an episodic memory that is refined over time. Nexus (Das et al. 2026) maintains a domain-level calibration loop that scores past forecasts against ground truth across several historical splits and converts the resulting error signature into review guidelines for subsequent forecasts. Notably, a guideline is produced only if it improves accuracy beyond a fixed threshold on a held-out validation split, which makes the write itself selective based on actual performance. AlphaCast (Zhang et al. 2025) uses historical cases together with domain knowledge and contextual information to support iterative forecasting, but these resources are assembled for each forecasting interaction rather than maintained as a persistent, evolving store.

A further group operates one level down, retaining state only for the duration of a single analysis: Cast-R1 carries decision-relevant evidence across the turns of a multi-step episode, ReasonTSC (Zhou et al. 2025) accumulates reasoning state before fusing it into a classification decision, and TS-Reasoner (Ye et al. 2026), TimeCopilot (Garza and Rosillo Garcia 2025), TimeSeriesScientist (Zhao et al. 2025), and the multi-agent ColaCare (Wang et al. 2025) retain intermediate findings and tool outputs within a single run. Cast-R1 is a useful boundary case: its learned policy decides what evidence to retain across reasoning steps, but the resulting state remains confined to a single forecasting episode. It therefore illustrates agentic *state management* without persistent memory across episodes. This is working memory in the sense of Hu et al. (2025). It is excluded from Table 3 because it does not persist past the episode that created it, and so cannot decay under regime change.

8.2.2 Semantic stores.

While episodic stores retain experiences, semantic stores distill reusable knowledge, and they differ mainly in what that knowledge is asked to do. MemCast supports verification by inducing *general laws* from extracted temporal features and applying them as criteria for reflective iteration. AnomaMind (Tao et al. 2026) instead supports grounding by providing anomaly-type descriptions and visual prototypes as external knowledge for its coarse-to-fine detection workflow. This knowledge is extracted offline from the training data and consulted alongside numerical diagnostic operators, so the store contributes contextual priors while evidence is gathered separately at inference. The core detection decisions are learned under verifiable feedback, so the semantic memory informs reasoning without serving as the final decision rule.

Argos (Gu et al. 2025) takes this idea a step further by turning stored knowledge into executable rules. These rules are generated, validated, and refined before deployment, making the repository both interpretable and directly testable. Each rule

is implemented as a function that classifies a time window as normal or anomalous, so the stored knowledge is evaluated by running it on data. Normality is captured through explicit anomaly *rules* that collaborating agents write from observed patterns, repair when they throw syntax or runtime errors, and score on validation data before admitting them. A dedicated review agent checks each candidate against validation data and, when it detects performance regressions, derives revisions rather than discarding the rule outright, so the store is curated through an explicit generate-validate-repair loop. At deployment, the admitted rules run at low cost for online scoring and are combined with existing detectors, allowing the semantic store to serve a continuously monitored stream. Importantly, the rule repository remains inspectable, allowing operators to examine and modify the rules, which provides the auditability property discussed in Section 3.

8.2.3 Procedural stores.

Beyond semantic knowledge, procedural stores capture how to act. CastFlow (Pan et al. 2026) maintains a *strategy memory* pairing each lookback window with its best-performing tool schedule. At planning time, the agent selects diagnostics by similarity to past windows, from change-point detection to spectral entropy, instead of scheduling tools zero-shot. The store is populated by expanding each training instance into several parallel exploration paths, scoring the resulting forecasts against ground truth, and archiving only the trajectory that performed best. Notably, the authors report a sweet spot in how many trajectories to recall: too few leave the agent under-guided, while too many inject redundant context that degrades the reasoning, which frames retrieval size as a design choice for a procedural store. Similarly, MemCast's middle tier plays the same role under the name *reasoning wisdom*, distilled from inference trajectories and used to select among candidate reasoning paths (Tao et al. 2026).

AnomaMind (Tao et al. 2026) pairs its semantic store with a procedural component that governs how the coarse-to-fine detection workflow is applied, and Agentic-RAG (Ravuru et al. 2024) organizes reusable skills as a hierarchical prompt pool. In the latter, a master agent routes a query to task-specific sub-agents, each drawing on its own prompt pool, and the procedural store is indexed by task. The memory content is constructed from successful runs. This success-only sourcing is also a limitation: the store records what worked, yet cannot capture the conditions under which a routine failed, so it cannot warn against reusing a strategy in a regime where it no longer applies.

8.3 Memory Management and Integration

In this section, we discuss five dimensions that separate agentic memory systems, each of which raises an open question.

What is written, and at what abstraction? The formation operator $\mathcal{F}$ is implemented at different levels of abstraction. Raw episodes are cheap to write and transfer poorly, while abstracted laws and routines transfer but risk laundering a coincidence into a rule. MemCast's three-tier design is comprehensive, keeping all three abstraction levels at once.

How is memory integrated at inference? The retrieval operator $\mathcal{R}$ determines how memory is integrated at inference. CastFlow reads at planning time and AnomaMind reads at every stage of a fixed coarse-to-fine workflow, letting the read point be built into the loop. Argos applies its whole rule repository to each window, discarding the need for a selective read mechanism. MemCast is the closest to a learned read, since a per-entry confidence score reweights what is recalled as entries prove useful, and Cast-R1 subsumes recall into an RL-learned decision policy. None of the systems we surveyed can operate without temporal context, so an uninformative store is queried at the same cost and shapes the output as much as an informative one.

How are memory operations controlled? Most systems use heuristic or prompt-level policies. Cast-R1 instead learns the surrounding decision policy by supervised fine-tuning followed by multi-turn reinforcement learning, and CastFlow refines its workflow with verifiable rewards. The general trend in agent memory, which exposes store, retrieve, update, and discard as learned actions rather than fixed heuristics, has not yet been applied to memory operations for time series.

When is the store allowed to change? The evolution operator $\mathcal{E}$ governs when stored content is consolidated, revised, or persisted. With the partial exception of Argos, the memories surveyed here are *distilled offline from the training split and frozen at inference*. MemCast adapts only per-entry confidence, and avoids leaking the test distribution, while CastFlow builds a case library per domain in advance, and Nexus calibrates on historical splits. Under our taxonomy, these systems are therefore closer to *curated retrieval*, a store whose contents an agent chose but which is then read-only, than to the fully online memory the term "agentic" suggests. The true online regime, in which a deployed forecaster writes and forgets as it goes, is largely underexplored.

How should different memory types decay? Once a store is allowed to change, the three types in Table 3 should not be governed by a common rule. Procedural knowledge about which tool to use can stay relevant after a regime change, while specific past episodes become misleading the moment the regime shifts. A single forgetting rule across all three types therefore either discards routines that were still useful or keeps episodes that are already outdated. Our survey found limited evidence of type-differentiated decay, a gap Section 11 develops further.

8.4 Coverage and Limitations

As shown in the *Scope* and *Forget* columns of Table 4, agentic memory mechanisms for time series are still emerging. Among surveyed methods, only one revises its store after deployment, while a second updates a confidence score without ever removing an entry. The rest build a store from the training split, freeze it, and read from it only. As Section 8.3 noted, this is largely the curated retrieval regime. Most systems are close to Section 7: an agent selects and structures the contents, but the store is still frozen at inference time. The literature has so far focused far more on *write* than on *forget*. The same immaturity is also visible in task coverage. Agentic memory is concentrated almost entirely in forecasting, and the few anomaly detection systems populate semantic rather than episodic stores (Table 4). This concentration leaves open whether the same memory mechanisms transfer to other time-series tasks.

Agentic memory is the class in this survey that most explicitly exposes memory revision as an operational mechanism, making it particularly suited to adaptation under drift. It accumulates experience without retraining, and its stores are inspectable enough to support audit. Against this, inference cost can be high and may grow with the size of the store, while write and forget policies are often hand-designed and rarely evaluated in isolation. Abstraction can also turn accidental correlations into spurious "laws", and an evolving store may amplify its own errors when incorrect predictions are written back as memory. Moreover, no benchmark yet evaluates memory decisions rather than final forecast accuracy (Section 10), leaving the benefit of curation over plain retrieval largely unmeasured.

9 Cross-Cutting View: Memory by Task

The previous sections organized memory by the *problem* it solves (Section 3) and by the *mechanism* that implements it (Sections 6 to 8). This section focuses more on the question a practitioner actually starts from, which is not "how does memory work?" but "I am building an anomaly detector, so what should I use?". It reorganizes the same methods by *task*, so the reader can look up a task and see which memory mechanisms have been tried and which method represents each.

The task view is very different from the problem view of Section 3. A single task usually spans several problems at once; for instance, forecasting involves long-horizon dependence, recurring analogues, and drift together, while anomaly detection is built almost entirely around recognizing departures from stored normal patterns. Because of this, the task view exposes a different perspective. Instead of showing which *problem* is under-served, it shows which *application areas* have barely adopted a given class of memory at all. Table 5 places representative time-series memory methods into a task row and a mechanism column.

Forecasting. Forecasting predicts future values of a series from its past. It is the only task where all four memory types are already well represented, because forecasting benefits from several forms of history such as compressed temporal state for recent dynamics, explicit memory for recurring motifs or cross-series structure, and retrieval for rare or distant analogues. Retrieval is particularly strong here because a forecast often depends on a small number of relevant past episodes rather than the entire history (Han et al. 2025; Liu et al. 2024). Agentic memory is newer, but becomes useful when the model must decide what historical evidence to retain or revise as conditions change (Tao et al. 2026).

Classification. Classification assigns a whole series to one of a set of classes. The task naturally favors explicit memory because class identity is often determined by matching a series against representative shapes, prototypes, or discriminative subsequences (Tang et al. 2020; Zhang et al. 2020). Retrieval, however, is almost absent. This suggests a possible gap, as retrieving similar labelled series could provide both stronger evidence and a direct explanation for the assigned class, especially when classes are rare or heterogeneous.

Table 5 Time-series memory methods by task and mechanism. The symbol ∅ *open* marks task–mechanism combinations the reviewed literature has barely addressed.

Task	Internal	Explicit	Retrieval	Agentic
Forecasting	DeepAR (Salinas et al. 2020), DA-RNN (Qin et al. 2017), TreNet (Lin et al. 2017), LSTNet (Lai et al. 2018), SAnD (Song et al. 2018), Cloudlstm (Zhang et al. 2021), P-sLSTM (Kong et al. 2025), FiLM (Zhou et al. 2022a), TimeMachine (Ahamed and Cheng 2024), MambaTS (Cai et al. 2024), ms-Mamba (Karadag et al. 2026), Bi-Mamba+ (Liang et al. 2024), SST (Xu et al. 2024), Attraos (Hu et al. 2024)	MTNet (Chang et al. 2018), MAGL (Liu et al. 2022), PM-MemNet (Lee et al. 2022), MegaCRN (Jiang et al. 2023), STanHop (Wu et al. 2024), HAMN (Lee et al. 2025), MEMTS (Yu et al. 2026), TS-Memory (Lyu et al. 2026)	RATD (Liu et al. 2024), RAFT (Han et al. 2025), MQ-ReTCNN (Yang et al. 2022), RATSF (Wang and Cui 2024), RAF (Tire et al. 2024), TimeRAG (Yang et al. 2025), RAST (Ruan et al. 2026), CRAFT (Kang et al. 2026), ReDiTT (Lyu et al. 2026), SERAF (Zhou et al. 2026), TRACE (Chen et al. 2025), ReTime (Jing et al. 2022), TimeRAF (Zhang et al. 2025), TS-RAG (Ning et al. 2025), SpecReTF (Nguyen et al. 2026a), GTR (Cao et al. 2026)	Cast-R1 (Tao et al. 2026), MemCast (Tao et al. 2026), CastFlow (Pan et al. 2026), Nexus (Das et al. 2026), Alpha-Cast (Zhang et al. 2025), Agentic-RAG (Ravuru et al. 2024), TimeSeriesScientist (Zhao et al. 2025)
Classification	LSTM-FCN (Karim et al. 2019), MambaSL (Jung and Kim 2026)	DPSN (Tang et al. 2020), ShapeNet (Li et al. 2021), TapNet (Zhang et al. 2020), Memory Shapelet (Wan et al. 2024)	TRACE (Chen et al. 2025)	Agentic-RAG (Ravuru et al. 2024), ReasonTSC (Zhou et al. 2025), ColaCare (Wang et al. 2025)
Anomaly detection	LSTM-AD (Malhotra et al. 2015), EncDec-AD (Malhotra et al. 2016), MAAT (Sellam et al. 2025)	MEMTO (Song et al. 2023), MOMEMTO (Yoon et al. 2025), MemMambaAD (Li et al. 2025), Anomaly Trans. (Xu et al. 2022)	LLMAD (Liu et al. 2025), RATFM (Maru and Sato 2025)	Argos (Gu et al. 2025), AnomaMind (Tao et al. 2026), Agentic-RAG (Ravuru et al. 2024)
Imputation	GRU-D (Che et al. 2018), BRITS (Cao et al. 2018)	MMNet (Miao et al. 2025), PRIME (Yu et al. 2025)	ALER-TI (Truong et al. 2026)	Agentic-RAG (Ravuru et al. 2024)
Reasoning	∅ *open*	ChatTS (Xie et al. 2025), ITFormer (Wang et al. 2025)	EMERGE (Zhu et al. 2024), REALM (Zhu et al. 2024)	TS-Reasoner (Ye et al. 2026), TimeCopilot (Garza and Rosillo Garcia 2025), TimeSeriesScientist (Zhao et al. 2025)
Decision-making	LSTM-DRL (Zou et al. 2024)	∅ *open*	∅ *open*	Cast-R1 (Tao et al. 2026), MemCast (Tao et al. 2026), CastFlow (Pan et al. 2026)

Anomaly detection. Anomaly detection flags inputs that depart from normal behaviour. Explicit memory fits this task particularly well because normal patterns can be stored as a reference set and anomalies identified by their failure to match that memory (Song et al. 2023). Retrieval provides a complementary view by comparing an input with relevant historical examples rather than a fixed learned prototype. Agentic memory is also promising under drift, where the definition of "normal" changes and stale memory must be updated rather than blindly reused.

Imputation. Imputation fills in missing values within a series from the observed context around them. Most existing methods still rely on internal and explicit memory to propagate information from nearby observations. Yet the task is naturally suited to retrieval where missing segments can often be reconstructed from analogous subsequences elsewhere in the same series or in related series, as explored by ALERTI (Truong et al. 2026). The relatively sparse literature suggests that memory has so far been used mainly to preserve local context rather than to exploit broader historical analogues.

Reasoning. This newer task asks a model to interpret a series and answer questions or explain what it shows, usually with a language model. Here, raw Transformer context already provides explicit memory over the observed series, but this alone limits reasoning to what is in the current window. Retrieval extends the evidence available to the model, while agentic memory supports multi-step analysis by retaining intermediate findings, tool outputs, and previous conclusions (Ye et al. 2026; Garza and Rosillo Garcia 2025). The task therefore shifts the role of memory from preserving signal history toward preserving and organizing evidence for reasoning.

Decision-making. Here, a model acts on a series over time, taking decisions that serve two distinct purposes: executing an immediate action on the series, and generating outcomes that feed back into the system. Because the model must remember not only what happened, but also what it did and what followed, agentic memory is the natural candidate to accumulate these observations and revise later decisions (Tao et al. 2026; Pan et al. 2026). Explicit and retrieval memory remain largely open, despite the potential value of recalling similar past decision episodes before acting.

Overall, Table 5 suggests that memory requirements are strongly task-dependent. Explicit memory is most useful when stored patterns directly support the task, as in classification and anomaly detection, while retrieval is strongest when relevant historical analogues matter, as in forecasting. Agentic memory is a relatively recent direction, but its general-purpose design allows it to support a broader range of downstream tasks. Existing approaches already span forecasting, anomaly detection, and other time-series tasks, although the literature remains concentrated primarily on forecasting. This broader flexibility stems from the ability of an agent to combine internal context, explicit stores, retrieval, and task-specific tools within a single workflow. In contrast, the other memory types provide more specific mechanisms for storing or accessing information, making them well suited to some tasks but less natural for others. The open cells in the table therefore highlight memory capabilities that remain underexplored for particular tasks.

10 Benchmarks, Evaluation, and Resources

We organize representative resources from conventional numerical benchmarks through foundation-model, multimodal, reasoning, and agentic settings. These families are not mutually exclusive; for example, a benchmark may be both multivariate and multimodal, but the organization exposes how evaluation requirements expand as models move outward along the memory spectrum. Table 6 summarizes these benchmark families and highlights how each stresses a different aspect of memory, from retaining temporal information and sharing cross-series structure to retrieving external context and maintaining information across multi-step interactions.

Table 6 Representative benchmark families relevant to time-series memory. Archival conference/journal papers are prioritized; † denotes an emerging non-archival or workshop benchmark included because no mature archival equivalent currently exists.

Family	Representative resources	Primary evaluation role
General	TFB (Qiu et al. 2024); Monash Archive (Godahewa et al. 2021); UCR (Dau et al. 2019); TSB-AD (Liu and Paparrizos 2024)	Standard forecasting, classification, and anomaly-detection evaluation across heterogeneous datasets.
Multivariate	BasicTS+ (Shao et al. 2025)	Multivariate forecasting across diverse temporal and cross-variable dependencies; useful for testing shared and cross-variable memory.
Spatiotemporal	LargeST (Liu et al. 2023)	Large-scale traffic forecasting with long temporal coverage, many sensors, and metadata; useful for testing memory across many spatially related entities.
Hierarchical	M5 (Makridakis et al. 2022)	Forecasting across sparse product–store series and multiple aggregation levels.
Foundation model	ProbTS (Zhang et al. 2024); Chronos (Ansari et al. 2024); BOOM (Cohen et al. 2025)	Broad-horizon, held-out/zero-shot, and large-scale evaluation of universal forecasters.
Multimodal	Time-MMD (Liu et al. 2024); CiK (Williams et al. 2025); Time-IMM (Chang et al. 2025); TRACE (Chen et al. 2025)	Time series paired with text, context, irregular modalities, or cross-modal retrieval targets.
Reasoning	TimeSeriesExamAgent (Gwiazc et al. 2026); EngineMT-QA (Wang et al. 2025); TSQA (Kong et al. 2025); ChatTS (Xie et al. 2025)	Pattern understanding, question answering, explanation, and multi-step temporal reasoning.
Agentic	MAFS (Huang et al. 2025); TimeSeriesGym† (Li et al. 2025)	Agent cooperation and end-to-end time-series ML workflows; persistent-memory evaluation remains immature.
Memory-specific	SynTSBench (Tan et al. 2025); TS-Haystack† (Zumarraga et al. 2026)	Controlled temporal capability tests and long-context retrieval under increasing context length.

10.1 Standard Time-Series Benchmarks

10.1.1 General Time Series

Established task benchmarks remain necessary because a memory mechanism is useful only if it improves a well-defined downstream problem. TFB (Qiu et al. 2024) is the most comprehensive forecasting resource in this group. It is introduced to address three sources of unreliable comparison: limited domain coverage, biased model selection, and inconsistent evaluation pipelines, and evaluates both univariate and multivariate methods under a unified protocol across heterogeneous domains. For memory research, this breadth is useful for separating a mechanism that genuinely provides reusable historical information from one whose gains are specific to a small collection of frequently reused forecasting datasets.

Other archives provide complementary task coverage. The Monash Forecasting Archive (Godahewa et al. 2021) exposes heterogeneous frequencies, lengths, and missing-value characteristics, while UCR (Dau et al. 2019) remains the standard archive for classification. Anomaly detection requires particular care because conclusions can change substantially with the dataset and metric. TSB-AD (Liu and Paparrizos 2024) addresses this issue directly by curating 1,070 series from 40 datasets, benchmarking a broad range of detectors, and analysing evaluation-measure reliability rather than treating the metric as fixed. This is especially relevant to memory-based anomaly detectors, whose apparent gains from matching against stored normal patterns should not be conflated with biases in permissive evaluation metrics.

Imputation benchmarking is less consolidated. Recent archival work has emphasized that evaluation should reflect realistic MCAR, MAR, and NMAR missingness rather than only random deletion (Toye et al. 2025). Such protocols are important for memory-based imputers because the usefulness of a stored prototype or retrieved analogue depends on how the missingness process alters the observable evidence; however, there is not yet a shared imputation benchmark designed specifically around memory mechanisms.

Existing results suggest that memory can help across these conventional tasks, but that its value depends on whether the store contains reusable evidence. Representative examples span several memory forms: RAFT (Han et al. 2025) uses retrieved historical analogues for forecasting, MEMTO (Song et al. 2023) uses prototype-style explicit memory for anomaly detection and classification. These results establish that memory is useful across multiple standard time-series tasks, but current evidence remains fragmented across task-specific benchmarks. A broader evaluation is still needed to determine whether particular memory mechanisms generalize across the heterogeneous regimes represented by TFB (Qiu et al. 2024), and UCR (Dau et al. 2019).

10.1.2 Multivariate Time Series

Multivariate benchmarks are particularly relevant to memory because predictive information may be distributed across variables as well as across time. A memory mechanism may therefore retain not only past states of individual channels, but also recurring multivariate configurations and dependencies that reappear across different temporal contexts.

BasicTS+ (Shao et al. 2025) is the main benchmark for this setting. It is motivated by inconsistent conclusions across multivariate forecasting studies and standardizes training and evaluation while explicitly characterizing datasets by their temporal and cross-variable heterogeneity. Its large comparison across heterogeneous datasets shows why a single "best" multivariate architecture is difficult to identify: the usefulness of temporal modelling and cross-variable interaction depends strongly on the structure of the dataset. For memory-based models, this heterogeneity provides a natural stress test. Datasets with strong cross-variable structure can test whether shared or explicit memory captures reusable dependencies among channels, whereas more heterogeneous datasets test whether the model can selectively access useful channel relationships rather than indiscriminately sharing stored information. BasicTS+ is not itself a memory benchmark, but it provides the controlled diversity needed to determine when multivariate memory is actually useful.

Existing retrieval results on datasets covered by the BasicTS+ ecosystem make this heterogeneity concrete. RAFT (Han et al. 2025), Zhang et al.'s CRAFT (Zhang et al. 2025), and SpecReTF (Nguyen et al. 2026a) provide representative examples of instance, channel-aware, and frequency-aware retrieval on commonly used multivariate forecasting datasets. Across these approaches, memory gains are not uniform: retrieval is most useful when past multivariate configurations provide reusable future precedents, whereas the benefit shrinks or can become detrimental when similarity in the observed history does not imply similar future dynamics. The resulting insight is that multivariate memory should not be judged only by average accuracy; its value depends on the temporal and cross-variable structure that determines whether historical experience is actually reusable. A controlled comparison of these memory approaches under the exact BasicTS+ protocol remains missing.

10.1.3 Spatiotemporal Time Series

Spatiotemporal benchmarks extend the multivariate setting by attaching observations to a structured set of locations or nodes. This is particularly relevant to memory systems because useful historical information may come from another entity whose dynamics are spatially related to the query, making the size and structure of the shared memory part of the evaluation problem.

LargeST (Liu et al. 2023) was introduced to move traffic forecasting beyond the relatively small California Performance Measurement System (PeMS) benchmarks (Li 2025). It contains five years of measurements from up to 8,600 sensors together with sensor metadata, substantially increasing both the temporal archive and the number of entities that a model must handle. For memory-based forecasting, this scale tests whether a shared or retrieved store remains useful when candidate historical evidence is distributed across thousands of correlated streams, rather than only whether a model can exploit a small fixed graph. The commonly used PeMSD3, PeMSD4, PeMSD7, and PeMSD8 (Song et al. 2020) datasets remain useful controlled baselines, while LargeST provides the stronger scalability test.

Current traffic results provide comparatively strong evidence for memory. PM-MemNet (Lee et al. 2022), MegaCRN (Jiang et al. 2023), and MAGL (Liu et al. 2022) provide representative explicit or learned-memory approaches that reuse recurring

traffic patterns and shared spatial structure, while retrieval-based RAST (Ruan et al. 2026) shows that recalling historical spatiotemporal precedents contributes beyond the forecasting backbone in controlled ablations. Taken together, these approaches indicate that traffic is a favorable regime for memory because useful patterns recur across both time and related sensors, and because cross-sensor structure provides additional reusable evidence. What remains less established is whether the same advantage persists when the searchable history expands to the multi-year, thousands-of-sensors scale represented by LargeST (Liu et al. 2023).

10.1.4 Hierarchical Time Series

Hierarchical benchmarks stress a different form of memory sharing: information can be reused across aggregation levels rather than only across neighbouring channels or nodes. The M5 Accuracy Competition (Makridakis et al. 2022) is the canonical example, with 42,840 retail sales series organized across product, store, state, and aggregate levels. Many bottom-level series are sparse or intermittent, so their own local histories may provide little evidence for forecasting. This makes M5 a natural testbed for the P4 setting in Section 3: memory can be evaluated by whether information retained from related products, stores, or higher-level aggregates improves a data-sparse target rather than by simply extending that target's lookback window.

Current hierarchical-memory results support this form of cross-level sharing. HAMN (Lee et al. 2025), for example, allows sparse lower-level series to draw on memory formed from richer aggregate series and improves forecasting across multiple hierarchical datasets. The main empirical insight is therefore that memory can compensate for weak local history when related levels contain reusable structure. However, this benefit has not yet been established at the substantially larger hierarchy and sparsity represented by M5 itself.

10.2 Foundation-Model Benchmarks

Foundation models change the evaluation question from performance on a fixed training distribution to generalization across datasets, frequencies, and domains. Chronos (Ansari et al. 2024) helped establish this setting by pretraining tokenized time-series models on a large mixture of public and synthetic data and evaluating them on a broad collection of datasets, including zero-shot targets not used for task-specific fitting. For memory research, this creates an important confound: an external memory may appear useful because it supplies new evidence, but some of that information may already be encoded parametrically during pretraining. Evaluation should therefore distinguish gains from retrieved or explicit memory from gains attributable to pretraining overlap.

Current foundation-model results indicate that parametric pretraining and additional memory are complementary rather than substitutes. Retrieval-based approaches such as TS-RAG (Ning et al. 2025) and TimeRAF (Zhang et al. 2025) show that query-specific historical evidence can still improve strong pretrained forecasters, while TS-Memory (Lyu et al. 2026) and MEMTS (Yu et al. 2026) show a complementary direction in which reusable knowledge is internalized into compact memory modules to

avoid datastore retrieval at inference. Together, these approaches suggest that broad pretraining does not eliminate the value of task- or domain-specific memory; instead, the useful memory form depends on the trade-off between adaptable external evidence and bounded inference cost. Their gains also make provenance increasingly important, since memory benefits can otherwise be confounded with information already encountered during pretraining.

10.3 Multimodal Time-Series Benchmarks

Multimodal benchmarks are the natural counterpart to the knowledge-memory setting of Section 7, where useful evidence may be textual, contextual, or structured rather than another numerical trajectory. The key evaluation question is therefore stronger than whether an additional modality improves average accuracy: the benchmark should reveal whether the model can identify and use the specific external information relevant to the current temporal state.

Context-is-Key (CiK) (Williams et al. 2025) is particularly informative in this respect. Each forecasting instance pairs numerical history with carefully constructed textual context, and the tasks are designed so that the context is necessary for solving the instance correctly. This makes CiK closer to a memory stress test than a conventional multimodal dataset: a model that ignores the supplied contextual evidence should fail even if its numerical forecasting backbone is strong. For retrieval or knowledge-memory systems, it therefore provides a way to test whether recalled context changes the forecast in the direction implied by the evidence rather than merely being fused without effect.

TRACE (Chen et al. 2025) targets the access side of knowledge memory more directly. It aligns time-series channels with textual descriptions in a shared representation and evaluates both text-to-series and series-to-text retrieval, so retrieval quality can be measured independently of a downstream forecaster. This is valuable for the decomposition in Section 7: TRACE evaluates whether the correct cross-modal memory can be found before asking whether a predictor can use it.

Other resources broaden the setting. Time-MMD (Liu et al. 2024) provides aligned numerical and textual series across nine domains together with MM-TSFlib for multimodal forecasting, while Time-IMM (Chang et al. 2025) introduces asynchronous sampling, missingness, and multiple causes of irregularity. These extensions are useful for testing whether knowledge memory remains accessible when modality alignment is imperfect, or evidence arrives at different times.

Existing multimodal results suggest that memory is most useful when the recalled evidence contributes information that is genuinely absent from the numerical history. TRACE (Chen et al. 2025), SERAF (Zhou et al. 2026), and EMERGE provide representative approaches that retrieve semantically aligned descriptions, combine numerical and semantic precedents, or ground temporal observations in external clinical knowledge. Across these examples, the gain comes not merely from adding another modality, but from retrieving complementary evidence that changes what can be inferred from the time series alone. The benchmark implication is therefore that multimodal memory should be judged by whether it finds temporally and semantically

aligned evidence, while robustness to asynchronous, missing, or misaligned modalities remains much less characterized.

10.4 Reasoning Benchmarks

Reasoning benchmarks evaluate capabilities that conventional forecasting errors cannot express. TimeSeriesExamAgent (Gwiazda et al. 2026) is an important recent example because it treats benchmark construction itself as a scalable problem. It first defines controlled reasoning categories, including pattern recognition, noise understanding, similarity, anomaly detection, and causality, and then uses an agentic generation pipeline to extend these tests to real-world domains. From a memory perspective, these tasks expose whether a model can retain and compare evidence across separated observations, although the benchmark still scores the resulting reasoning answer rather than the underlying memory operations.

EngineMT-QA (Wang et al. 2025), TSQA from Time-MQA (Kong et al. 2025), and ChatTS (Xie et al. 2025) extend evaluation to temporal–textual question answering, numerical and open-ended questions, and multivariate reasoning. Collectively, they provide increasingly rich tasks in which intermediate evidence may need to remain available across several reasoning steps. What remains missing is an explicit intervention on memory itself, for example, controlling which earlier observation must be retained, when an intermediate conclusion should be stored, or whether the model can recover that evidence after a long reasoning trajectory.

Current results reveal an important distinction between long context and effective memory. ChatTS (Xie et al. 2025) represents explicit within-context memory, while TS-Reasoner (Ye et al. 2026) and TimeCopilot (Garza and Rosillo Garcia 2025) retain intermediate findings or tool outputs as working memory during multi-step analysis. Yet TS-Haystack (Zumarraga et al. 2026) shows that access to relevant evidence still deteriorates when informative events are buried within increasingly long temporal histories, whereas explicit retrieval remains substantially more reliable. Taken together, these approaches suggest that simply exposing a model to more history, or retaining intermediate reasoning state, does not guarantee that the required distant evidence remains recoverable. Most reasoning benchmarks still lack matched with-memory and without-memory variants, so the specific contribution of persistent memory remains difficult to isolate.

10.5 Agentic Evaluation

Agentic evaluation is substantially less mature because most existing systems are still judged by the quality of their final downstream prediction. MAFS (Huang et al. 2025), for example, evaluates a multi-agent forecasting system across multiple conventional forecasting datasets; this establishes end-task effectiveness but does not isolate whether persistent memory contributes to the result.

TimeSeriesGym (Li et al. 2025) moves evaluation closer to the agent level. Rather than restricting the task to producing a forecast, it evaluates agents on practical time-series machine-learning engineering challenges involving data handling, repository understanding, code, models, and multiple generated artifacts. This broader

interaction horizon creates opportunities for an agent to reuse earlier findings and decisions, but persistent memory is still not an explicit evaluation target. Neither TimeSeriesGym (Li et al. 2025) nor current forecasting-agent benchmarks directly measure the central operations of Section 8: whether the controller writes a useful experience, updates it when evidence changes, recalls it in a later episode, or removes it when it becomes stale. The absence of an archival standardized benchmark for these operations is therefore itself a finding of this survey.

Memory-enabled forecasting agents such as MemCast (Tao et al. 2026), CastFlow (Pan et al. 2026), and Nexus (Das et al. 2026) provide representative episodic, procedural, and error-derived memory stores, and their end-task results suggest that reusing accumulated experience can improve agentic forecasting workflows. However, current agentic evaluations usually compare the complete system rather than an otherwise identical agent with and without persistent memory. MAFS (Huang et al. 2025), for example, shows clear gains from specialization and inter-agent information exchange, but does not isolate a persistent-memory effect. The current evidence therefore supports the effectiveness of memory-enabled agentic systems, but not yet the causal contribution of their write, recall, update, and forget operations.

10.6 Memory-Specific Evaluation

Standard task accuracy conflates the store, the access mechanism, and the predictor. A memory-specific benchmark should instead control what information is needed, where it is located, and how long it must remain recoverable.

TS-Haystack (Zumarraga et al. 2026) is the closest existing resource to this objective. It inserts known short events into longer sensor streams and evaluates direct retrieval, temporal reasoning, multi-step reasoning, and contextual anomaly tasks as context length increases. Because the location of the inserted event is controlled, performance can be interpreted as an ability to preserve and recover temporally localized information rather than only as aggregate task accuracy. Its reported comparison between compression and retrieval is particularly relevant to the memory spectrum: representations that remain sufficient for classification can still lose the local detail required to retrieve a specific event. TS-Haystack is currently a workshop benchmark, rather than a mature field standard.

SynTSBench (Tan et al. 2025) provides a complementary methodological precedent. Its programmable synthetic generation isolates temporal patterns, irregularities, and robustness factors, allowing model capabilities to be varied independently rather than inherited from a fixed real-world dataset. It is not designed specifically around memory, but the same controlled-generation principle could place informative events at known temporal distances or recurrence frequencies and thereby quantify how memory fidelity degrades.

Building on these directions, we recommend that time-series memory evaluation report at least four quantities separately from final task error. First, *retention* should measure whether a known informative event remains recoverable as its temporal distance from the query increases. Second, *retrieval quality* should measure whether the relevant record appears among the retrieved items, with random, no-retrieval, and, where possible, oracle controls. Third, *temporal validity* should test whether a model

rejects or down-weights memories from obsolete regimes. Fourth, *memory management* should score write, update, and forget decisions over repeated episodes under a fixed storage budget. These measurements would turn memory from an architectural label into an independently testable capability and connect directly to the effective temporal memory capacity discussed in Section 11.

10.7 Open-Source Resources

Benchmarking is also supported by mature software ecosystems. GluonTS provides reference implementations and evaluation tools for probabilistic forecasting and anomaly detection (Alexandrov et al. 2020); aeon unifies forecasting, classification, regression, clustering, similarity search, and related time-series utilities (Middlehurst et al. 2024); and Merlion provides forecasting and anomaly-detection pipelines with deployment-oriented evaluation for univariate and multivariate series (Bhatnagar et al. 2023). More specialized resources such as TFB, BasicTS+, LargeST, MM-TSFlib, and TSB-AD supply task-specific datasets and reproducible protocols. Complementing these general-purpose resources, the accompanying collection for this survey is organized by the memory spectrum (explicit, retrieval, and agentic) and records the stored representation, read/write semantics, target task, venue, and available code for each method.

11 Open Challenges and Future Directions

The memory spectrum presented in this survey suggests that the central challenge in time series modelling is not simply how to access longer histories, but how to determine *which parts of history should remain available, in what form, and at what cost*. Internal memory addresses long histories through compression, external memory through persistent storage and retrieval, and agentic memory further introduces explicit decisions about what to store, retrieve, update, and forget. We identify seven interconnected challenges that are likely to shape future time-series memory systems. These challenges can be summarized by seven guiding research questions, which provide an overview of the key issues discussed in the following subsections (Table 7).

11.1 Memory Allocation and Effective Temporal Capacity

Early neural memory research established a fundamental distinction between compressing history into internal model representations and retaining information in an explicit, addressable external store (Graves et al. 2014; Weston et al. 2014; Sukhbaatar et al. 2015). Complementary directions sought to extend usable history through temporal segment caching (Dai et al. 2019; Rae et al. 2020) or large-scale external retrieval (Guu et al. 2020; Borgeaud et al. 2022). However, directly porting these capacity-scaling paradigms from natural language to time series exposes a fundamental misalignment. In NLP, information density is relatively uniform; in time series, it is radically asymmetrical. A sensor stream may contain millions of routine observations, yet only a handful of critical causal precursors (e.g., an anomalous vibration prior to failure). Consequently, two systems with identical memory budgets can retain vastly

Table 7 Guiding research questions for future time-series memory systems.

Section	Direction	Guiding research question
11.1	Memory Allocation and Effective Temporal Capacity	*How can architectures selectively retain the most informative time-series data, and how should the effective capacity of this recoverable information be quantified?*
11.2	From Information Retrieval to Temporal Retrieval	*What makes a historical experience useful beyond surface-level morphological similarity?*
11.3	Memory Validity under Distribution Shift	*How should time-series architectures continuously assess whether stored knowledge remains valid under changing conditions?*
11.4	Hierarchical Multimodal Memory	*How should time-series architectures organize multimodal information across memory representations while preserving complementary information across modalities?*
11.5	Beyond LLM-Centric Agentic Systems	*How should heterogeneous memory representations be coordinated?*
11.6	Memory from Privileged Information	*Can privileged information during training shape memory under inference-time constraints?*
11.7	Causal Memory for Temporal Reasoning	*How can temporal memory preserve causal structure that enables models to simulate how a system evolved under different interventions?*

different predictive capabilities if they do not carefully manage *what* they choose to remember.

As highlighted by continual learning research, memory is not merely a storage problem, but a "selection problem" (Rolnick et al. 2019; Parisi et al. 2019). Maximizing raw storage capacity introduces redundancy, increases retrieval cost, and obscures critical experiences beneath routine noise—much like human cognition, which preferentially consolidates salient events over highly predictable background stimuli (LaBar and Cabeza 2006; Kensinger and Ford 2020). Because the value of a time-series observation may only become apparent months after it occurs, the goal of temporal memory is to determine the *future predictive utility* of an event at the time of observation. This is especially relevant in long-horizon forecasting, where autoregressive rollout compounds errors step by step, so small early inaccuracies amplify over the horizon (Nguyen et al. 2026). Retaining the right distant precursors, not just recent forecasts, is crucial for preventing long-horizon predictions from drifting.

Resolving this requires architectures that actively counteract temporal smoothing by decoupling the memory write rate from the data sampling rate. Rather than uniformly compressing histories at fixed intervals, future time-series memory must become *event-driven* and *information-aware*. A promising solution is a two-track memory system. Routine, predictable patterns (like normal daily seasonality) can be heavily compressed into the model's standard hidden states. However, unexpected events, detected when the model makes a large prediction error or sees a sudden shift in the data, should be routed to a separate, permanent external memory. By triggering

memory updates based on "surprise" rather than just the passage of time, the system can ensure that rare but critical clues are kept intact for when they are needed again.

To validate such systems, we argue for evaluating *effective temporal memory capacity*, which measures how much temporally distant information a model can selectively retain, recover, and use to condition future predictions. Effective temporal memory capacity should be assessed by measuring how reliably task-relevant information remains recoverable as its temporal distance from the prediction target increases. While recent evaluation suites (discussed further in Section 10) have made significant strides in stress-testing temporal capabilities—such as BasicTS+ (Shao et al. 2025) and LargeST (Liu et al. 2023) for scaling cross-variable and spatiotemporal dependencies, or SynTSBench (Tan et al. 2025) and TS-Haystack (Zumarraga et al. 2026) for controlled long-context retrieval—they still exhibit critical gaps. Specifically, these benchmarks largely evaluate retrieval from predefined, static context windows rather than testing a model's ability to autonomously manage an evolving memory constraint over unbounded streams. They do not explicitly evaluate the dynamic *selection* problem. Future benchmarks must evolve to place informative causal precursors at highly variable temporal distances from the target window across continuous data streams. Measuring whether those specific events remain recoverable will provide a principled way to compare latent states, compressed memories, and persistent external stores, answering two core questions: *First, by what mechanisms should an architecture selectively retain the most informative events from time-series histories? Second, how should we benchmark the effective capacity of the historical information that remains recoverable?*

11.2 From information retrieval to temporal retrieval.

Retrieval-augmented time series forecasting has begun to move beyond simply extending the input context by allowing models to access relevant historical experiences. Initial efforts in retrieval-augmented forecasting, such as RATD (Liu et al. 2024) and RAFT (Han et al. 2025), demonstrated the utility of fetching external historical trajectories based primarily on time-domain similarity. However, as recently highlighted by Nguyen et al. (2026a), anchoring retrieval strictly in the time domain exposes critical structural vulnerabilities. Firstly, time-domain distance metrics inherently ignore how energy is distributed across frequency bands, causing models to severely misidentify underlying periodic patterns when phase shifts occur (Nguyen et al. 2026a). Furthermore, these retrieval mechanisms often suffer from a lack of temporal recency weighting. All historical observations are treated equally, ignoring the reality that under non-stationary regime shifts, recent data often carries substantially stronger predictive power than distant matches. Table 5 shows that retrieval-based memory is particularly well represented in forecasting, while comparatively fewer approaches have explored retrieval for classification, anomaly detection, and imputation, and retrieval-based memory remains largely unexplored for decision-making.

While incorporating spectral and recency constraints represents a necessary progression toward dynamics-aware retrieval, a deeper mechanical bottleneck remains. Retrieval paradigms, even when operating across both time and frequency domains, still rely fundamentally on *morphological similarity* (e.g., Euclidean distance or cosine

similarity in latent spaces). Morphological similarity is not necessarily equivalent to predictive relevance. Two historical sequences may exhibit virtually identical electrical demand profiles, yet possess completely divergent subsequent dynamics due to unobserved confounders (e.g., a standard summer day versus a latent heatwave). Conversely, a true historical causal precursor may exhibit high phase misalignment or amplitude distortion relative to the current window, causing it to be erroneously discarded by strictly symmetric matching criteria.

Beyond finding the right data, there is a bottleneck in how retrieved memories are actually integrated into the model. Current systems treat retrieval as an isolated step and lack a principled way to blend the retrieved historical data with current observations. Simply stitching a non-adjacent retrieved sequence next to the current one creates unnatural structural jumps at the boundaries, disrupting the continuous temporal manifold (Nie et al. 2023; Zhang and Yan 2023). On the other hand, standard attention provides a flexible mechanism for blending the two sources, but may also dilute distinctive temporal patterns when retrieved and current signals are combined indiscriminately. For retrieved memories, the broader challenge is therefore to integrate historical evidence without washing out the distinctive temporal patterns that made the memory useful in the first place.

Moving forward, time-series retrieval methods must transition from purely *similarity-based matching* towards *task- and dynamics-aware temporal retrieval*. A promising solution involves learning asymmetric similarity functions, where queries and keys are explicitly optimized to match current observations with historical precursors that led to similar *future* outcomes, rather than just similar pasts. Furthermore, to resolve the fusion bottleneck (Nguyen et al. 2026b), these retrieved trajectories should be integrated via uncertainty-aware gating mechanisms that dynamically adjust for phase, scale, and the specific downstream task (e.g., forecasting vs. anomaly detection). Ultimately, future architectures must move beyond surface-level similarity to answer a single core question: *What makes a historical experience useful beyond surface-level morphological similarity?*

11.3 Memory Validity under Distribution Shift

Most memory systems treat stored information as persistent evidence that can be reused whenever it appears relevant. However, as the underlying data-generating process evolves, the relationship between past observations and future outcomes may change. A historical pattern can therefore remain retrievable while becoming unreliable for the current process. The central challenge is consequently not simply to retrieve relevant history, but to determine whether that history remains trustworthy under the current conditions.

This problem is particularly acute in time series, where changes in consumption behaviour, infrastructure, operating conditions, or component ageing can alter the underlying dynamics. Historical observations may therefore remain useful, become outdated, or become useful again when similar conditions return. Recent time-series models increasingly address this problem through drift-aware adaptation, state disentanglement, and retention or retrieval of historical concepts (Cai et al. 2025; Chen et al. 2025; Zhan et al. 2025). However, these approaches do not explicitly formulate

memory maintenance as a continuous assessment of predictive validity that determines whether a stored pattern should remain active, be temporarily suppressed, reactivated, or permanently retired.

Addressing this challenge requires reconceptualizing memory as a dynamic system with mechanisms for continuous validation and controlled decay. Such a system must be condition-aware, distinguishing between *permanent structural breaks* and *transient regime shifts*, because a loss of reliability does not necessarily mean that a memory has become obsolete; it may instead reflect a temporary departure from the regime in which that memory is useful. If a process undergoes a fundamental shift, such as a factory replacing a core component, the memory system should progressively discount, replace, or retire memories associated with the obsolete regime. Conversely, if the shift is cyclical or transient, such as a seasonal change or temporary economic shock, the relevant memories should not be destroyed, but temporarily deactivated until similar conditions return.

Such condition-aware memory would complement temporal retrieval with a second mechanism for assessing whether retrieved knowledge remains reliable under the current regime. A memory would therefore not be reused solely because it resembles the present context; its continued use would depend on whether the knowledge it encodes remains consistent with ongoing observations and outcomes. This creates a memory lifecycle in which knowledge can remain active, become uncertain and temporarily suppressed, recover when its reliability is restored, or be retired when evidence indicates a persistent structural break. The challenge is therefore to move from passive storage to active memory maintenance, raising a critical question: *How should time-series architectures continuously assess whether stored knowledge remains valid under changing conditions, and determine when memories should remain active?*

11.4 Hierarchical Multimodal Memory

As discussed throughout this survey, memory mechanisms differ in how they represent, retain, and access historical information. Internal states provide efficient access to recent or recurring information, compressed representations extend temporal coverage at reduced fidelity, while external memory enables persistent retrieval of selected experiences. Recent time-series foundation models further broaden this design space by incorporating information beyond the numerical time series itself (Das et al. 2024; Xie et al. 2025; Shi et al. 2025). For example, ChatTS maps time-series patches and text tokens into a shared embedding space through a lightweight projection layer before jointly processing them with an LLM (Xie et al. 2025), while Time-VLM separately encodes temporal, visual, and textual information and then fuses these representations through cross-modal attention (Zhong et al. 2025). These studies demonstrate that heterogeneous information can be jointly represented and accessed, but largely treat the representation and storage configuration as fixed design choices rather than a problem of adaptive memory allocation. The distribution in Table 5 similarly shows that existing memory mechanisms are primarily distinguished by how information is stored and accessed, while the organization of heterogeneous information across these mechanisms remains largely unaddressed.

The challenge is to organize heterogeneous modalities across a hierarchy of memory representations. Different modalities provide complementary information at different temporal scales and levels of abstraction: continuous sensor streams may capture fine-grained dynamics that are efficiently represented in compact latent states, while textual or visual observations may preserve semantic information that is difficult to recover from the time series alone. A hierarchical memory could therefore maintain information at multiple levels, from shared low-dimensional representations for common information to modality-specific high-fidelity representations for complementary details. The key difficulty is determining when modalities should be jointly encoded, separately preserved, or promoted to richer representations while avoiding redundant storage and unnecessary memory cost.

An important research question is *how should time-series architectures dynamically organize multimodal information across shared and modality-specific memory representations while preserving complementary information across modalities?* Such a framework would move beyond simply combining multimodal inputs toward learning how heterogeneous information should be represented at different levels of abstraction and when it should be shared or preserved separately.

11.5 Beyond LLM-Centric Agentic Systems

Recent agentic time-series systems increasingly use LLMs as the central controller for reasoning, memory management, and tool use (see Section 8). These systems demonstrate that language models can organize historical information (Tao et al. 2026,?), generate summaries of complex workflows (Zhang et al. 2025; Zhao et al. 2025), retrieve relevant experiences (Ravuru et al. 2024; Tao et al. 2026), and coordinate multiple sources of information (Das et al. 2026). Table Table 5 reveals a strong concentration of agentic memory around LLM-centric architectures, particularly in forecasting, reasoning, and decision-making. This concentration suggests that the emerging agentic paradigm has largely inherited language-model-centric architectures rather than developing memory and control mechanisms specialized for continuous numerical dynamics. While effective for semantic reasoning and instruction following, positioning an LLM as the universal cognitive engine creates a fundamental architectural bottleneck when managing complex, continuous temporal dynamics.

In adjacent AI domains, early agent architectures were predominantly language-centric, but researchers quickly realized that forcing all perception through a textual bottleneck degrades performance. Consequently, recent systems natively incorporate visual and other modalities to capture complex information, a shift clearly demonstrated by the emergence of Vision-Language Models (VLMs). Time-series agents have reached a parallel inflection point. An LLM is primarily trained to model language, whereas time series data are numerical, continuous, high-dimensional, and governed by precise temporal relationships. Converting a long sensor stream into textual descriptions, or asking a language model to reason directly over numerical values, may therefore introduce information loss. At the same time, textual information can provide contextual information that is difficult to infer from the numerical signal alone.

A promising future direction is therefore to move towards heterogeneous multi-model agent architectures, where the agent's perception, memory, and control loop

are co-governed by multiple specialized foundation models rather than centralized in a single LLM. Numerical encoders could preserve fine-grained temporal patterns, language models could represent semantic events and domain knowledge, and multimodal models could connect temporal observations with text, images, and other contextual information. This leads to an open question of *how should heterogeneous memory representations be coordinated so that language-based reasoning complements the numerical and temporal structure of the underlying data?*

11.6 Memory from Privileged Information

Another promising direction for time-series memory is to use privileged information. Under the Learning Using Privileged Information (LUPI) paradigm, models exploit information available during training but unavailable at inference, often transferring this knowledge through teacher–student learning or generalized distillation (Vapnik and Izmailov 2015). This setting is particularly natural for time series, where offline training provides richer temporal context than is available to an online model. For example, Karlsson et al. (Karlsson et al. 2022) study time-series privileged information in which intermediate observations between the prediction time and the target outcome are available during training but not at test time. Similarly, recent privileged knowledge distillation approaches have explored richer representations for multivariate time-series forecasting (Liu et al. 2025). However, these methods primarily use privileged information to improve prediction, while its potential for directly supervising *memory formation* remains largely unexplored.

The key challenge is that a memory mechanism must decide what to retain before knowing what will matter. A bounded-memory model must continuously determine which historical information to store, compress, or discard, even though the importance of an observation may only become apparent much later. Privileged temporal context provides a natural training signal for this problem. A teacher with access to the complete trajectory can identify which parts of the observed history were important for subsequent dynamics and transfer this knowledge to a causal student that operates only on information available at inference.

Such privileged memory distillation could supervise what a memory system retains, compresses, overwrites, and retrieves. A privileged teacher could use future observations, complete trajectories, additional sensor channels, delayed labels, or expert annotations to identify which historical information is most relevant, while a causal student learns to reproduce this memory using only information available at inference. This is particularly valuable in long-horizon and streaming settings, where limited memory capacity requires selective retention. The central question is therefore: *Can privileged information during training shape memory under inference-time constraints?*

11.7 Causal Memory for Temporal Reasoning

Internal memory offers an efficient means of compressing long temporal histories, but existing memory mechanisms are largely optimized for predictive accuracy rather than reasoning over the dynamics they encode. Meanwhile, causal time-series research

has developed methods for identifying temporal dependencies and causal relationships (Assaad et al. 2022; Gong et al. 2024), with recent work incorporating causal representations into forecasting (Cai et al. 2025). Yet causal structure is typically treated as a means to improve prediction, rather than as information that should be explicitly retained within memory for downstream reasoning. Consequently, a latent memory may capture recurring associations between events without preserving the mechanisms needed to explain why they occur or how they would respond to interventions.

Integrating causal structure into temporal memory could instead enable models to reason over past experience. A causal memory could retain temporal dependencies, delayed effects, regime-specific mechanisms, and intervention outcomes, allowing an agent to reason about the factors underlying an observed trajectory. This is particularly important for counterfactual reasoning, where the goal is not simply to predict what will happen, but to infer how a trajectory would have evolved under an alternative intervention. For example, rather than remembering only that increased load preceded a temperature rise, a causal memory could retain the relationship between load, temperature, and system state, enabling the model to reason about the consequences of reducing the load.

The key challenge is therefore to make causal structure a *first-class component of temporal memory* that can be updated, retrieved, and reasoned over, rather than implicitly compressed into a predictive state. This leads to a fundamental question: *How can temporal memory preserve and update causal mechanisms to support reasoning about why events occurred and how trajectories would change under alternative interventions?*

12 Conclusion

We recast a decade of time-series methods through a unified spectrum of memory, spanning implicit parametric state, explicit and compressed representations, persistent retrieval, and agent-curated stores. This perspective reveals a continuity often obscured by architecture-specific accounts, where different approaches vary not only in how much history they retain, but also in how historical information is represented, accessed, updated, and forgotten. It also makes several gaps in time-series memory research more visible. Long-term retrieval remains concentrated in forecasting, memory-specific evaluation is still limited, and agentic memory for time series remains at an early stage. More broadly, the survey suggests that the central challenge is shifting from simply extending temporal context towards learning what historical information is worth retaining, how it should be represented, where it should reside, and when it should still be trusted. We hope that the memory spectrum, taxonomy, and accompanying online repository provide a durable framework for organizing this emerging literature and for situating future methods by the role that memory plays in temporal learning rather than by architectural era.

Acknowledgements. The authors thank colleagues at the Applied Artificial Intelligence Initiative (A^2I^2), Deakin University, for helpful discussions.

Declarations

Funding

This research was funded (partially or fully) by the Australian Government through the Australian Research Council. Dr Hung Le is the recipient of an Australian Research Council Discovery Early Career Researcher Award (project number DE250100355) funded by the Australian Government.

Conflict of interest

The authors declare no competing interests relevant to the content of this article.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Data availability

This survey does not generate new experimental data. The curated collection of papers reviewed in this work, together with their categorisation, is publicly available and maintained at https://github.com/DA2I2-SLM/Time-series-memory.

Materials availability

Not applicable.

Code availability

No custom code was developed for this study.

Author contributions

HL contributed to conceptualization, analysis, and writing of the original draft. MHN, HHN, MN, and VDD contributed to the literature review and manuscript preparation. DN contributed to supervision, critical review, and editing. All authors read and approved the final manuscript.

Appendix A Glossary

This glossary collects the time-series-memory terminology introduced or used consistently throughout the survey, together with a small number of foundational concepts the survey builds on. Full treatments, formal definitions, and citations are given in the referenced sections; entries here are deliberately short.

Agentic memory (agentic curated memory). An external store $\mathcal{M}$ over which a controller π exercises a *write*, *read*, and *forget* policy, so the store is actively revised in light of new interactions rather than fixed once built. Explicit in representation, policy-bounded in capacity, controller-issued in read/write, and online in persistence (Section 8).

Auditability. The property that a prediction can be traced back to the specific stored episode, prototype, or record that produced it, e.g., via the read weights of Equation (18). Cited as a benefit of explicit, retrieval, and agentic memory over internal memory (Section 3).

Capacity (axis). One of the four axes for comparing memory mechanisms: *bounded* capacity is fixed in dimension independent of sequence length (e.g., a hidden state or a fixed slot bank); *unbounded* capacity grows with the data available, as in a retrieval index (Section 2.4).

Causal memory. A proposed extension in which temporal memory retains causal structure, i.e., dependencies, delayed effects, and intervention outcomes, rather than only associations, so that stored experience can support counterfactual reasoning (Section 11.7).

Controller. The component (rule-based, learned, or an LLM-based agent) that issues the write, read, and forget operations over an agentic memory store, denoted π in Equation (25) (Section 8.1).

Curated retrieval. An intermediate regime, between passive retrieval and fully online agentic memory, in which a store's contents are chosen or distilled by an agent but then remain read-only at deployment; most surveyed "agentic" time-series systems fall here rather than in the fully online regime (Section 8.3).

Effective temporal memory capacity. A proposed evaluation notion: the degree to which actionable, temporally distant information remains recoverable and usable to condition future predictions, characterized by how retrieval fidelity degrades with temporal distance (Section 11.1).

Episodic memory. An agentic memory type that records specific past events, e.g., a context/forecast/error triple from a single interaction; typically written per instance and read at prediction or reflection time, with fast decay under regime change (Table 3).

Evolution operator ($\mathcal{E}$). The agentic-memory operator that consolidates, revises, and evicts stored content after admission. It is the identity in retrieval memory, where the store is built once and thereafter only read, and is non-trivial only where a forget policy actually runs (Section 7, Section 8.1, Section 8.3).

Explicit memory. A store of individually addressable entries (slots, vectors, or records), each written and read as a unit rather than entangled in weights or a dense state; subdivided into verbatim, slot, and learned memory (Section 6).

External memory. Historical information maintained outside the model's latent state in a separately addressable store that the model can read from and, depending on the mechanism, write to. External memory can retain information beyond the model's fixed internal capacity and may incorporate content not seen during training. It encompasses explicit, retrieval, and agentic memory, whose capacity and ability to incorporate new content vary by subclass (Section 2).

Forget policy. The evolution operator $\mathcal{E}$ acts on content already stored in memory, determining what should be consolidated, revised, demoted, or discarded, e.g., when knowledge becomes stale under a regime shift. Together, $\mathcal{F}$ and $\mathcal{E}$ form the write operator $\mathcal{W}_\phi$ in

Equation (1), with $\mathcal{F}$ operating on newly generated content and $\mathcal{E}$ on existing memory (Section 8.1).

Foundation model (time-series foundation model, TSFM). A model pretrained on a broad mixture of time-series data and applied across datasets, frequencies, and domains, often zero-shot; its verbatim attention cache and its use as a retrieval backbone recur throughout the survey, and it raises the confound that an external memory's apparent gain may already be encoded parametrically during pretraining (Section 6.2.1, Section 10).

Instance retrieval memory. Retrieval memory that retains concrete historical cases, i.e., windows, trajectories, or context–future pairs, so a query can be grounded in specific observed precedents (Section 7.1.1).

Internal memory. Historical information encoded in dynamically updated, fixed-dimensional latent states rather than retained as separate, addressable entries. Learned parameters govern how these states are updated but do not themselves constitute memory. Internal memory is subdivided into recurrent-state and structured-state memory (Section 5).

Knowledge retrieval memory. Retrieval memory that stores information other than another realization of the target series, e.g., text, exogenous events, or structured domain knowledge, providing context about the underlying process rather than a trajectory to imitate (Section 7.1.3).

Latent retrieval memory. Retrieval memory whose persistently stored and retrieved object is itself a compact derived representation (an embedding, latent state, or learned token) rather than the original observation (Section 7.1.2).

Learned memory. Explicit memory that stores compact representations learned from data, such as prototypes, centroids, or shared pattern banks, and explains a query as a mixture of a few such entries rather than many concrete episodes (Section 6.2.3).

Learning Using Privileged Information (LUPI). A foundational paradigm in which a model exploits information available during training but unavailable at inference (e.g., via teacher–student distillation); proposed as a training signal for supervising what a time-series memory system should retain, compress, or discard (Section 11.6).

Long-horizon dependence (P1). The problem setting in which useful predictive information lies far outside any feasible input window, e.g., a value tied to the previous year's seasonal peak (Section 3.1).

Lookback window. The immediate span of recent observations, $\mathbf{x}_{t-L+1:t}$, that a model conditions on directly; memory is defined relative to this window as information the model uses that is *not* present in it (Section 2).

Memory (of a model). The memory of a time-series model refers to any mechanism that makes historical information available to the predictor beyond the information contained in its current prediction target.(Definition 1, Section 2).

Memory management (evaluation measure). A proposed benchmark quantity scoring the quality of write, update, and forget decisions over repeated episodes under a fixed storage budget (Section 10).

Nearest-neighbor search. A hard (arg max) form of content-based addressing that retrieves the single closest stored entry to a query rather than a weighted mixture; the sharpened limit of Equation (18) (Section 6).

Non-stationarity / regime shift / drift (P3). The problem setting in which the data-generating process changes over time, so historical data can become uninformative or actively misleading; the central motivation for forget/update policies in agentic memory (Section 3.3).

Persistence (axis). One of the four axes for comparing memory mechanisms, describing how long stored information survives: *per-sequence* (resets with each new input), *per-dataset* (a fixed store or index, frozen after training/construction), or *online* (continues to evolve across an interaction) (Section 2.4).

Procedural memory. An agentic memory type that stores knowledge of how to perform a task, e.g., which tool or retrieval strategy to use in a given situation; typically written once evidence accrues and decays slowly, since it can outlive individual episodic entries (Table 3).

Read/write mechanism (axis). One of the four axes for comparing memory mechanisms, describing how information enters and leaves a store: gated recurrence, structured linear recurrence, content attention, nearest-neighbor search, or controller-issued operations (Section 2.4).

Recurrent-state memory. Internal memory in which a hidden state $\mathbf{h}_t$ is updated at each time step from the previous state and the current observation (e.g., RNN, LSTM, GRU), so it depends on the entire history $\mathbf{x}_{1:t}$ while remaining fixed in dimension (Section 5.2.1).

Recurring patterns (P2). The problem setting in which current conditions resemble a past event whose outcome carries predictive value, e.g., matching a demand spike to a similar one the previous year; internal state compression blends past episodes and is ill-suited to it, motivating explicit, addressable memory (Section 3.2).

Representation (axis). One of the four axes for comparing memory mechanisms: *implicit* representation is entangled in weights or a dense state; *explicit* representation consists of identifiable slots, vectors, or records (Section 2.4).

Retrieval-augmented generation (RAG). The foundational (originally NLP) paradigm of conditioning a model's output on documents retrieved from an external corpus; the conceptual ancestor of retrieval-augmented memory for time series (Section 4.1).

Retrieval-augmented memory. Memory that retains information in an external store and selectively recalls a query-dependent subset (top-K by similarity) to condition the current prediction; capacity is unbounded and grows independently of the predictor. Subdivided into instance, latent, and knowledge retrieval memory (Section 7).

Retrieval quality (evaluation measure). A proposed benchmark quantity measuring whether the relevant record appears among the retrieved items, assessed against random, no-retrieval, and (where possible) oracle controls (Section 10).

Semantic memory. An agentic memory type that stores generalized knowledge distilled from past observations, e.g., recurring regimes, calendar rules, or exogenous facts, rather than specific episodes; written once evidence accrues, with medium decay (Table 3).

Slot collapse. A failure mode of explicit memory in which multiple slots converge on similar content, leaving an effective capacity far below the nominal number of slots N (Section 6.3).

Slot memory. Explicit memory built from dedicated, learned-addressing locations (descended from the Neural Turing Machine and Differentiable Neural Computer) whose contents are model states rather than raw observations (Section 6.2.2).

Sparse and hierarchical data (P4). The problem setting in which an individual series lacks sufficient local history (e.g., a new product or newly deployed sensor) and must draw on related series or higher levels of aggregation (Section 3.4).

Structured-state memory (state-space model, SSM). Internal memory in which a fixed-dimensional hidden state evolves through parameterized, often linear, state transitions (e.g., S4, Mamba) rather than nonlinear gating, admitting a closed-form compression map (Equation (17)) and, in selective variants, input-dependent transitions (Section 5.2.2).

Temporal validity (evaluation measure). A proposed benchmark quantity testing whether a model appropriately rejects or down-weights memories originating from an obsolete regime (Section 10).

Verbatim memory. Explicit memory that gives every observed element its own entry, encoded individually and never merged or summarized into a coarser state, so forgetting occurs only through truncation; exemplified by the attention context of a Transformer (Section 6.2.1).

Working memory. Non-persistent, within-episode state that an agent accumulates during a single analysis (intermediate evidence, tool outputs, partial conclusions) and discards once the episode ends, distinguishing it from the three persistent agentic memory types (Section 8.2).

Write policy. The agentic-memory policy $\mathcal{F}$ that determines which artifacts of an interaction are persisted to the store and in what abstracted form (Section 8.1).

Appendix B Search and Selection Protocol

This section documents how the surveyed corpus was assembled. We report the time window (Section B.1), the study selection and screening criteria (Section B.2), the search terms (Section B.3), the inclusion and exclusion criteria (Section B.4), the resulting composition of the corpus (Section B.5) and the threats to validity (Section B.6), so that the coverage claims in Table 1 can be independently assessed.

B.1 Scope and Time Window

The primary search window is 2015–2026, matching the temporal axis of Figure 1. The lower bound is set at 2015 because the memory mechanisms this survey is organized around, namely explicitly addressable stores, retrieval augmentation, and agentically curated stores, only begin to appear in the time-series literature after the neural external-memory architectures of 2014–2016. The search was last refreshed on **04/09/2026**.

Four works published before this window are retained as *foundational exceptions*: the recurrent formulation (Elman 1990), gated recurrence (Hochreiter and Schmidhuber 1997; Cho et al. 2014), and the Neural Turing Machine (Graves et al. 2014). These are included not as surveyed time-series methods but because the mechanisms reviewed in Section 5 and Section 6 are direct descendants of them, and the memory lineage is not intelligible without them.

B.2 Study Selection and Screening Procedure

The study-selection process consisted of record identification, deduplication, title and abstract screening, and full-text eligibility assessment. The inclusion and exclusion criteria in

Section B.4 were applied to determine whether a candidate study contributed an identifiable memory mechanism relevant to deep time-series modeling.

Deduplication.

Records retrieved from different sources were consolidated into a single screening collection. Duplicate records were identified by comparing bibliographic metadata, including titles, DOIs, arXiv identifiers, and author and publication year where available. Potential matches that could not be resolved through metadata comparison were manually inspected to determine whether they referred to the same work. When multiple versions of the same work were available, we applied the version-selection rule in Table B2 and retained the version that best represented the final published work, preferring the peer-reviewed version of record over preprint versions when available.

Title and abstract screening.

The deduplicated records were screened against the inclusion and exclusion criteria using titles and abstracts. Records were excluded at this stage when they clearly fell outside the scope of the review, including studies that did not address memory mechanisms in time series, did not describe a relevant method or architecture, or focused on applications unrelated to the review's research questions. Records that were clearly non-research publications or otherwise failed to meet the predefined inclusion criteria were also excluded. Records for which eligibility could not be determined reliably from the title and abstract were retained for full-text assessment.

Full-text screening.

The full texts of the remaining records were assessed against all inclusion and exclusion criteria. Particular attention was paid to whether the study retained historical information beyond the immediate prediction context and whether its memory mechanism could be characterized in terms of representation, writing, access, and persistence.

B.3 Sources and Search Terms

We searched IEEE Xplore, ACM Digital Library, OpenReview, arXiv, and Google Schola and also manually reviewed the proceedings of NeurIPS, ICML, ICLR, KDD, IJCAI, AAAI, and ICDM, as well as relevant journals represented in our reference set, including IEEE TPAMI, TMLR, International Journal of Forecasting, Journal of Forecasting, Neurocomputing, and Engineering Applications of AI. Because similar memory mechanisms are often described using different terms—for example, a memory bank" in one paper may be called a prototype set" or "datastore" in another—we did not rely on keyword searches alone. We complemented the initial search with backward snowballing, by examining the references of included papers, and forward snowballing, by following later works that cite the key papers in each memory category.

Table B1 lists the query terms. Each mechanism term was conjoined with at least one time-series term.

B.4 Inclusion and Exclusion Criteria

The inclusion and exclusion criteria are summarized in Table B2. At the full-text stage, we did not require a paper to explicitly use the term *memory*. Instead, we asked whether its method could be described using the four axes in Section 2.4: what information is retained,

Table B1 Search terms. Each mechanism term (right) was conjoined with at least one domain term (top block) and, where a class-specific vocabulary exists, with its class terms. Terms are lower-cased and matched on title, abstract, and keywords.

Group	Terms
Domain (required)	"time series", "temporal", "forecasting", "spatiotemporal", "sequential data", "multivariate series", "anomaly detection", "imputation", "time-series classification"
Cross-cutting memory	"memory", "memory-augmented", "external memory", "memory network", "memory module", "memory bank", "long-term dependency", "long-range dependence"
Internal	"recurrent", "LSTM", "GRU", "hidden state", "state space model", "structured state", "selective state", "Mamba", "linear recurrence"
Explicit	"slot memory", "memory slot", "addressable memory", "neural Turing machine", "differentiable neural computer", "prototype", "pattern bank", "shapelet", "codebook", "Hopfield", "key-value memory"
Retrieval	"retrieval-augmented", "retrieval augmentation", "RAG", "nearest neighbour retrieval", "datastore", "exemplar retrieval", "analogue retrieval", "retrieve and refine", "in-context examples"
Agentic	"agent", "agentic", "LLM agent", "tool use", "episodic memory", "semantic memory", "procedural memory", "reflection", "experience replay buffer", "self-improving"
Evaluation	"benchmark", "long-context", "needle in a haystack", "retrieval benchmark", "memory evaluation"

how it is written, how it is retrieved, and how long it remains available. Papers for which these aspects could not be clearly identified were considered outside the scope of the survey.

B.5 Composition of the Final Corpus

The final corpus comprised **118** eligible studies published between 2015 and 2026, including 85 peer-reviewed publications and 33 preprints admitted under the criteria in Table B2.

To characterize the coverage of the review, we classified the included studies according to their principal memory architecture, primary task and publication status. Each study was assigned a primary memory category according to the taxonomy in section 4.

Table B3 summarizes the resulting corpus. These counts describe the composition of the reviewed literature; they should not be interpreted as estimates of the relative effectiveness of the memory architectures, because the included studies differ in their tasks, datasets, experimental settings, and evaluation protocols.

B.6 Threats to Validity

Several limitations affect the coverage and reproducibility of this review.

Table B2 Inclusion and exclusion criteria. IC = inclusion, EC = exclusion. A work was retained only if it satisfied every inclusion criterion and triggered no exclusion criterion.

ID	Criterion
IC1	**Temporal data.** The method operates on time series, event sequences, or spatiotemporal streams as its primary modality.
IC2	**Identifiable memory mechanism.** The work can be placed on all four axes of Section 2.4: what is retained, how it is written, how it is read, and how long it persists.
IC3	**Memory is load-bearing.** Retention beyond the immediate input window is a stated mechanism of the contribution, not an incidental property of the backbone.
IC4	**Task relevance.** The work targets forecasting, classification, anomaly detection, imputation, or the emerging reasoning and decision-making settings of Section 9.
IC5	**Retrievable and in English.** Full text available; peer-reviewed, or a preprint meeting EC3.
EC1	**Non-temporal modality only.** Memory architectures evaluated solely on text, static vision, or tabular data, unless they are an anchor mechanism a surveyed time-series method descends from (Section B.1).
EC2	**Classical methods.** Statistical and non-deep approaches (ARIMA, exponential smoothing, classical nearest-neighbour forecasting) that predate the memory framing.
EC3	**Unvetted preprints.** Preprints were admitted only where they are the sole source for an emerging mechanism, and are marked as such; this applies chiefly to the agentic and 2026 retrieval literature, where archival versions do not yet exist.
EC4	**Superseded versions.** Where a workshop paper, preprint, and archival version describe the same system, only the most mature version is counted.

Search coverage.

The literature search used IEEE Xplore, ACM Digital Library, OpenReview, arXiv, and Google Scholar, supplemented by manual searches of relevant publication venues and citation-based snowballing. These complementary methods helped identify studies whose terminology differed from the search vocabulary. However, they do not guarantee complete coverage of the literature. In particular, Scopus and Web of Science were not searched as dedicated bibliographic databases, and some relevant studies may therefore have been missed. The review may also be affected by the coverage of the selected electronic sources, the search terms used, and the publication and indexing status of studies available at the time of the search.

Terminological variation.

Memory-related mechanisms are described using inconsistent terminology across time-series research. Relevant methods may therefore be missed when their titles, abstracts, or keywords do not contain the search terms listed in Table 8. Backward and forward snowballing were used to mitigate this risk, but cannot eliminate it.

Table B3 Composition of the final review corpus.

Category	Number of studies
Primary memory architecture	
Internal memory	29
Explicit memory	41
Retrieval memory	25
Agentic memory	23
Primary task	
Forecasting	54
Classification	12
Anomaly detection	23
Imputation	8
Reasoning	13
Decision-making	8
Publication status	
Peer-reviewed publications	85
Preprints	33

Publication and recency bias.

The corpus includes literature published during 2015–2026, including recent work whose archival publication status may not yet be settled. Consequently, the coverage of emerging approaches, particularly agentic memory, should be interpreted as a snapshot of the literature available at the final search date rather than an exhaustive account of subsequent developments.

Selection and classification judgement.

Applying the inclusion criteria and assigning memory categories require judgement when methods combine multiple mechanisms or fall near the boundaries of the proposed taxonomy.

These limitations should be considered when interpreting the corpus composition, architectural comparisons, and research gaps identified in the main paper.

References

Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D.C., Rangapuram, S., Salinas, D., Schulz, J., Stella, L., Türkmen, A.C., Wang, Y.: GluonTS: Probabilistic and neural time series modeling in Python. Journal of Machine Learning Research **21**(116), 1–6 (2020)

Ahamed, M.A., Cheng, Q.: TimeMachine: A time series is worth 4 Mambas for long-term forecasting. arXiv preprint arXiv:2403.09898 (2024) https://doi.org/10.48550/arXiv.2403.09898

Assaad, C.K., Devijver, E., Gaussier, E.: Survey and evaluation of causal discovery methods for time series. Journal of Artificial Intelligence Research **73**, 767–819 (2022) https://doi.org/10.1613/jair.1.13428

Alnegheimish, S., He, Z., Reimherr, M., Chandrayan, A., Pradhan, A., D'Angelo, L.: M^2AD: Multi-sensor multi-system anomaly detection through global scoring and calibrated thresholding. In: Proceedings of the 28th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 258, pp. 4384–4392 (2025)

Ansari, A.F., Stella, L., Türkmen, A.C., Zhang, X., Mercado, P., Shen, H., Shchur, O., Rangapuram, S.S., Arango, S.P., Kapoor, S., Zschiegner, J., Maddix, D.C., Wang, H., Mahoney, M.W., Torkkola, K., Wilson, A.G., Bohlke-Schneider, M., Wang, B.: Chronos: Learning the language of time series. Transactions on Machine Learning Research (2024)

Bhatnagar, A., Kassianik, P., Liu, C., Lan, T., Yang, W., Cassius, R., Sahoo, D., Arpit, D., Subramanian, S., Woo, G., Saha, A., Jagota, A.K., Gopalakrishnan, G., Singh, M., Krithika, K.C., Maddineni, S., Cho, D., Zong, B., Zhou, Y., Xiong, C., Savarese, S., Hoi, S., Wang, H.: Merlion: End-to-end machine learning for time series. Journal of Machine Learning Research **24**(226), 1–6 (2023)

Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., Driessche, G.B., Lespiau, J.-B., Damoc, B., Clark, A., *et al.*: Improving language models by retrieving from trillions of tokens. In: Proceedings of the 39th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 162, pp. 2206–2240 (2022)

Behrouz, A., Zhong, P., Mirrokni, V.: Titans: Learning to memorize at test time. arXiv preprint arXiv:2501.00663 (2025) https://doi.org/10.48550/arXiv.2501.00663

Cao, F., Dai, L., Han, J., Xiong, H.: Enhancing multivariate time series forecasting with global temporal retrieval. In: International Conference on Learning Representations (2026)

Cai, R., Huang, H., Jiang, Z., Li, Z., Zhou, C., Liu, Y., Liu, Y., Hao, Z.: Disentangling long-short term state under unknown interventions for online time series forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 39, pp. 15641–15649 (2025)

Chang, C., Hwang, J., Shi, Y., Wang, H., Wang, W., Peng, W.-C., Chen, T.-F.: TimeIMM: A dataset and benchmark for irregular multimodal multivariate time series. In: Advances in Neural Information Processing Systems (Datasets and Benchmarks Track), vol. 38 (2025)

Cohen, B., Khwaja, E., Doubli, Y., Lemaachi, S., Lettieri, C., Masson, C., Miccinilli, H., Ramé, E., Ren, Q., Rostamizadeh, A., Terrail, J., Toon, A.-M., Wang, K., Xie, S., Xu, Z., Zhukova, V., Asker, D., Talwalkar, A.S., Abou-Amal, O.: This time is different: An observability perspective on time series foundation models. In: Advances in Neural Information Processing Systems, vol. 38 (2025)

Che, Z., Purushotham, S., Cho, K., Sontag, D., Liu, Y.: Recurrent neural networks for multivariate time series with missing values. Scientific Reports **8**(1), 6085 (2018) https://doi.org/10.1038/s41598-018-24271-9

Cai, S., Peng, H., Liu, R., Chen, P.: Causal-oriented representation learning for time-series forecasting based on the spatiotemporal information transformation. Communications Physics **8**(1), 242 (2025)

Chang, C., Shi, Y., Cao, D., Yang, W., Hwang, J., Wang, H., Pang, J., Wang, W., Liu, Y., Peng, W.-C., et al.: A survey of reasoning and agentic systems in time series with large language models. Transactions on Machine Learning Research (2025)

Chang, Y.-Y., Sun, F.-Y., Wu, Y.-H., Lin, S.-D.: A memory-network based solution for multivariate time-series forecasting. arXiv preprint arXiv:1809.02105 (2018) https://doi.org/10.48550/arXiv.1809.02105

Cheng, M., Tao, X., Liu, Q., Guo, Z., Chen, E.: Position: Beyond model-centric prediction—agentic time series forecasting. arXiv preprint arXiv:2602.01776 (2026) https://doi.org/10.48550/arXiv.2602.01776

Cho, K., Merriënboer, B., Gulçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724–1734 (2014). https://doi.org/10.3115/v1/D14-1179

Cao, W., Wang, D., Li, J., Zhou, H., Li, L., Li, Y.: BRITS: Bidirectional recurrent imputation for time series. In: Advances in Neural Information Processing Systems, vol. 31, pp. 6776–6786 (2018)

Chen, J., Zhao, Z., Nurbek, G., Feng, A., Maatouk, A., Tassiulas, L., Gao, Y., Ying, R.: TRACE: Grounding time series in context for multimodal embedding and retrieval. In: Advances in Neural Information Processing Systems, vol. 38 (2025)

Cai, X., Zhu, Y., Wang, X., Yao, Y.: MambaTS: Improved selective state space models for long-term time series forecasting. arXiv preprint arXiv:2405.16440 (2024) https://doi.org/10.48550/arXiv.2405.16440

Chen, W., Zhu, Z., Zhang, Y., Shen, L., Yang, L., Wen, Q., Sun, L.: Learning to extrapolate and adjust: Two-stage meta-learning for concept drift in online time series forecasting. In: Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI), pp. 4869–4877 (2025)

Dau, H.A., Bagnall, A., Kamgar, K., Yeh, C.-C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Keogh, E.: The UCR time series archive. IEEE/CAA Journal of Automatica Sinica **6**(6), 1293–1305 (2019) https://doi.org/10.1109/JAS.2019.1911747

Das, S.S.S., Goyal, P., Parmar, M., Peng, N., Tirumalashetty, V., Li, C.-L., Zhang, R., Yoon, J., Pfister, T.: Nexus: An agentic framework for time series forecasting. arXiv preprint arXiv:2605.14389 (2026) https://doi.org/10.48550/arXiv.2605.14389

Das, A., Kong, W., Sen, R., Zhou, Y.: A decoder-only foundation model for time-series forecasting. In: Proceedings of the 41st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 235, pp. 10148–10167 (2024)

Do, V.D., Nguyen, H.H., Nguyen, M.H., Le, H.: SteerCast: Retrieval-based latent steering for decoder-only time series forecasting. In: Advances in Neural Information Processing Systems, vol. 39 (2026)

Dai, Z., Yang, Z., Yang, Y., Carbonell, J.G., Le, Q., Salakhutdinov, R.: Transformer-XL: Attentive language models beyond a fixed-length context. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 2978–2988 (2019). https://doi.org/10.18653/v1/P19-1285

Elman, J.L.: Finding structure in time. Cognitive Science **14**(2), 179–211 (1990) https://doi.org/10.1207/s15516709cog1402_1

Godahewa, R., Bergmeir, C., Webb, G.I., Hyndman, R.J., Montero-Manso, P.: Monash time series forecasting archive. In: Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (2021)

Gwiazda, M., Cai, Y., Goswami, M., Choudhry, A., Dubrawski, A.: TimeSeriesExamAgent: Creating time series reasoning benchmarks at scale. In: International Conference on Learning Representations (2026). https://iclr.cc/virtual/2026/poster/10010760

Garza, A., Challu, C., Mergenthaler-Canseco, M.: TimeGPT-1. arXiv preprint arXiv:2310.03589 (2023) https://doi.org/10.48550/arXiv.2310.03589

Gu, A., Dao, T.: Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752 (2023) https://doi.org/10.48550/arXiv.2312.00752

Gu, A., Goel, K., Ré, C.: Efficiently modeling long sequences with structured state spaces. In: International Conference on Learning Representations (2022). https://openreview.net/forum?id=uYLFoz1vlAC

Gong, D., Liu, L., Le, V., Saha, B., Mansour, M.R., Venkatesh, S., Hengel, A.: Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 1705–1714 (2019). https://doi.org/10.1109/ICCV.2019.00179

Guu, K., Lee, K., Tung, Z., Pasupat, P., Chang, M.: Retrieval augmented language model pre-training. In: Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 3929–3938 (2020)

Garza, A., Rosillo Garcia, R.: TimeCopilot. In: NeurIPS 2025 Workshop on Recent Advances in Time Series Foundation Models: Have We Reached the 'BERT Moment'? (2025). https://openreview.net/forum?id=A11xyQvSmT

Goswami, M., Szafer, K., Choudhry, A., Cai, Y., Li, S., Dubrawski, A.: MOMENT: A family of open time-series foundation models. In: Proceedings of the 41st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 235, pp. 16115–16152 (2024)

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

Gu, Y., Xiong, Y., Mace, J., Jiang, Y., Hu, Y., Kasikci, B., Cheng, P.: Argos: Agentic time-series anomaly detection with autonomous rule generation via large language models. arXiv preprint arXiv:2501.14170 (2025) https://doi.org/10.48550/arXiv.2501.14170

Gong, C., Zhang, C., Yao, D., Bi, J., Li, W., Xu, Y.: Causal discovery from temporal data: An overview and new perspectives. ACM Computing Surveys **57**(4), 1–38 (2024) https://doi.org/10.1145/3705297

Hewamalage, H., Bergmeir, C., Bandara, K.: Recurrent neural networks for time series forecasting: Current status and future directions. International Journal of Forecasting **37**(1), 388–427 (2021) https://doi.org/10.1016/j.ijforecast.2020.06.008

Hu, J., Hu, Y., Chen, W., Jin, M., Pan, S., Wen, Q., Liang, Y.: Attractor memory for long-term time series forecasting: A chaos perspective. In: Advances in Neural Information Processing Systems, vol. 37 (2024)

Han, S., Lee, S., Cha, M., Arik, S.O., Yoon, J.: Retrieval augmented time series forecasting. In: Proceedings of the 42nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 267, pp. 21774–21797 (2025). https://proceedings.mlr.press/v267/han25d.html

Hu, Y., Liu, S., Yue, Y., Zhang, G., *et al.*: Memory in the age of AI agents: A survey. arXiv preprint arXiv:2512.13564 (2025) https://doi.org/10.48550/arXiv.2512.13564

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

Huang, Q., Zhou, Z., Li, Y., Yang, K., Wang, B., Wang, Y.: Many minds, one goal: Time series forecasting via sub-task specialization and inter-agent cooperation. In: Advances in Neural Information Processing Systems, vol. 38 (2025)

Jung, Y.-M., Kim, L.: MambaSL: Exploring single-layer Mamba for time series classification. arXiv preprint arXiv:2604.15174 (2026) https://doi.org/10.48550/arXiv.2604.15174

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

Jiang, H., Pan, S., Dong, Y., Wang, J.: Probabilistic electricity price forecasting based on penalized temporal fusion transformer. Journal of Forecasting **43**(5), 1465–1491 (2024)

Jin, M., Wang, S., Ma, L., Chu, Z., Zhang, J.Y., Shi, X., Chen, P.-Y., Liang, Y., Li, Y.-F., Pan, S., Wen, Q.: Time-LLM: Time series forecasting by reprogramming large language models. In: International Conference on Learning Representations (2024). https://openreview.net/forum?id=Unb5CVPtae

Jiang, R., Wang, Z., Yong, J., Jeph, P., Chen, Q., Kobayashi, Y., Song, X., Fukushima, S., Suzumura, T.: Spatio-temporal meta-graph learning for traffic forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, pp. 8078–8086 (2023). https://doi.org/10.1609/aaai.v37i7.25976

Jing, B., Zhang, S., Zhu, Y., Peng, B., Guan, K., Margenot, A., Tong, H.: Retrieval based time series forecasting. In: CIKM 2022 Workshop on Applied Machine Learning Methods for Time Series Forecasting (AMLTS) (2022). https://doi.org/10.48550/arXiv.2209.13525

Kensinger, E.A., Ford, J.H.: Retrieval of emotional events from memory. Annual Review of Psychology **71**(1), 251–272 (2020) https://doi.org/10.1146/annurev-psych-010419-051123

Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L., Lewis, M.: Generalization through memorization: Nearest neighbor language models. In: International Conference on Learning Representations (2020). https://openreview.net/forum?id=HklBjCEKvH

Karim, F., Majumdar, S., Darabi, H., Harford, S.: Multivariate LSTM-FCNs for time series classification. Neural Networks **116**, 237–245 (2019) https://doi.org/10.1016/

j.neunet.2019.04.014

Kang, J., Seo, J., Park, S., Han, S., Bae, S., Choe, H., Lee, S.: Channel-wise retrieval for multivariate time series forecasting. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2026). https://doi.org/10.1109/ICASSP55912.2026.11463178

Karadag, Y.M., Talaz, I., Dino, I.G., Kalkan, S.: ms-Mamba: Multi-scale Mamba for time-series forecasting. Neurocomputing, 133226 (2026) https://doi.org/10.1016/j.neucom.2026.133226

Karlsson, R.K.A., Willbo, M., Hussain, Z.M., Krishnan, R.G., Sontag, D., Johansson, F.: Using time-series privileged information for provably efficient learning of prediction models. In: Proceedings of the 25th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 151, pp. 5459–5484 (2022)

Kong, Y., Wang, Z., Nie, Y., Zhou, T., Zohren, S., Liang, Y., Sun, P., Wen, Q.: Unlocking the power of LSTM for long term time series forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 39, pp. 11968–11976 (2025)

Kong, Y., Yang, Y., Hwang, Y., Du, W., Zohren, S., Wang, Z., Jin, M., Wen, Q.: Time-MQA: Time series multi-task question answering with context enhancement. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 29736–29753 (2025). https://doi.org/10.18653/v1/2025.acl-long.1437

Le, H., Abbas, S., Nguyen, M.H., Do, V.D., Nguyen, H.H., Nguyen, D.: Accelerating long-term molecular dynamics with physics-informed time-series forecasting. In: Proceedings of the IEEE International Conference on Data Mining (ICDM), pp. 1340–1349 (2025). https://doi.org/10.1109/ICDM65498.2025.00143

Lee, J., Byun, Y., Koo, B., Baek, J.-G.: Memory augmented coherent probabilistic forecasts for hierarchically related time series. Neurocomputing **653**, 131075 (2025) https://doi.org/10.1016/j.neucom.2025.131075

LaBar, K.S., Cabeza, R.: Cognitive neuroscience of emotional memory. Nature Reviews Neuroscience **7**(1), 54–64 (2006) https://doi.org/10.1038/nrn1825

Li, X., Cai, Y., Goswami, M., Wiliński, M., Welter, G., Dubrawski, A.: TimeSeriesGym: A scalable benchmark for (time series) machine learning engineering agents. In: Women in Machine Learning Workshop at NeurIPS 2025 (2025). https://openreview.net/forum?id=8M3qAX6e5M

Li, G., Choi, B., Xu, J., Bhowmick, S.S., Chun, K.-P., Wong, G.L.-H.: ShapeNet: A shapelet-neural network approach for multivariate time series classification. In:

Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 8375–8383 (2021). https://doi.org/10.1609/aaai.v35i9.17018

Lai, G., Chang, W.-C., Yang, Y., Liu, H.: Modeling long- and short-term temporal patterns with deep neural networks. In: Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, pp. 95–104 (2018). https://doi.org/10.1145/3209978.3210006

Lin, T., Guo, T., Aberer, K.: Hybrid neural networks for learning the trend in time series. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI), pp. 2273–2279 (2017). https://doi.org/10.24963/ijcai.2017/316

Li, G., Ge, M., Wan, J., Han, D., Li, M., Zhou, M.: MemMambaAD: Memory-augmented state space model for multivariate time series anomaly detection. Engineering Applications of Artificial Intelligence **158**, 111308 (2025) https://doi.org/10.1016/j.engappai.2025.111308

Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., Long, M.: iTransformer: Inverted transformers are effective for time series forecasting. In: International Conference on Learning Representations (2024). https://openreview.net/forum?id=JePfAI8fah

Li, Y.: PeMS (2025). https://doi.org/10.21227/yj7k-b369

Lee, H., Jin, S., Chu, H., Lim, H., Ko, S.: Learning to remember patterns: Pattern matching memory networks for traffic forecasting. In: International Conference on Learning Representations (2022). https://openreview.net/forum?id=wwDg3bbYBIq

Liang, A., Jiang, X., Sun, Y., Shi, X., Li, K.: Bi-Mamba+: Bidirectional Mamba for time series forecasting. arXiv preprint arXiv:2404.15772 (2024) https://doi.org/10.48550/arXiv.2404.15772

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., Yan, X.: Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In: Advances in Neural Information Processing Systems, vol. 32 (2019)

Liu, X., Lyu, X., Zhang, X., Gao, J., Chen, J.: Memory augmented graph learning networks for multivariate time series forecasting. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM), pp. 4254–4258 (2022)

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: Proceedings of the IEEE 41st International Conference on Data Engineering (ICDE), pp. 3165–3178 (2025)

Liu, Q., Paparrizos, J.: The elephant in the room: Towards a reliable time-series

anomaly detection benchmark. In: Advances in Neural Information Processing Systems (Datasets and Benchmarks Track), vol. 37 (2024). https://doi.org/10.52202/079017-3437

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)

Le, H., Tran, T., Nguyen, T., Venkatesh, S.: Variational memory encoder-decoder. In: Advances in Neural Information Processing Systems, vol. 31 (2018)

Liang, Y., Wen, H., Nie, Y., Jiang, Y., Jin, M., Song, D., Pan, S., Wen, Q.: Foundation models for time series analysis: A tutorial and survey. In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 6555–6565 (2024). https://doi.org/10.1145/3637528.3671451

Liu, X., Xia, Y., Liang, Y., Hu, J., Wang, Y., Bai, L., Huang, C., Liu, Z., Hooi, B., Zimmermann, R.: LargeST: A benchmark dataset for large-scale traffic forecasting. In: Advances in Neural Information Processing Systems (Datasets and Benchmarks Track), vol. 36 (2023)

Liu, H., Xu, S., Zhao, Z., Kong, L., Kamarthi, H., Sasanur, A.B., Sharma, M., Cui, J., Wen, Q., Zhang, C., Prakash, B.A.: Time-MMD: Multi-domain multimodal dataset for time series analysis. In: Advances in Neural Information Processing Systems (Datasets and Benchmarks Track), vol. 37 (2024). https://doi.org/10.52202/079017-2476

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)

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

Lyu, S., Zhong, S., Chen, T., Ruan, W., Liu, Q., Lv, T., Wen, Q., Wong, R.C.-W., Liang, Y.: TS-Memory: Plug-and-play memory for time series foundation models. In: Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 3562–3572 (2026)

Lyu, S., Zhang, Z., Deng, R., Durand, T.: ReDiTT: Retrieval augmented conditional diffusion transformers for asynchronous time series. arXiv preprint arXiv:2607.12391 (2026) https://doi.org/10.48550/arXiv.2607.12391

Liu, Y., Zhang, H., Li, C., Huang, X., Wang, J., Long, M.: Timer: Generative pre-trained transformers are large time series models. In: Proceedings of the 41st International Conference on Machine Learning. Proceedings of Machine Learning

Research, vol. 235, pp. 32369–32399 (2024)

Liu, J., Zhang, C., Qian, J., Ma, M., Qin, S., Bansal, C., Lin, Q., Rajmohan, S., Zhang, D.: Large language models can deliver accurate and interpretable time series anomaly detection. In: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2, pp. 4623–4634 (2025). https://doi.org/10.1145/3711896.3737239

Middlehurst, M., Ismail-Fawaz, A., Guillaume, A., Holder, C., Guijo-Rubio, D., Bulatova, G., Tsaprounis, L., Mentel, L., Walter, M., Schäfer, P., Bagnall, A.: aeon: A Python toolkit for learning from time series. Journal of Machine Learning Research **25**(289), 1–10 (2024)

Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., Shroff, G.: LSTM-based encoder-decoder for multi-sensor anomaly detection. arXiv preprint arXiv:1607.00148 (2016) https://doi.org/10.48550/arXiv.1607.00148

Maru, C., Sato, S.: RATFM: Retrieval-augmented time series foundation model for anomaly detection. arXiv preprint arXiv:2506.02081 (2025) https://doi.org/10.48550/arXiv.2506.02081

Makridakis, S., Spiliotis, E., Assimakopoulos, V.: M5 accuracy competition: Results, findings, and conclusions. International Journal of Forecasting **38**(4), 1346–1364 (2022) https://doi.org/10.1016/j.ijforecast.2021.11.013

Miao, X., Shi, H., Yuan, Y., Pan, D., Wu, Y., Pan, X.: MMNet: Missing-aware and memory-enhanced network for multivariate time series imputation. In: Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI), pp. 3208–3216 (2025). https://doi.org/10.24963/ijcai.2025/357

Malhotra, P., Vig, L., Shroff, G., Agarwal, P.: Long short term memory networks for anomaly detection in time series. In: Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), pp. 89–94 (2015)

Nguyen, M.H., Nguyen, H.H., Nguyen, D., Do, K., Le, H., *et al.*: Reviving error correction in modern deep time-series forecasting. In: Proceedings of the 43rd International Conference on Machine Learning. Proceedings of Machine Learning Research (2026)

Nguyen, H.H., Nguyen, M.H., Nguyen, D., Le, H.: Spectral retrieval-augmented time-series forecasting. arXiv preprint arXiv:2606.19412 (2026) https://doi.org/10.48550/arXiv.2606.19412

Nguyen, H.H., Nguyen, M.H., Nguyen, D., Le, H.: Spectral text fusion: A frequency-aware approach to multimodal time-series forecasting. In: Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS).

Nie, Y., Nguyen, N.H., Sinthong, P., Kalagnanam, J.: A time series is worth 64 words: Long-term forecasting with transformers. In: International Conference on Learning Representations (2023). https://openreview.net/forum?id=Jbdc0vTOcol

Ning, K., Pan, Z., Liu, Y., Jiang, Y., Zhang, J., 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. In: Advances in Neural Information Processing Systems, vol. 38 (2025). https://doi.org/10.52202/085713-5448

Pan, B., Cheng, M., Liu, Z., Yu, S., Tao, X., Wu, Y., Liu, Q., Lian, D., Chen, E.: CastFlow: Learning role-specialized agentic workflows for time series forecasting. arXiv preprint arXiv:2604.27840 (2026) https://doi.org/10.48550/arXiv.2604.27840

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

Park, H., Noh, J., Ham, B.: Learning memory-guided normality for anomaly detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 14372–14381 (2020). https://doi.org/10.1109/CVPR42600.2020.01438

Qiu, X., Hu, J., Zhou, L., Wu, X., Du, J., Zhang, B., Guo, C., Zhou, A., Jensen, C.S., Sheng, Z., Yang, B.: TFB: Towards comprehensive and fair benchmarking of time series forecasting methods. Proceedings of the VLDB Endowment **17**(9), 2363–2377 (2024) https://doi.org/10.14778/3665844.3665863

Qin, Y., Song, D., Cheng, H., Cheng, W., Jiang, G., Cottrell, G.W.: A dual-stage attention-based recurrent neural network for time series prediction. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI), pp. 2627–2633 (2017). https://doi.org/10.24963/ijcai.2017/366

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., Wayne, G.: Experience replay for continual learning. In: Advances in Neural Information Processing Systems, vol. 32 (2019)

Ruan, W., Dang, X., Zhou, Z., Lyu, S., Liang, Y.: A retrieval augmented spatio-temporal framework for traffic prediction. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 40, pp. 39163–39172 (2026). https://doi.org/10.1609/aaai.v40i46.41264

Rae, J.W., Potapenko, A., Jayakumar, S.M., Lillicrap, T.P.: Compressive transformers for long-range sequence modelling. In: International Conference on Learning Representations (2020). https://openreview.net/forum?id=SylKikSYDH

Ravuru, C., Sakhinana, S.S., Runkana, V.: Agentic retrieval-augmented generation for time series analysis. arXiv preprint arXiv:2408.14484 (2024) https://doi.org/10.48550/arXiv.2408.14484

Sellam, A.Z., Benaissa, I., Taleb-Ahmed, A., Patrono, L., Distante, C.: Mamba adaptive anomaly transformer with association discrepancy for time series. Engineering Applications of Artificial Intelligence **160**, 111685 (2025) https://doi.org/10.1016/j.engappai.2025.111685

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., Yao, S.: Reflexion: Language agents with verbal reinforcement learning. In: Advances in Neural Information Processing Systems, vol. 36 (2023)

Singh, A., Ehtesham, A., Kumar, S., Khoei, T.T.: Agentic retrieval-augmented generation: A survey on agentic RAG. arXiv preprint arXiv:2501.09136 (2025) https://doi.org/10.48550/arXiv.2501.09136

Shi, Y., Fu, Z., Chen, S., Zhao, B., Xu, W., Zhang, C., Li, J.: Kronos: A foundation model for the language of financial markets. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 40, pp. 25366–25373 (2026)

Salinas, D., Flunkert, V., Gasthaus, J., Januschowski, T.: DeepAR: Probabilistic forecasting with autoregressive recurrent networks. International Journal of Forecasting **36**(3), 1181–1191 (2020) https://doi.org/10.1016/j.ijforecast.2019.07.001

Staniek, M., Fracarolli, M., Hagmann, M., Riezler, S.: Early prediction of causes (not effects) in healthcare by long-term clinical time series forecasting. In: Proceedings of the 9th Machine Learning for Healthcare Conference. Proceedings of Machine Learning Research, vol. 252 (2024)

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, C., Lin, Y., Guo, S., Wan, H.: Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 914–921 (2020). https://doi.org/10.1609/aaai.v34i01.5438

Song, H., Rajan, D., Thiagarajan, J.J., Spanias, A.: Attend and diagnose: Clinical time series analysis using attention models. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, pp. 4091–4098 (2018). https://doi.org/10.1609/aaai.v32i1.11635

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 **37**(1), 291–305 (2025) https://doi.org/10.1109/TKDE.2024.3484454

Sumers, T.R., Yao, S., Narasimhan, K., Griffiths, T.L.: Cognitive architectures for language agents. Transactions on Machine Learning Research (2024)

Tao, X., Cheng, M., Guo, Z., Yu, S., Liu, Y., Liu, Q., Wang, S.: MemCast: Memory-driven time series forecasting with experience-conditioned reasoning. arXiv preprint arXiv:2602.03164 (2026) https://doi.org/10.48550/arXiv.2602.03164

Tao, X., Cheng, M., Jiang, C., Gao, T., Zhang, H., Liu, Y., Liu, Q.: Cast-R1: Learning tool-augmented sequential decision policies for time series forecasting. arXiv preprint arXiv:2602.13802 (2026) https://doi.org/10.48550/arXiv.2602.13802

Toye, A.A., Celik, A., Kleinberg, S.: Benchmarking missing data imputation methods for time series using real-world test cases. In: Proceedings of the Sixth Conference on Health, Inference, and Learning. Proceedings of Machine Learning Research, vol. 287, pp. 480–501 (2025). https://proceedings.mlr.press/v287/toye25a.html

Tan, Q., Chen, Y., Li, M., Gu, R., Su, Y., Zhang, X.-P.: SynTSBench: Rethinking temporal pattern learning in deep learning models for time series. In: Advances in Neural Information Processing Systems (Datasets and Benchmarks Track), vol. 38 (2025)

Truong, X.-T., Le, T.-K., Kieu, T., Nguyen, T.-T., Nguyen, N.-H.: ALER-TI: Aligned latent embedding retrieval for time series imputation. arXiv preprint arXiv:2607.07640 (2026) https://doi.org/10.48550/arXiv.2607.07640

Tang, W., Liu, L., Long, G.: Interpretable time-series classification on few-shot samples. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2020)

Tire, K., Taga, E.O., Ildiz, M.E., Oymak, S.: Retrieval augmented time series forecasting. arXiv preprint arXiv:2411.08249 (2024) https://doi.org/10.48550/arXiv.2411.08249

Tao, X., Wu, Y., Cheng, M., Guo, Z., Gao, T.: AnomaMind: Agentic time series anomaly detection with tool-augmented reasoning. arXiv preprint arXiv:2602.13807 (2026) https://doi.org/10.48550/arXiv.2602.13807

Tu, S., Zhang, Y., Zhang, J., Fu, Z., Zhang, Y., Yang, Y.: PowerPM: Foundation model for power systems. In: Advances in Neural Information Processing Systems, vol. 37, pp. 115233–115260 (2024)

Vapnik, V., Izmailov, R.: Learning using privileged information: Similarity control and knowledge transfer. Journal of Machine Learning Research **16**(61), 2023–2049 (2015)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30, pp. 5998–6008 (2017). https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

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 (2025). https://proceedings.mlr.press/v267/williams25a.html

Wang, T., Cui, G.: RATSF: Empowering customer service volume management through retrieval-augmented time-series forecasting. arXiv preprint arXiv:2403.04180 (2024) https://doi.org/10.48550/arXiv.2403.04180

Weston, J., Chopra, S., Bordes, A.: Memory networks. arXiv preprint arXiv:1410.3916 (2014) https://doi.org/10.48550/arXiv.1410.3916

Wan, X., Cen, L., Chen, X., Xie, Y., Gui, W.: Memory shapelet learning for early classification of streaming time series. IEEE Transactions on Cybernetics **54**(5), 2757–2770 (2024) https://doi.org/10.1109/TCYB.2023.3337550

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., Yang, X., Zhao, H., Wang, D., Zhang, Y.: Is Mamba effective for time series forecasting? Neurocomputing **619**, 129178 (2025) https://doi.org/10.1016/j.neucom.2024.129178

Woo, G., Liu, C., Kumar, A., Xiong, C., Savarese, S., Sahoo, D.: Unified training of universal time series forecasting transformers. In: Proceedings of the 41st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 235, pp. 53140–53164 (2024)

Wang, Y., Lei, P., Song, J., Hao, Y., Chen, T., Zhang, Y., Jia, L., Li, Y., Wei, Z.: ITFormer: Bridging time series and natural language for multi-modal QA with

large-scale multitask dataset. In: Proceedings of the 42nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 267, pp. 63324–63344 (2025)

Wang, Z.Z., Mao, J., Fried, D., Neubig, G.: Agent workflow memory. arXiv preprint arXiv:2409.07429 (2024) https://doi.org/10.48550/arXiv.2409.07429

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)

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 (IJCAI), pp. 6778–6786 (2023). https://doi.org/10.24963/ijcai.2023/759

Wang, Z., Zhu, Y., Zhao, H., Zheng, X., Sui, D., Wang, T., Tang, W., Wang, Y., Harrison, E., Pan, C., *et al.*: ColaCare: Enhancing electronic health record modeling through large language model-driven multi-agent collaboration. In: Proceedings of the ACM Web Conference 2025, pp. 2250–2261 (2025)

Xu, X., Chen, C., Liang, Y., Huang, B., Bai, G., Zhao, L., Shu, K.: SST: Multi-scale hybrid Mamba-Transformer experts for long-short range time series forecasting. arXiv preprint arXiv:2404.14757 (2024) https://doi.org/10.48550/arXiv.2404.14757

Xie, Z., Li, Z., He, X., Xu, L., Wen, X., Zhang, T., Chen, J., Shi, R., Pei, D.: ChatTS: Aligning time series with LLMs via synthetic data for enhanced understanding and reasoning. Proceedings of the VLDB Endowment **18**(8), 2385–2398 (2025) https://doi.org/10.14778/3742728.3742735

Xu, J., Wu, H., Wang, J., Long, M.: Anomaly transformer: Time series anomaly detection with association discrepancy. In: International Conference on Learning Representations (2022). https://openreview.net/forum?id=LzQQ89U1qm_

Xu, Y., Xu, S., Ramprassad, M., Tumanov, A., Zhang, C.: TransEHR: Self-supervised transformer for clinical time series data. In: Proceedings of the 3rd Machine Learning for Health Symposium (ML4H). Proceedings of Machine Learning Research, vol. 225, pp. 623–635 (2023)

Yang, S., Eisenach, C., Madeka, D.: MQ-ReTCNN: Multi-horizon time series forecasting with retrieval-augmentation. In: KDD 2022 Workshop on Mining and Learning from Time Series – Deep Forecasting: Models, Interpretability, and Applications (2022)

Yu, X., Fan, L., Qiu, X., Dong, N., Huang, Y., Qi, H., Pu, G., Ouyang, W., Chen, X., Hu, J.: MEMTS: Internalizing domain knowledge via parameterized memory for

retrieval-free domain adaptation of time series foundation models. In: Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (2026). https://doi.org/10.1145/3770855.3818032

Yoon, S., Kim, J., Ha, J., Ko, Y.M.: MoMemTo: Patch-based memory gate model in time series foundation model. arXiv preprint arXiv:2509.18751 (2025) https://doi.org/10.48550/arXiv.2509.18751

Yu, Z.-H., Ma, L.-T., Wang, Y.-S., *et al.*: Imputation with inter-series information from prototypes for healthcare time series. Journal of Computer Science and Technology **40**(6), 1499–1511 (2025) https://doi.org/10.1007/s11390-025-4333-3

Yang, S., Wang, D., Zheng, H., Jin, R.: TimeRAG: Boosting LLM time series forecasting via retrieval-augmented generation. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1–5 (2025). https://doi.org/10.1109/ICASSP49660.2025.10889933

Ye, W., Yang, W., Cao, D., Zhang, Y., Tang, L., Cai, J., Liu, Y.: TS-Reasoner: Domain-oriented time series inference agents for reasoning and automated analysis. Transactions on Machine Learning Research (2026)

Zhang, Y., Bu, K., Zhuang, Z., Xie, T., Yu, Y., Li, D., Guo, Y., Lv, D.: CRAFT: Time series forecasting with cross-future behavior awareness. In: Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI), pp. 7062–7070 (2025). https://doi.org/10.24963/ijcai.2025/785

Zeng, A., Chen, M., Zhang, L., Xu, Q.: Are transformers effective for time series forecasting? In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, pp. 11121–11128 (2023). https://doi.org/10.1609/aaai.v37i9.26317

Zhang, Z., Dai, Q., Bo, X., Ma, C., Li, R., Chen, X., Zhu, J., Dong, Z., Wen, J.-R.: A survey on the memory mechanism of large language model-based agents. ACM Transactions on Information Systems **43**(6), 1–47 (2025)

Zhang, C., Fiore, M., Murray, I., Patras, P.: CloudLSTM: A recurrent neural model for spatiotemporal point-cloud stream forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 10851–10858 (2021)

Zhang, X., Gao, T., Cheng, M., Pan, B., Guo, Z., Liu, Y., Tao, X., Liu, Q.: AlphaCast: A human wisdom–LLM intelligence co-reasoning framework for interactive time series forecasting. arXiv preprint arXiv:2511.08947 (2025) https://doi.org/10.48550/arXiv.2511.08947

Zhong, W., Guo, L., Gao, Q., Ye, H., Wang, Y.: MemoryBank: Enhancing large language models with long-term memory. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, pp. 19724–19731 (2024). https://doi.org/10.1609/aaai.v38i17.29946

Zhang, X., Gao, Y., Lin, J., Lu, C.-T.: TapNet: Multivariate time series classification with attentional prototypical network. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 6845–6852 (2020). https://doi.org/10.1609/aaai.v34i04.6165

Zhao, A., Huang, D., Xu, Q., Lin, M., Liu, Y.-J., Huang, G.: ExpeL: LLM agents are experiential learners. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, pp. 19632–19642 (2024). https://doi.org/10.1609/aaai.v38i17.29936

Zhan, T., Jin, M., He, Y., Liang, Y., Deng, Y., Pan, S.: Continuous evolution pool: Taming recurring concept drift in online time series forecasting. arXiv preprint arXiv:2506.14790 (2025) https://doi.org/10.48550/arXiv.2506.14790

Zumarraga, N., Kaar, T., Wang, N., Xu, M.A., Rosenblattl, M., Kreft, M., O'Sullivan, K., Schmiedmayer, P., Langer, P., Jakob, R.: TS-Haystack: A multi-scale retrieval benchmark for time series language models. In: ICLR 2026 Workshop on Time Series in the Age of Large Models (2026). https://openreview.net/forum?id=vGMayMuH83

Zhou, J., Li, D., Li, L., Chen, Z., Wu, S., Ye, H., Lou, J., Spanos, C.J.: Enhancing LLM reasoning for time series classification by tailored thinking and fused decision. arXiv preprint arXiv:2506.00807 (2025) https://doi.org/10.48550/arXiv.2506.00807

Zou, J., Lou, J., Wang, B., Liu, S.: A novel deep reinforcement learning based automated stock trading system using cascaded LSTM networks. Expert Systems with Applications **242**, 122801 (2024) https://doi.org/10.1016/j.eswa.2023.122801

Zhou, T., Ma, Z., Wang, X., Wen, Q., Sun, L., Yao, T., Yin, W., Jin, R.: FiLM: Frequency improved Legendre memory model for long-term time series forecasting. In: Advances in Neural Information Processing Systems, vol. 35, pp. 12677–12690 (2022)

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., Jin, R.: FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In: Proceedings of the 39th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 162, pp. 27268–27286 (2022)

Zhong, S., Ruan, W., Jin, M., Li, H., Wen, Q., Liang, Y.: Time-VLM: Exploring multimodal vision-language models for augmented time series forecasting. In: Proceedings of the 42nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 267, pp. 78478–78497 (2025)

Zhu, Y., Ren, C., Wang, Z., Zheng, X., Xie, S., Feng, J., Zhu, X., Li, Z., Ma, L., Pan, C.: EMERGE: Enhancing multimodal electronic health records predictive modeling with retrieval-augmented generation. In: Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM), pp. 3549–3559 (2024). https://doi.org/10.1145/3627673.3679582

Zhu, Y., Ren, C., Xie, S., Liu, S., Ji, H., Wang, Z., Sun, T., He, L., Li, Z., Zhu, X., *et al.*: REALM: RAG-driven enhancement of multimodal electronic health records analysis via large language models. arXiv preprint arXiv:2402.07016 (2024) https://doi.org/10.48550/arXiv.2402.07016

Zhou, S., Wu, Z., Schöner, H., Fouché, E., Wilson, I.A.G., Wang, S.: Semantics-enhanced retrieval-augmented time series forecasting. In: ICML 2026 Workshop on Forecasting as a New Frontier of Intelligence (2026). https://doi.org/10.48550/arXiv.2606.14941

Zhang, J., Wen, X., Zhang, Z., Zheng, S., Li, J., Bian, J.: ProbTS: Benchmarking point and distributional forecasting across diverse prediction horizons. In: Advances in Neural Information Processing Systems (Datasets and Benchmarks Track), vol. 37 (2024). https://doi.org/10.52202/079017-1523

Zhang, H., Xu, C., Zhang, Y.-F., Zhang, Z., Wang, L., Bian, J.: TimeRAF: Retrieval-augmented foundation model for zero-shot time series forecasting. IEEE Transactions on Knowledge and Data Engineering **37**(9), 5654–5665 (2025) https://doi.org/10.1109/TKDE.2025.3579137

Zhang, Y., Yan, J.: Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In: International Conference on Learning Representations (2023). https://openreview.net/forum?id=vSVLM2j9eie

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

Zhao, H., Zhang, X., Wei, J., Xu, Y., He, Y., Sun, S., You, C.: TimeSeriesScientist: A general-purpose AI agent for time series analysis. arXiv preprint arXiv:2510.01538 (2025) https://doi.org/10.48550/arXiv.2510.01538