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.

Small Language Models for Time Series Prediction: Advances, Challenges, and Future Directions

Atefeh MirAhmadi
Shahid Bahonar University of Kerman
Elham Salari
Shahid Bahonar University of Kerman
Mahdi Eftekhari
m.eftekhari@uk.ac.ir
Shahid Bahonar University of Kerman
Sahar Vahdati
Leibniz University Hannover

Research Article

Keywords: Small Language Models, Time Series Forecasting, Parameter-Efficient Fine-Tuning, Knowledge Distillation, Transformer Architectures, Edge Deployment and Efficiency.

Posted Date: June 25th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9938847/v1

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

Additional Declarations: No competing interests reported.

Small Language Models for Time Series Prediction: Advances, Challenges, and Future Directions

A. MirAhmadi[1†], E. Salari[1†], M. Eftekhari[1,2*], Sahar Vahdati[2,3]

[1*]Department of Computer Engineering, Shahid Bahonar University, Kerman, Iran.
[2]Institute for Applied Computer Science (InfAI), Nature-Inspired Machine Intelligence, Dresden, Germany.
[3]Leibniz University Hannover (LUH), Hanover, Germany.

*Corresponding author(s). E-mail(s): m.eftekhari@uk.ac.ir;
Contributing authors: Sahar.Vahdati@tib.eu;
†These authors contributed equally to this work.

Abstract

Time series forecasting is essential across finance, energy, healthcare, and Internet of Things (IoT) domains, yet deploying Large Language Models (LLMs) for this task faces substantial computational and privacy constraints. Small Language Models (SLMs), with 100 million to 10 billion parameters, offer efficient alternatives enabling edge deployment while maintaining strong predictive performance. This survey systematically reviews 168 publications (2013–2026), examining SLM architectures, training paradigms, and applications in temporal forecasting. The analysis covers construction methodologies including knowledge distillation, pruning, quantization, and Parameter-Efficient Fine-Tuning (PEFT), presenting a comprehensive taxonomy of architectures from distilled Transformers to retrieval-augmented and neuro-symbolic systems. Four principal paradigms emerge: Prompt Engineering for direct prediction, domain-specific Fine-Tuning, Retrieval-Augmented approaches, and Hybrid Pipelines. Through multi-domain evaluation, the survey demonstrates compelling trade-offs of SLMs across accuracy, efficiency, privacy, and cost. Critical assessment addresses challenges including the continuous-discrete, context limitations, and distribution shifts, while identifying promising research directions in efficient attention mechanisms, semantic tokenization, dynamic knowledge integration, and standardized benchmarking. This survey guides practitioners in selecting appropriate SLM strategies and highlights opportunities for advancing this rapidly evolving field.

Keywords: Small Language Models, Time Series Forecasting, Parameter-Efficient Fine-Tuning, Knowledge Distillation, Transformer Architectures, Edge Deployment and Efficiency.

1 Introduction

Time series forecasting, the task of predicting future values from historical sequential data, is a critical capability across numerous domains. In **finance**, SLMs facilitate accurate projections of asset prices, market trends, and economic indicators through efficient fine-tuning on financial corpora [1–3]. Within the **energy sector**, forecasting is integral for load prediction, electricity price estimation, and balancing supply with demand—particularly crucial for integrating intermittent renewable energy sources [3–5]. In **healthcare**, predictive modeling assists in anticipating patient admissions, detecting early disease outbreaks, and tailoring personalized treatment strategies, thereby improving operational efficiency and patient outcomes [6–8]. Within the **IoT**, time series analysis underpins predictive maintenance, anomaly detection, and adaptive control in connected devices, allowing proactive and cost-effective interventions [9, 10]. The deployment of SLMs in these sectors demonstrates that improvements in forecast accuracy directly translate to better decision-making, reduced costs, and enhanced resilience, while addressing the inference latency and computational efficiency constraints that often limit the practical application of larger models in real-world scenarios.

Recent years have seen remarkable advances in **LLMs**, with billions of parameters enabling state-of-the-art performance on diverse tasks [11–13]. Comprehensive surveys review the application of these foundation models to time series analysis, covering architectures, pre-training techniques, and adaptation methods [14, 15]. Despite the fact that increasing the size of models can significantly increase performance on various tasks, training such large models requires expensive computational resources and their large sizes may result in unaffordable inference training budgets, which limits research access. SLMs are designed as lightweight versions of traditional language models to perform efficiently in resource-constrained environments such as smartphones, embedded systems, or low-power computers. While LLMs have hundreds of billions or even trillions of parameters, SLMs are significantly smaller, with dimensions ranging from 100 million to 10 billion parameters, but still retain the core capabilities [16–18]. This compact architecture offers several advantages: their low computational requirements allow them to run on edge devices and consumer laptops, and they are an environmentally friendly option with reduced power consumption. Additionally, the smaller size leads to faster inference, which is critical for real-time applications, and enables the implementation of on-device AI, which increases privacy and security by eliminating the need for cloud connectivity. Finally, lower deployment costs and easy customization for specific tasks (such as legal document analysis) make SLMs an accessible and powerful tool for startups and developers [19, 20]. The motivation for choosing between SLMs and LLMs involves a critical trade-off across multiple dimensions beyond raw accuracy. While LLMs achieve superior performance on complex tasks,

SLMs offer compelling advantages in computational efficiency, deployment flexibility, privacy preservation, and cost-effectiveness for edge and real-time applications.

The remainder of this survey is organized as follows: Section 2 presents the Research Methodology, detailing the systematic review process following PRISMA 2020 guidelines. This section describes the comprehensive search strategy across major databases (IEEE Xplore, arXiv, Elsevier, Springer, ACM, and leading AI conferences), and specifies the inclusion and exclusion criteria. The three-stage study selection process—Identification, Screening, and Eligibility Assessment—is illustrated through a PRISMA flow diagram, culminating in a final corpus of 168 studies spanning 2013–2026. Section 3 establishes the Background on SLMs, providing the theoretical and technical foundations for understanding SLMs. We begin by characterizing SLMs through an operational definition (100 million to 10 billion parameters) and examine their expanding application landscape in time series forecasting. The section then comprehensively reviews the methodologies for building and optimizing SLMs, including knowledge distillation, pruning, quantization, and PEFT. A comparative analysis of these techniques and an architectural outlook—including emerging alternatives like Structured State Space Models (**Mamba**)—conclude this foundational section.

Section 4 presents a detailed Taxonomy of SLM Architectures and Training Paradigms. This section systematically categorizes the four principal paradigms: (1) Distilled Transformer Variants, examining models like **DistilBERT** and **TinyBERT** that compress knowledge from larger teachers; (2) PEFT Paradigms, covering adapters, prefix-tuning, and Low-Rank Adaptation (LoRA) for efficient task adaptation; (3) Efficient Attention Architectures, including sparse and linear-attention models such as Reformer and Long-Short Transformer that address quadratic complexity; and (4) Hybrid Frameworks, encompassing Retrieval-Augmented SLMs (RAG) with specialized encoders like **Patch Time Series Transformer (PatchTST)**, as well as neuro-symbolic systems that combine neural flexibility with logical precision.

Section 5 explores Applications to Time Series Prediction, demonstrating how SLMs are deployed across diverse forecasting scenarios. This section presents a four-fold classification of methodological approaches: (1) Prompt engineering for direct prediction, which transforms time series data into textual representations for zero-shot or few-shot inference; (2) Fine-tuning strategies, adapting SLMs to temporal data through efficient windowing and domain-specific optimization; (3) Retrieval-augmented models, overcoming context limitations by connecting models to dynamic knowledge bases of historical patterns; and (4) Hybrid pipelines, combining the reasoning capabilities of language models with the numerical precision of classical statistical methods. Domain-specific applications in finance, energy, healthcare, and IoT are examined with representative case studies.

Section 6 addresses Evaluation Metrics and Benchmarking, providing a comprehensive overview of quantitative measures for assessing SLM forecasting performance. Standard metrics including MAE, RMSE, MAPE, and domain-specific indicators are discussed, alongside emerging benchmarks for comparing SLMs against both traditional methods and larger language models.

Datasets and Benchmarks are described in Section 7. The challenges and limitations facing SLM deployment in time series forecasting are identified in Section 8. The

key ethical, privacy, and reliability challenges are highlighted in Section 9 that arise when applying small language models to time series forecasting, especially in sensitive domains such as finance and healthcare. It emphasizes risks stemming from model opacity ("black-box" behavior), potential leakage of private data during training, vulnerability to adversarial manipulation, and the transfer of hidden statistical or cultural biases into downstream applications. The section also stresses the need for robust fairness evaluation, privacy-preserving deployment strategies for edge/IoT environments, and mechanisms to prevent unreliable or hallucinated forecasts. Key issues examined include the continuous-discrete modality gap between numerical data and language tokens, context window limitations for long-horizon dependencies, distribution shifts and concept drift in non-stationary environments, and interpretability concerns in safety-critical applications. To bridge the gap between theoretical advancements and practical implementation, real-world deployments of compact and language-inspired time series models across diverse domains are described in Section 10. These case studies highlight how efficiency, adaptability, and system-level considerations shape the successful adoption of such models in operational environments.

Section 11 outlines Future Research Directions, highlighting promising avenues for advancing the field. These include efficient attention mechanisms for extended temporal contexts, semantic tokenization strategies that preserve numerical precision, dynamic knowledge Integration through adaptive retrieval systems, federated learning approaches for privacy-preserving forecasting, and the development of standardized benchmarks and evaluation protocols. Eventually, Section 12 concludes this research.

2 Research Methodology

The study selection process follows a structured screening protocol and is presented using a PRISMA-style flow diagram, adapted to the interdisciplinary scope of this review, which spans computer science, machine learning, and time series forecasting applications. Fig. 1 summarizes the workflow from data collection through quality assessment and in-depth review to the final corpus.

This systematic review follows the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA 2020) guidelines to ensure transparency and reproducibility in the identification, screening, and synthesis of relevant literature on SLMs for time series forecasting.

2.1 Data Collection

We queried five complementary sources:

- **IEEE Xplore Digital Library**, for engineering and computer-science venues;
- **ACM Digital Library**, for computing and AI research publications;
- **arXiv and Preprint Servers**, capturing recent advances not yet peer-reviewed;
- **Elsevier (ScienceDirect) and Springer**, for established journal publications;
- **Major AI Conference Proceedings** (NeurIPS, ICML, AAAI, ICLR), for cutting-edge research in machine learning and artificial intelligence.

Table 1 Sources, representative query logic, and retrieved records used in the study-identification stage.

Source	Search dates	Representative query logic	Records
IEEE Xplore Digital Library	Jan. 2025–May. 2026	Boolean searches combining SLM terms (`"small language model"`, `SLM`, `"lightweight language model"`, `"efficient language model"`, `"compact Transformer"`) with forecasting terms (`"time series forecasting"`, `"temporal prediction"`, `"sequence forecasting"`) and methodology terms (`"knowledge distillation"`, `pruning`, `quantization`, `LoRA`, `PEFT`).	105
ACM Digital Library	Jan. 2025–May. 2026	Representative queries included `"small language model" AND "time series"`, `"efficient Transformer" AND forecasting`, `"lightweight LLM" AND prediction`, and related variants combining SLM terminology with temporal modeling and edge deployment concepts.	88
arXiv and Preprint Servers	Jan. 2025–May. 2026	Queries targeting recent preprints using combinations such as `"small language model"`, `"efficient LLM"`, `"time series forecasting"`, `"parameter-efficient fine-tuning"`, `"edge deployment"`, and `"real-time prediction"`.	132
Elsevier (ScienceDirect) and Springer	Jan. 2025–May. 2026	Boolean searches combining `"language model" AND "time series"`, `"Transformer" AND "forecasting"`, `"deep learning" AND "temporal prediction"`, with domain filters for finance, energy, healthcare, and IoT applications.	75
Major AI Conference Proceedings (NeurIPS, ICML, AAAI, ICLR)	Jan. 2025–May. 2026	Manual searches in conference proceedings (covering 2020–2025) using combinations such as `"efficient language model"`, `"small Transformer"`, `"time series"`, `"forecasting"`, `"knowledge distillation"`, and `"quantization"`.	50
Total identified records (Jan. 2025–May. 2026)			**450**

Queries combined SLM-related terms with time series forecasting terminology. The search terms were refined iteratively during the search process: when preliminary screening revealed recurrent concepts not yet covered by the initial queries, additional terms were introduced. The inclusion criteria required publications to be from January 2013 to May 2026. Table 1 details the search strings used for each source and the number of records retrieved. The initial pool included both primary research studies and review articles. Review papers that met the search criteria are included in the final set of 168 studies; they primarily provide context in the Introduction, Background, and foundational discussions, while the majority of the studies analyzed in detail in the body sections are original research.

The systematic database search was originally completed in 2025. The search dates in Table 1 are listed as extending to May 2026 to account for three studies that were initially identified as 2025 preprints and subsequently received a 2026 publication date upon final publication. A confirmatory check in May 2026 verified that no additional eligible records had appeared since the original search. The total count of identified records therefore reflects the outcome of the initial search, as no new studies were discovered in the confirmatory check.

2.2 Quality Assessment

The complete eligibility criteria used in the review are summarized in Table 2. Across all stages, studies had to fall within the 2013–2026 publication window and have an accessible full text in English. The initial identification pool included both primary studies and review articles. As shown in Fig. 1, review papers that met the search criteria were retained and, where relevant, passed through the same screening stages; they are included among the final 168 studies and serve primarily as contextual references in the Introduction, Background, and foundational discussions. After duplicate or overlapping records were removed, approximately 378 candidate records remained for formal screening (including review papers that met the initial inclusion criteria).

In **Step 1 of Fig. 1** (technical screening), the candidate records were assessed by reviewers with expertise in machine learning and Natural Language Processing (NLP). At this stage, a study had to satisfy the technical eligibility criteria listed in Table 2: it had to apply at least one SLM, Transformer-based, or parameter-efficient method; include experimental or dataset-based evaluation rather than only a conceptual proposal; and report quantitative evaluation results. Records that failed these technical or empirical requirements, or were otherwise outside the technical scope of the review, were excluded. This step resulted in the exclusion of 127 records, leaving 251 papers.

In **Step 2 of Fig. 1** (domain screening), the remaining papers were assessed by reviewers with domain expertise in time series forecasting and edge deployment. At this stage, a study had to satisfy the domain eligibility criteria listed in Table 2: it had to focus on time series forecasting or temporal prediction tasks and address a practically meaningful application scenario, such as finance, energy, healthcare, IoT, or real-time systems. Papers centered on non-forecasting tasks or lacking clear relevance to time series applications were excluded. This step led to the exclusion of 48 additional records, leaving 203 papers for in-depth scientific review.

2.3 In-Depth Scientific Review

The 203 papers that passed both screening stages advanced to full-text review (**Steps 3 to 5 in Fig. 1**). Each paper was read in full by at least two of the reviewing authors to verify that the study satisfied all inclusion criteria in Table 2 when examined beyond the title and abstract. At this stage, papers were excluded if full-text examination revealed insufficient methodological detail (for example, no description of the training procedure, dataset, or evaluation protocol), if the study focused exclusively on LLMs without SLM comparison or adaptation, or if the scope of the study

fell outside the review upon closer reading. For borderline cases, inclusion or exclusion was decided through discussion among the reviewing authors until agreement was reached. Thirty-five papers were excluded at the in-depth scientific review stage due to: absence of empirical evaluation (n = 14), exclusive LLM focus without SLM consideration (n = 11), no forecasting task (n = 6), or insufficient methodological detail (n = 4).

The pipeline converged on a final corpus of **168 studies** published between 2013 and 2026.

2.4 Data Extraction and Categorization

For every included paper, we extracted: (1) publication year; (2) title and authors; (3) model architecture and parameter count; (4) training paradigm (e.g., knowledge distillation, pruning, PEFT); (5) application domain; (6) forecasting task and evaluation metrics; and (7) two classification labels described below.

Application domain.

Each paper was assigned to one of four primary application domains based on the forecasting scenario it addresses: **Finance and Economics**, **Energy and Utilities**, **Healthcare and Biomedical**, and **IoT and Edge Computing**. Cross-domain studies were assigned to the domain most emphasized in their experimental evaluation.

Thematic category.

Within each domain, every paper was further assigned to one of ten thematic categories based on its primary contribution or methodological focus. The categories are:

1. **Foundational & Background.** Studies providing theoretical foundations, surveys, or background on language models and time series forecasting [14, 21–24].
2. **SLM Architecture & Construction.** Studies proposing novel SLM architectures or construction methodologies specifically designed for efficiency [18, 20, 25, 26].
3. **Efficient Attention Mechanisms.** Studies focusing on sparse attention, linear attention or other computationally efficient attention variants [27–31].
4. **Prompt Engineering & Alignment.** Studies addressing prompt design, instruction tuning, or alignment techniques for time series tasks [32–37].
5. **Fine-Tuning & Adaptation.** Studies on PEFT, domain adaptation, or transfer learning for SLMs [38–42].
6. **Retrieval-Augmented Generation (RAG).** Studies combining retrieval mechanisms with SLMs for enhanced forecasting [43–47].
7. **Hybrid & Neuro-Symbolic Methods.** Studies integrating SLMs with classical statistical models or symbolic reasoning [48–52].
8. **Domain Applications.** Studies demonstrating SLM applications in specific domains: energy, healthcare, finance, demand forecasting and IoT [4–7, 9, 10, 53–58].
9. **Edge Deployment & Efficiency.** Studies addressing deployment on resource-constrained devices, latency optimization, or energy efficiency [59–63].

Table 2 Inclusion and exclusion criteria used during screening.

Criterion	Description
Inclusion criteria (all must be met)	
Model scope	Study focuses on language models with 100M–10B parameters (SLMs).
AI/ML method	Study applies at least one SLM, Transformer-based, or parameter-efficient method, including knowledge distillation, pruning, quantization, LoRA, or other PEFT approaches.
Quantitative evaluation	Reports at least one standard metric (MAE, RMSE, MAPE, accuracy, F1-score, etc.).
Forecasting task	Model output addresses time series forecasting, temporal prediction, or sequence modeling.
Full text	Full text available in English.
Publication period	Published January 2013–May 2026.
Peer review status	Peer-reviewed journal/conference papers or preprints with empirical evaluation.
Exclusion criteria (any one is sufficient)	
No SLM method	Study does not apply any Small Language Model or parameter-efficient approach.
No quantitative results	No performance metrics reported.
Exclusive LLM focus	Study focuses exclusively on LLMs (>10B parameters) without SLM comparison.
No forecasting task	Study does not address time series forecasting or temporal prediction.
Conceptual only	System described without implementation or experimental evaluation.
Insufficient detail	Full-text review shows no description of training procedure, dataset, or evaluation protocol.
No full text	Full text not accessible.
Non-English	Publication not in English.
Duplicate	Duplicate publications or extended abstracts without full methodology.

10. **Evaluation, Robustness & Trustworthiness.** Studies focusing on evaluation methodologies, robustness testing, or trustworthiness of SLM predictions [64–71].

Table 3 presents the distribution of the 168 included studies across these major thematic categories.

Table 4 summarizes the key dimensions of comparison between SLMs and LLMs.

As seen, Table 5 illustrates the temporal distribution of the 168 included studies, demonstrating the rapid growth of research interest in SLMs for time series forecasting, particularly from 2023 onwards.

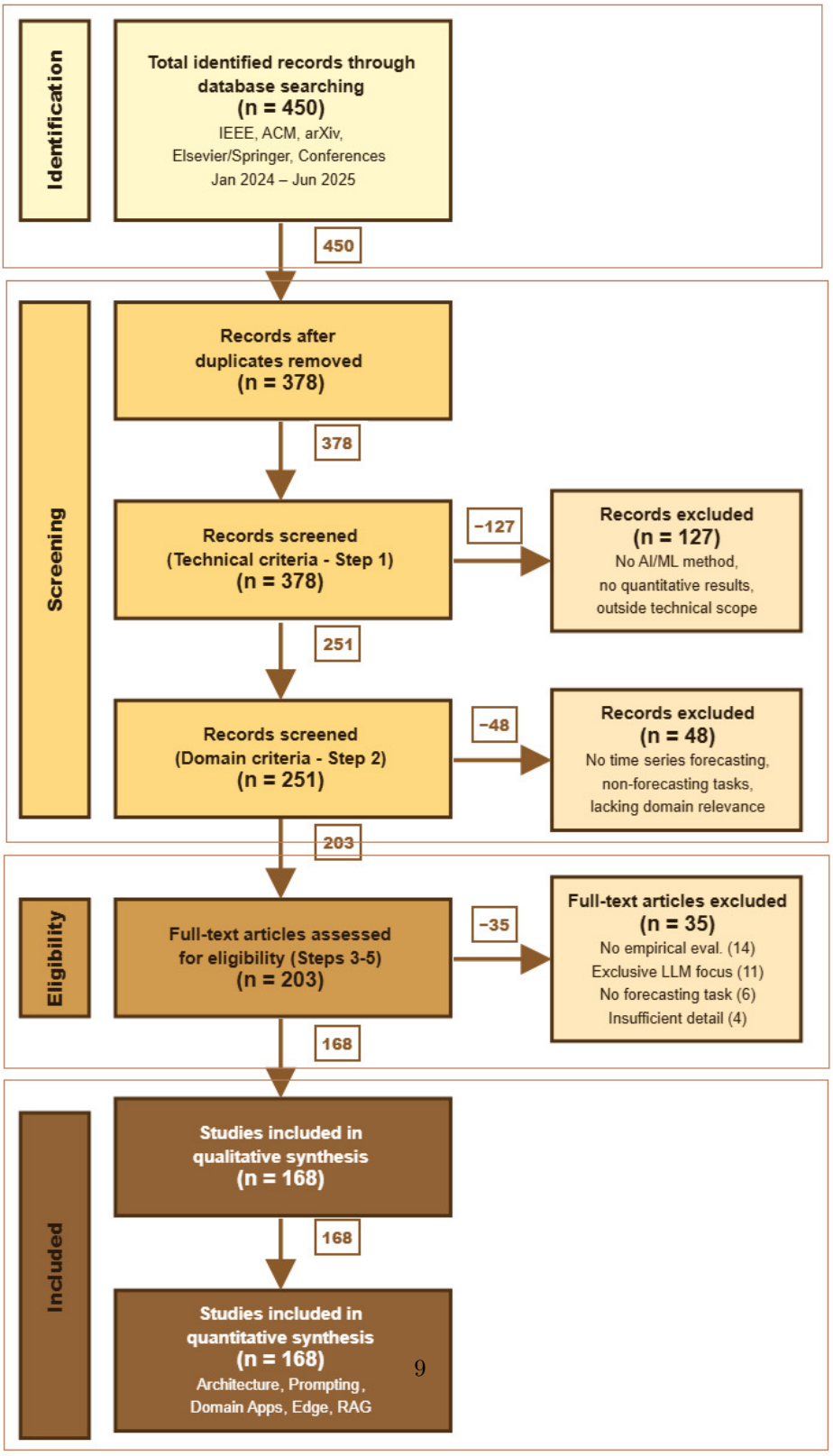


Fig. 1 PRISMA flow diagram illustrating the systematic study selection process for the review of SLMs in time series forecasting. Starting from 450 identified records, the pipeline converges on a final corpus of 168 studies spanning 2013–2026.

Table 3 Distribution of 168 included studies across major thematic categories.

Category	Count	Percentage
Foundational & Background	27	16.07%
SLM Architecture & Construction	13	7.74%
Efficient Attention Mechanisms	13	7.74%
Prompt Engineering & Alignment	9	5.36%
Fine-Tuning & Adaptation	20	11.9%
Retrieval-Augmented Generation (RAG)	6	3.57%
Hybrid & Neuro-Symbolic Methods	21	12.5%
Domain Applications (Finance, Energy, Healthcare, IoT)	18	10.71%
Edge Deployment & Efficiency	18	10.71%
Evaluation, Robustness & Trustworthiness	23	13.69%
Total	**168**	**100%**

Table 4 Comparison of SLMs and LLMs across key deployment dimensions.

Dimension	SLMs	LLMs
Parameter count	100M–10B	10B–1T+
Inference latency	Low (ms)	High (s)
Memory footprint	<16 GB	>80 GB
Edge deployment	✓	×
Privacy (on-device)	✓	×
Training cost	Low–Moderate	Very High
Real-time capability	✓	Limited
Domain customization	Easy (PEFT)	Expensive

3 Background on SLMs

The introduction of the Transformer architecture by Vaswani et al. [21], with its seminal attention mechanism, fundamentally altered the landscape of NLP and established the foundation for modern language models [21]. This architectural paradigm catalyzed the development of LLMs, demonstrating remarkable performance on a diverse array of tasks [12]. The scaling of these models, however, presents substantial impediments, including immense computational requirements, high inference costs, and significant data privacy concerns, which often necessitate cloud-based deployment [23, 26]. In response to these limitations, a parallel research trajectory has emerged focusing on SLMs a class of models engineered to provide efficient, accessible, and secure solutions for specialized applications and resource-constrained environments [23, 24]. This section establishes an operational definition for SLMs and provides a critical review of the foundational methodologies for their construction and optimization.

Table 5 Temporal distribution of the 168 included studies by publication year.

Year	Count	Cumulative
2013–2019	10	10
2020	7	17
2021	9	26
2022	7	33
2023	21	54
2024	47	101
2025	64	165
2026 (Jan–May)	3	168

3.1 Characterizing SLMs: Operational Definition and Application Landscape for Time Series Forecasting

The delineation between "small" and "large" models is a fluid concept, subject to redefinition with ongoing technological advancements [24]. For the scope of this review, we establish an operational definition that considers models with parameter counts in the range of 100 million to 10 billion as SLMs. This range is chosen to encompass models that are sufficiently large to exhibit meaningful capabilities, yet small enough to be deployed on high-end consumer hardware [24, 49, 72]. This is in stark contrast to LLMs, which can scale to hundreds of billions of parameters and demand extensive computational infrastructure [24]. The principal advantage of SLMs lies in their computational efficiency, which facilitates on-device or edge deployment. Figure 2 visually illustrates this operational definition, depicting the parameter count spectrum and corresponding deployment targets for SLMs. As shown, the 100 million to 10 billion parameter range enables deployment across various platforms, from mobile devices to edge servers, while maintaining computational efficiency. This paradigm inherently reduces latency and preserves data privacy by processing information locally [24, 26]. Notably, the training strategy for many SLMs deviates from the compute-optimal scaling laws proposed by the Chinchilla model [22]; this process, often termed "over-training," involves training on a significantly larger volume of tokens to maximize the model's performance for a given inference budget [72].

As illustrated in Figure 3, the application landscape of SLMs for time series forecasting has expanded significantly between 2019 and 2025. Early studies primarily focused on a limited set of domains and exploratory analyses. For instance, initial works investigated the feasibility of leveraging compact language model architectures for sequential prediction and reasoning over structured temporal data [23, 26]. These early contributions demonstrated that SLMs could capture temporal dependencies and contextual patterns while maintaining computational efficiency.

More recent studies have accelerated this trend and highlighted the growing versatility of SLM-based forecasting frameworks. In particular, works such as [24, 49, 72] emphasize the integration of lightweight language models into domain-specific pipelines, enabling interpretable and adaptable forecasting systems. As shown

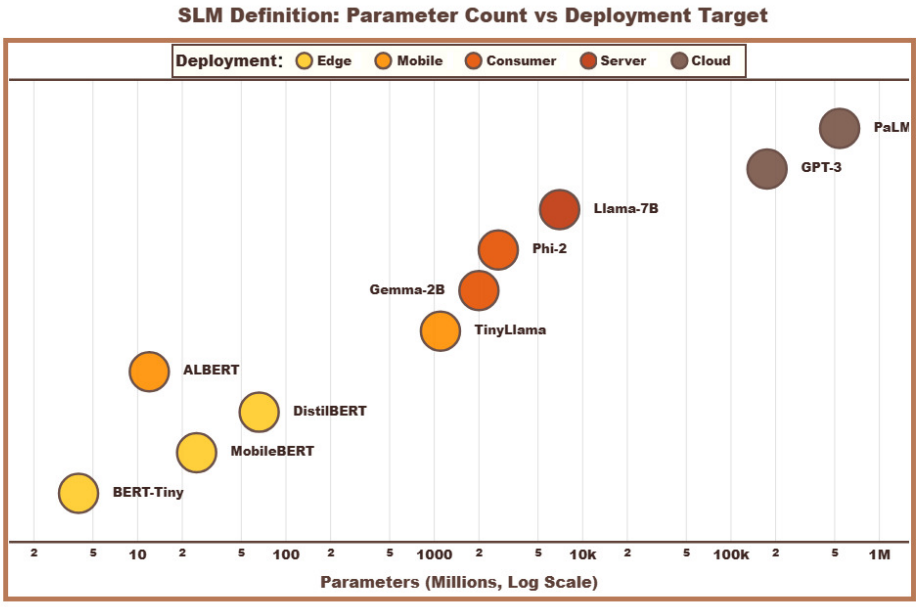


Fig. 2 Operational definition of SLMs showing the parameter count range (100 million to 10 billion) and corresponding deployment targets including mobile devices, edge computing platforms, and consumer-grade hardware. The visualization highlights the trade-off between model capacity and deployment feasibility that characterizes SLMs.

in the figure, the number of applications increases sharply after 2023, reflecting the rising interest in deploying SLMs across multiple real-world sectors.

The emerging applications span a wide range of domains, including healthcare demand prediction, energy consumption forecasting, and financial market analysis [23, 26, 49]. In addition, new research directions have appeared in IoT monitoring, transportation flow prediction, and supply chain optimization. Environmental and societal applications are also gaining attention, such as weather and climate modeling, agricultural yield prediction, and smart city analytics. Furthermore, industrial and commercial sectors—including manufacturing and e-commerce—are beginning to adopt SLM-based forecasting techniques to support decision-making and operational planning. Overall, the trend shown in Figure 3 illustrates a transition from exploratory research to broad cross-domain adoption. The rapid diversification of applications after 2023 indicates that SLMs are becoming a practical and scalable alternative for time series forecasting tasks, particularly in scenarios where computational efficiency, interpretability, and domain adaptability are critical.

3.2 Methodologies for Building and Optimizing SLMs

The creation of SLMs follows two primary pathways: training from scratch or the more prevalent approach of compressing and adapting pre-trained LLMs. The latter

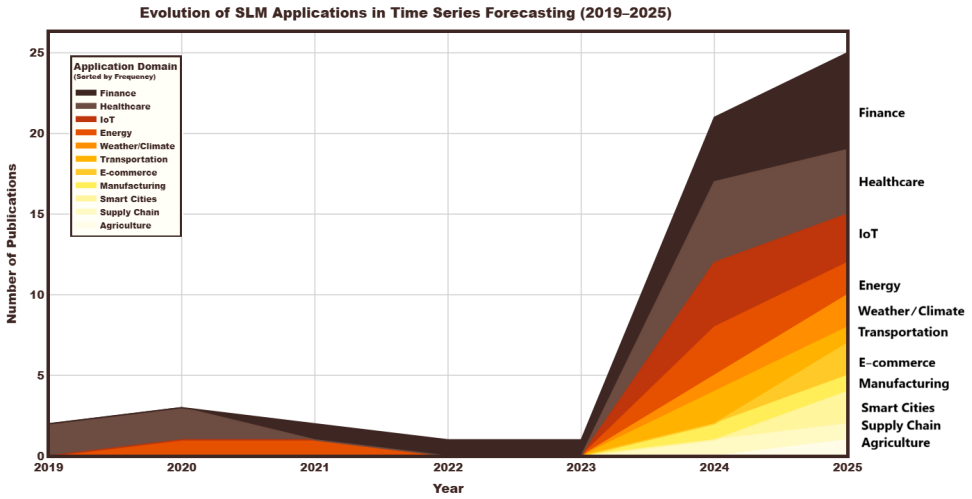


Fig. 3 Evolution of SLM applications in time series forecasting across domains (2019-2025). The figure shows the growing number of publications and increasing diversity of application areas, with a vertical gradient from light (bottom) to dark (top) indicating total publication volume. The sharp post-2023 rise reflects surging research interest in domain-specific SLM deployment for forecasting tasks.

leverages the extensive knowledge encoded within LLMs and transfers it into a more resource-efficient format. The key techniques for this process are reviewed below.

As a form of "behavior transfer," **knowledge distillation** involves training a smaller "student" model to replicate the outputs of a larger "teacher" model. The student learns not only from ground-truth labels (hard targets) but also from the "soft targets": the full output probability distribution of the teacher. These soft targets, produced by applying a "temperature" parameter to the softmax function, convey the teacher's nuanced understanding of inter-class relationships and generalization patterns [73]. DistilBERT serves as a prominent example, retaining 97% of BERT's capabilities while being 40% smaller in parameter count [74].

Model compression via pruning and quantization techniques aim to create a smaller model by structurally modifying a larger one. **Pruning** is a method that enhances computational efficiency by removing redundant or less salient parameters from a pre-trained network. It can be performed in an unstructured manner, where individual weights are removed, or through structured pruning, where entire components like neurons or attention heads are eliminated. The latter is generally more compatible with standard hardware, facilitating tangible inference speedups [24]. Advanced one-shot methods like SparseGPT can induce up to 60% sparsity in massive models without requiring any retraining [75]. **Quantization** reduces the numerical precision of a model's parameters and/or activations (e.g., from 16-bit floating-point to 4-bit integers), thereby decreasing memory footprint and accelerating computation [24, 76]. A primary challenge is the presence of outliers in activation tensors; SmoothQuant mitigates this by mathematically shifting the quantization difficulty

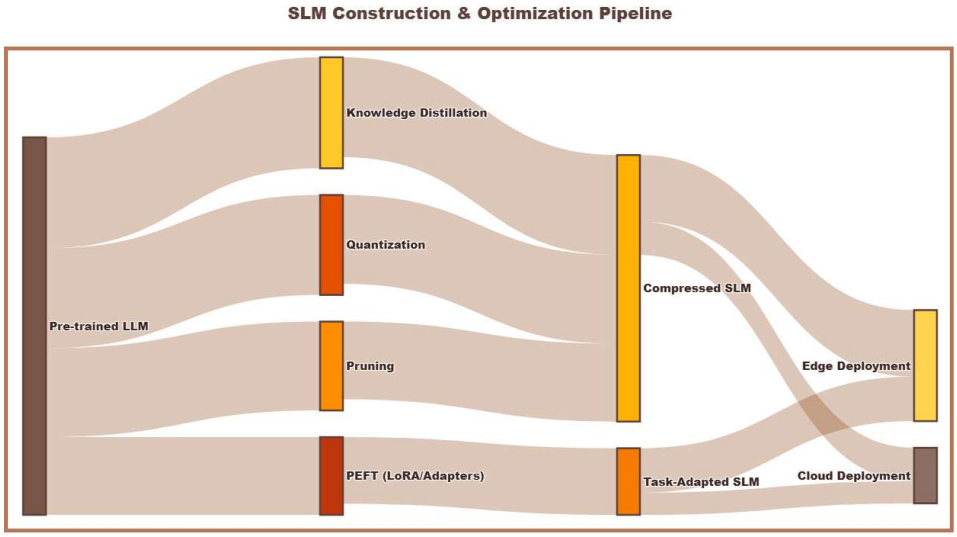


Fig. 4 The SLM construction and optimization pipeline illustrating the flow from pre-trained LLMs to deployment targets through various compression and adaptation techniques.

from the sensitive activations to the more robust weights [77]. Architectures like BitNet have taken this concept to its logical conclusion, employing 1.58-bit weights (ternary values of $\{-1, 0, 1\}$) to replace most matrix multiplications with additions [78].

Distinct from compression, **PEFT** methods focus on efficiently adapting a single pre-trained model for various downstream tasks by updating only a small fraction of its parameters [24, 39]. **Adapters** involve inserting small, trainable modules with a bottleneck architecture between the layers of a frozen pre-trained model. Only the parameters of these adapter modules are tuned for a new task [40]. **Prompt tuning** keeps the entire pre-trained model frozen and instead learns a "soft prompt" (a sequence of trainable continuous vectors prepended to the input) to steer the model's behavior [79]. **LoRA** postulates that weight updates during adaptation have a low "intrinsic rank." It approximates the update matrix ΔW with a low-rank decomposition, $\Delta W = BA$, where A and B are much smaller matrices. Only these smaller matrices are trained. A key benefit is the absence of inference latency, as the trained matrices can be merged with the original weights during deployment [39].

Figure 4 presents a Sankey diagram that visualizes the comprehensive pipeline for constructing and optimizing SLMs. The diagram illustrates the transformation flow from a pre-trained LLM to final deployment targets through multiple optimization pathways.

The pipeline begins with a **pre-trained LLM** as the source, which branches into four primary optimization techniques: **knowledge distillation**, **pruning**, **quantization**, and **PEFT**.

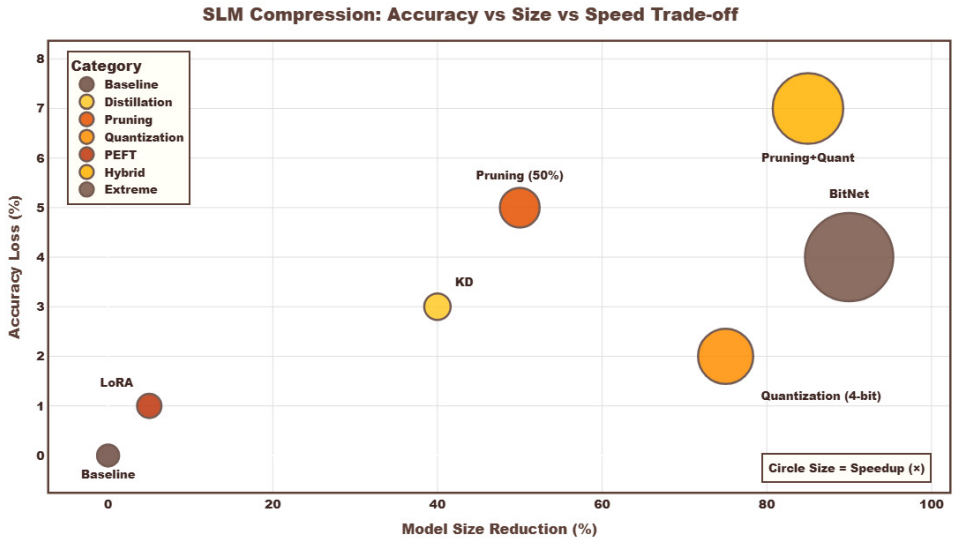


Fig. 5 Trade-off between model size reduction, accuracy loss, and speedup for different SLM compression methods. Larger circles correspond to higher speedups.

The first three techniques (knowledge distillation, pruning, and quantization) converge to produce a **compressed SLM**, which represents a structurally reduced version of the original model. Meanwhile, PEFT methods lead to a **task-adapted SLM**, optimized for specific downstream applications without significant structural modifications.

Both the compressed SLM and task-adapted SLM can be deployed to either **edge deployment** environments (e.g., mobile devices, IoT sensors, embedded systems) or **cloud deployment** infrastructure, depending on the application requirements. The flow weights in the diagram indicate the relative prevalence of each pathway, with knowledge distillation and pruning representing the most commonly adopted compression strategies, while Edge Deployment emerges as the predominant target for optimized SLMs due to the inherent advantages of local processing, reduced latency, and enhanced data privacy.

The following Figure 5 illustrates the trade-off between model size reduction, accuracy loss, and inference speedup across several SLM compression and optimization methods. Larger circles correspond to higher speedups, while colors indicate different technique categories such as knowledge distillation, pruning, quantization, PEFT, and hybrid approaches.

3.3 Comparative Analysis and Architectural Outlook

The methodologies for developing SLMs each present a distinct spectrum of trade-offs. Knowledge distillation provides a flexible approach for behavior transfer that is agnostic to architectural specifics, whereas pruning and quantization are structural

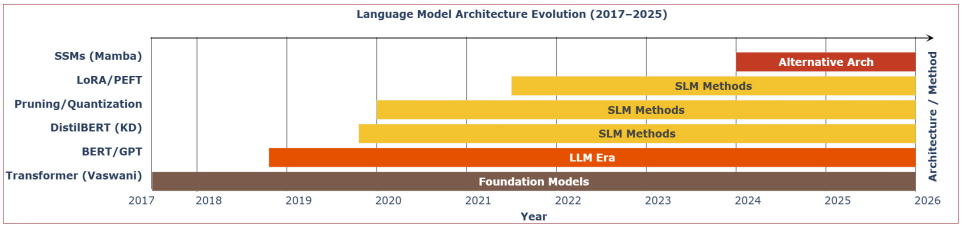


Fig. 6 Timeline of major LLM and SLM architectural developments from 2017 to 2025.

compression methods that reduce model size while preserving the original architecture. PEFT methods, conversely, prioritize efficient adaptation, enabling a single large model to be repurposed for multiple tasks with minimal storage overhead. The optimal choice among these techniques is contingent upon the specific application's constraints regarding accuracy, inference speed, memory footprint, and adaptability.

While the Transformer remains the predominant architectural backbone, it is crucial to acknowledge the emergence of alternative designs with linear-time complexity. Notably, Structured State Space Models (SSMs) like Mamba are proving to be powerful and efficient backbones for SLMs, particularly for tasks involving very long sequences [80]. This rich and evolving ecosystem of techniques underscores the growing importance of SLMs. Having established this foundational landscape of SLM methodologies, the subsequent sections will delve into their specific application within the domain of time series forecasting.

Figure 6 provides a chronological overview of the development of major LLM and SLM architectures from 2017 to 2025. The diagram highlights key milestones such as the introduction of the Transformer architecture, the emergence of BERT, GPT, and the subsequent progression toward lightweight and efficient SLM methods including knowledge distillation, pruning, quantization, LoRA, PEFT, and modern SSMs such as Mamba. This timeline illustrates how the field has evolved from general-purpose foundation models to resource-efficient architectures suitable for edge deployment.

4 Taxonomy of SLM Architectures & Training Paradigms

Having surveyed the foundational methodologies for creating SLMs—namely knowledge distillation, pruning, quantization, and PEFT—in Section 3.2, we now transition to a structured taxonomy of the specific architectures and training paradigms that implement these very techniques. While the prior section introduced the mechanisms, this section provides a hierarchical classification of the resulting model families and their defining implementations, from distilled variants to efficient attention and hybrid systems. We first analyze **"Distilled Transformer Variants,"** a paradigm centered on creating compact, stand-alone models designed to capture and compress the knowledge of larger counterparts [23, 73]. We then transition to **"PEFT"** methods, which are designed for the efficient specialization of large base models for specific

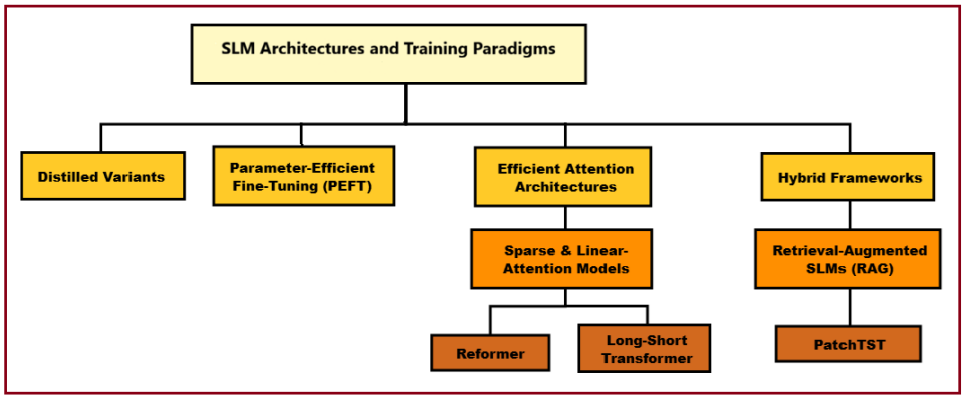


Fig. 7 Taxonomy of SLM architectures and training paradigms where the color gradient from yellow to brown indicates the depth from foundational concepts to specific implementations

downstream tasks [24, 25, 39]. Following this, we investigate efficient attention architectures designed to overcome the quadratic computational complexity of standard Transformers, including sparse and linear-attention models like Reformer and the Long-Short Transformer [21, 30, 31, 81]. Finally, we explore emerging hybrid frameworks, specifically the application of **RAG**, which often involves specialized encoders like **PatchTST** [47, 82]. As illustrated in Figure 7, the taxonomy organizes the four main SLM paradigms—Distilled Transformer Variants, PEFT, Efficient Attention Architectures, and Hybrid Frameworks (e.g., RAG)—into a hierarchical structure, where the color gradient from yellow to brown indicates the depth from foundational concepts to specific implementations.

4.1 Distilled Transformer Variants

As introduced in Section 3.2, knowledge distillation transfers knowledge from a teacher to a student model using soft targets. Building directly on this concept, this subsection categorizes the prominent Transformer variants engineered through this very strategy, detailing their specific architectural modifications and training objectives. knowledge distillation has emerged as a key model compression strategy, leading to the creation of numerous smaller Transformer variants [23]. These models aim to deliver competitive performance despite having significantly fewer parameters, leveraging knowledge transferred from large "teacher" models [73].

DistilBERT and DistilGPT models are prime examples of applying distillation during the pre-training phase [74, 83]. The central objective was to produce smaller, general-purpose language models capable of fine-tuning for diverse tasks, mirroring their larger predecessors. For instance, DistilBERT utilized the BERT model as its teacher, halving the layer count (from 12 to 6) to achieve a 40% reduction in model size. This resulted in a model that operated 60% faster yet retained 97% of the teacher's language understanding capabilities (according to the GLUE benchmark). A key innovation in its training was a **triple loss function**, combining the standard

distillation loss (mimicking the teacher's soft outputs), the **Masked Language Modeling (MLM) loss**, and a **cosine similarity loss** (to align hidden state vectors) [74]. Similar strategies have been employed for distilling **GPT-based models** (like DistilGPT2), often paired with methods like pruning for greater compression [83].

TinyBERT model was introduced by Jiao et al. [84], implements a **two-stage distillation process**. Diverging from DistilBERT, TinyBERT applies knowledge distillation during both the general pre-training and the task-specific fine-tuning phases. Beyond just mimicking the final output, TinyBERT adopts a **layer-to-layer distillation strategy**. This technique involves mimicking the intermediate representations (hidden states) and attention matrices of the teacher (BERT) at each corresponding layer. This more intensive distillation process allows TinyBERT to retain considerable performance on language understanding benchmarks despite its significantly smaller size [84].

4.2 PEFT Paradigms

Recall from Section 3.2 that PEFT methods offer an alternative to structural compression by instead freezing the core weights of a Pre-trained Language Model (PLM) and updating only a minimal parameter set. We now examine the principal PEFT paradigms—adapters, prefix-tuning, and LoRA—as distinct architectural branches within the SLM taxonomy, highlighting their specific implementations. Rather than creating entirely new smaller models, the PEFT paradigm centers on the **efficient adaptation** of a single large PLM for numerous downstream tasks [24, 39]. These methods are characterized by **freezing the vast majority** of the PLM's parameters and training only a minimal, task-specific subset [39, 40, 79].

Adapter method involves inserting small, trainable modules called adapters into the base model's architecture, typically following each main Transformer sub-layer. These modules utilize a **bottleneck architecture**: they first reduce the input dimension, apply a non-linear transformation, and then project the representation back to its original dimension. During adaptation, only the parameters of these adapters (and sometimes the layer normalization layers) are trained [40].

Prefix-tuning was introduced by Li and Liang [41], this method keeps the LLM parameters completely frozen. In this approach, a short sequence of trainable continuous vectors (a **"prefix"**) is prepended to the input. The Transformer model attends to this prefix as though it were a sequence of "virtual tokens." This prefix functions as a set of trainable, task-specific parameters that steer the base model's behavior without modifying its weights. The activations for the prefix positions are read directly from these trainable parameters, influencing the computation of subsequent token activations through the attention mechanism [41].

LoRA is a widely-used technique, based on the hypothesis that the weight update matrix (ΔW) required for adaptation possesses a low **"intrinsic rank."** Therefore, rather than learning the full ΔW matrix directly, LoRA approximates it as the product of two much smaller, low-rank matrices (A and B, such that $\Delta W = BA$). Only these low-rank matrices are updated during training, drastically reducing the number

of trainable parameters. A primary advantage of LoRA is that it introduces **no additional inference latency**, as the low-rank matrices A and B can be algebraically merged with the original weights W_0 post-training ($W' = W_0 + BA$) [39].

4.3 Sparse or Linear-Attention Models

In contrast to the post-hoc compression methods in Section 3.2, efficient attention architectures redesign the self-attention mechanism from the ground up to achieve linear or near-linear complexity for long sequences. Transformer models are very difficult to train due to the need for special optimizers and heuristics [27]. The main goal of these mechanisms is to overcome the quadratic computational and memory complexity ($\mathcal{O}(L^2)$) of standard Transformer self-attention to allow modeling of very long sequences [28]. One of the most important applications is the ability to convert pre-trained or task-specific Transformers into linear versions. This allows for almost complete recovery of the original model performance (accuracy), while the model becomes linear in terms of memory and time [29]. Linear attention methods achieve linear complexity through kernel approximations, recurrent formulations, or fast-weight dynamics, thereby enabling scalable inference with reduced computational overhead. Sparse attention techniques, in contrast, limit attention computation to selected subsets of tokens based on fixed patterns, block-wise routing, or clustering strategies, enhancing efficiency while preserving contextual coverage [85].

Linear attention-based methods replace or approximate the softmax function with linear operations, recurrent formulations, or kernel-based approximations. This simple model can reproduce all the distinctive features of the learning dynamics of full Transformers [85].

sparse attention is a mechanism that restricts attention calculations to a small subset of key/value tokens, rather than a query paying attention to all tokens in the sequence. This restriction (choosing a fixed number) is the same as the operation of "sparseness" or "thinness" [86].

4.3.1 Reformer

Reformer is a pioneering model in the field of sparse attention, demonstrating that high performance can be achieved while preserving the quality of the Transformer architecture through carefully designed sparsity strategies. **Standard Transformer models** face two fundamental limitations when processing long sequences, namely quadratic computational complexity and excessive memory consumption. Reformer overcomes both of these challenges by introducing two principal techniques that deliver performance comparable to standard Transformers, but with significantly greater efficiency and speed on long sequences.

The first technique, **Locality-Sensitive Hashing (LSH) Attention**, is an innovation that addresses the quadratic time complexity problem ($\mathcal{O}(L^2)$) inherent in standard attention mechanisms. In a standard Transformer, computing the attention weight of a query token (Q_i) requires calculating its similarity to all key tokens (K_j) across the entire sequence.

Reformer approximates this costly operation by employing LSH, which is a randomized function that maps vectors (i.e., tokens Q and K) into distinct "buckets." The underlying principle is that similar vectors, those that are close together in the embedding space, are likely to be assigned to the same bucket, whereas dissimilar vectors are mapped to different buckets.

Building on this hashing mechanism, Reformer introduces sparsity by assuming that each query token Q_i should attend only to those key tokens (K_j) that fall within the same hash bucket. This restriction effectively limits the attention computation to a small and proportional subset of tokens. To mitigate the risk of hashing errors, where two semantically important tokens might randomly be assigned to different buckets, Reformer employs multiple rounds of hashing and combines their results. As a consequence of this carefully designed sparsity, the computational complexity of the attention mechanism is reduced from $\mathcal{O}(L^2)$ to $\mathcal{O}(L \log L)$, which represents a substantial improvement for long sequences.

The second technique, **Reversible Residual Layers**, is an innovation that addresses the memory complexity problem during training, which becomes critical when training deep models with a large number of layers (N). In the standard Transformer architecture, residual connections are employed across layers, but performing backpropagation requires storing the activations (outputs) of each layer in order to compute gradients. This requirement causes memory consumption to scale as $\mathcal{O}(N)$. Reformer draws inspiration from the concept of Reversible Networks (RevNets) to eliminate this bottleneck.

Specifically, Reformer divides the activations into two complementary parts (X_1 and X_2) and designs the computation process such that each layer always receives two inputs and produces two outputs. The layer structure is formulated so that the outputs (Y_1, Y_2) can be exactly reconstructed back to the original inputs (X_1, X_2) using a simple inverse function. Thanks to this reversibility property, there is no need to store intermediate activations during the forward pass. Instead, when gradients are computed during backpropagation, the necessary inputs for each layer are reconstructed on the fly.

This technique reduces memory consumption from $\mathcal{O}(N)$, which scales with the number of layers, to $\mathcal{O}(1)$, which remains constant regardless of depth. As a result, Reformer enables the training of models with a larger number of layers and substantially longer sequences using the same hardware resources [30].

4.3.2 Long Short Transformer

Long Short Transformer is a hybrid architecture that strategically separates the self-attention mechanism into two complementary components: a long-range linear component to ensure efficiency and model global dependencies, and a local softmax component to capture fine-grained correlations and maintain accuracy [87] **Long-Range Attention** as a component of the Transformer-LS hybrid attention mechanism aims to model dependencies and relationships that are far apart in the sequence (such as the relationship between a word at the beginning of a document and a word at the end) with linear complexity. This mechanism works based on dynamic projection. In this method, keys (K) and values (V) are mapped (projected) to a

lower-dimensional space instead of remaining in the original dimensional space of the sequence. "Dynamic" means that these mappings (projections) are computed based on the input tokens themselves and are not fixed. This low-dimensional projection makes key-value matrix operations ($K^T V$) much faster and the computational efficiency for long-distance relationships becomes linear (similar to Linear Attention).

Short-Term Attention as a component is used to compensate for the weakness of linearized long-range attention; linearized attention may not model local details accurately. **Local attention limits** the computation to a small window around the current query token. This ensures that strong and subtle dependencies between adjacent tokens (often critical in language and vision) are not lost. Short-term attention often uses the same **softmax formulation** as standard attention, but because the window length (W) is small, its complexity is also linear ($\mathcal{O}(L \cdot W)$) [31].

4.4 Retrieval-Augmented SLMs for Time Series Context

SLMs such as Gemma 2B, Llama 3-8B, and Phi-3 have advantages over giant LLMs due to their smaller parameter size. They require fewer hardware and memory resources, which enables their deployment in limited computing environments (such as edge devices or low-cost servers). They are much cheaper and faster to train and use. They can be better specialized and focused on a specific domain or task (such as time series forecasting) than LLMs, using methods such as instruction tuning. Although SLMs are efficient, like LLMs, they have two major drawbacks when working with time series data that are discussed in the following. First is the continuous tokens versus discrete text. SLMs are trained on discrete linguistic tokens, while time series data is continuous. For consistency, the data must be transformed into tokens (such as patching or quantization), but this process may lose important information. The second issue is the limitation of context window. Even with large windows, models cannot access the entire history of the data. In time series, seasonal patterns or rare anomalies may occur much further back than the current context window [16]. Time series data present three main challenges that traditional models, and even pure LLM-based approaches, struggle to address:

Complex spatiotemporal dependencies: Patterns are intertwined across temporal and multivariate dimensions.

Distribution shifts: The underlying rules governing the data change over time (e.g., sudden changes in market behavior or anomalies).

Lack of external knowledge: Pre-trained models (such as LLMs/SLMs) lack explicit knowledge of domain-specific historical patterns and contextual information, both of which are critical for accurate time series decision-making.

To address these challenges, retrieval-augmented paradigms–including agentic variants–have been proposed for time series modeling [47]. The core principle is to equip SLMs with access to a dynamic external memory (e.g., prompt pools of historical patterns), thereby mitigating the context-window limitation and enabling zero-shot adaptation to distributional shifts. The detailed architecture of an agentic RAG system and its multi-level retrieval strategies are examined in Section 5.3. Meanwhile, the remainder of this subsection introduces the time series encoders (e.g., PatchTST) and channel independence strategies commonly employed in such pipelines. For retrieval, a

time series encoder (such as PatchTST or TimeSformer) is typically used that Transforms the current time series input into a vector. This vector is then searched in the knowledge base for the most similar knowledge prompts [47]. **PatchTST models** are inspired by the **Vision Transformer (ViT)** architecture in image processing; however, rather than dividing images into pixel patches, they segment time series data into fixed-length patches for representation learning. The input time series $\mathbf{X} \in \mathbb{R}^{C \times L}$ (where C is the number of channels/variables and L is the length of the history) is divided into patches. Each patch $\mathbf{P}_i$ is a subseries of length P. The patching process is characterized by a patch size (P) and a stride (S). If the stride $S = P$, the patches do not overlap; otherwise, they overlap. By applying patching, the input is transformed from a long sequence of individual points to a short sequence of subseries-level tokens. The final number of tokens $N = \lfloor (L-P)/S \rfloor + 1$ is much less than the original length L. Patch Embedding: Each patch $\mathbf{P}_i$ is transformed into a token embedding vector via a linear or other embedding layer, which is then fed into the Transformer. This method reduces the time complexity of the attention mechanism from $O(L^2)$ to $O(N^2)$. Since $N \ll L$, this leads to a drastic saving in computation and memory, and enables the model to see a longer history at a fixed computational budget. Local semantics preservation: instead of processing individual data points, the model considers local patterns within each patch (e.g., small trends, fluctuations) as a semantic unit. This improves the model's understanding of time series features [82]. The **channel independence method** is a crucial design strategy for multivariate time series processing, introduced with the aim of increasing generalization and reducing overfitting in deep models (especially Transformers and SLMs). The method starts by decomposing the multivariate time series into C univariate time series. Suppose our input data is a matrix $\mathbf{X} \in \mathbb{R}^{C \times L}$, where C is the number of variables/channels (e.g., temperature, pressure, humidity) and L is the length of the history (look-back window). This matrix is decomposed into C independent sequences: $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_C$ where each $\mathbf{X}_i \in \mathbb{R}^L$ represents a time series of a single variable. After decomposition, each sequence $\mathbf{X}_i$ is fed into the model kernel separately, but with a common base model. A single base model (which in PatchTST is a Transformer and can be an efficient SLM) is defined. This model has a set of parameters Θ. Each univariate sequence $\mathbf{X}_i$ passes through the base model separately:

$$\mathbf{H}_i = \text{BaseModel}(\mathbf{X}_i; \boldsymbol{\Theta})$$

where $\mathbf{H}_i$ is the hidden representation. The most important point is that all channels $(\mathbf{X}_1, \ldots, \mathbf{X}_C)$ use the same set of weights $\boldsymbol{\Theta}$. The model does not have any dedicated layers for combining variables in Transformer layers (multi-head attention, feed-forward networks). This weight sharing greatly reduces the computational and memory burden and allows small SLMs to train the model with fewer resources, since the number of parameters is only $O(\text{size})$ instead of $O(C \times \text{size})$. Final output combination only after deep and independent processing by the model core, the outputs of the channels are combined to produce the final prediction. A linear layer or a simpler layer is used to combine $\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_C$ and produce the prediction $\hat{\mathbf{Y}}$.

$$\hat{\mathbf{Y}} = \text{OutputLayer}(\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_C)$$

This final layer is the only place where the model learns the relationships between variables (and that too only based on the temporal representations it has previously learned) [44]. The application of these encoding strategies within full retrieval-augmented forecasting systems, including in-context fine-tuning and multi-agent orchestrations, is elaborated in Section 5.3.

4.5 Hybrid Neuro-Symbolic Systems

Traditional rule-based KBC methods provide verifiable and interpretable reasoning but lack the flexibility required to generalize across diverse scenarios. In contrast LLMs, which are built on neural networks and deep embedding techniques, have demonstrated an unparalleled ability to learn complex linguistic patterns and generate fluent text. The overarching objective is therefore to combine the expressive and representational strengths of LLMs with the logical rigor and precision of symbolic reasoning, thereby improving both the reliability and flexibility of inferring new facts, which can be viewed as a form of prediction [50]. This motivation naturally leads to the development of **Hybrid Neuro-Symbolic Systems**. In such systems, the neural part (LLM/embedding) is responsible for processing observations from the environment and predicting (i.e., selecting) the most appropriate next action. By leveraging its extensive pretrained knowledge, the LLM interprets the task context and determines an effective strategy. Complementing this capability, the symbolic section consists of tools that perform precise and deterministic reasoning, such as calculators, sorting algorithms, or navigation systems. Logical constraints—including physical laws, market regulations, or legal rules—are incorporated into prompts or embedded within specific layers of the LLM to ensure that the generated predictions remain logically valid.

Within this neuro-symbolic pipeline, the LLM first translates unstructured textual inputs into structured symbolic representations, such as programming commands or knowledge-base queries. A symbolic system then carries out the final prediction or inference with formal accuracy. The LLM produces initial candidate predictions, the symbolic module verifies and refines them, and the validated results are subsequently fed back to the LLM to guide further prediction refinement. This iterative feedback loop plays a central role in maintaining accuracy and consistency [51, 52, 88]. Consequently, to achieve reliable predictions using LLMs or SLMs, it is insufficient to rely solely on embedding-based representations. Neuro-symbolic approaches that integrate neural and symbolic components provide a principled pathway for combining statistical learning power with logical precision, enabling predictive systems to infer not only what to predict, but also how and why, supported by explicit logical justification [51, 88]. Hybrid pipelines specifically tailored for time series forecasting—where statistical and deep learning components are integrated with language models—are discussed in Section 5.4.

5 Applications to Time Series Prediction

The emergence and spread of language models has transformed the forecasting perspective of time series from purely numerical modeling to meaningful, context-based

systems [38, 45, 89–91]. Unlike traditional statistical methods that rely solely on historical numerical patterns, new approaches attempt to exploit the reasoning capabilities, prior knowledge, and semantic understanding of linguistic patterns to model complex temporal dynamics [44, 47, 92–94]. A systematic review of the research literature shows that recent advances in this field can be classified into four main categories:

1. Prompt Engineering for direct prediction: Focuses on transforming time series data into text space and inferring language models without changing the underlying architecture [35, 36].
2. Fine-Tuning of language models: Adapting the representation space of models to the temporally continuous data through efficient windowing strategies and adapters [38, 95, 96].
3. Retrieval-Augmented models: Overcoming the limitations of static knowledge and textual windows by connecting the model to dynamic knowledge bases of historical patterns [43–45, 47].
4. Hybrid Pipelines: Combine the abstract reasoning power of language models with the numerical precision of statistical and classical models in a single architecture [49, 97].

Table 6 provides a summary of the features, indicator models, and analysis of the strengths and weaknesses of each of these four approaches. To systematically analyze the landscape of SLM applications in time series forecasting, we present three complementary visualizations that progressively detail the taxonomy and methodological approaches in this emerging field.

Also, Figure 8 provides a high-level sunburst diagram organizing the four dominant paradigms—Prompt Engineering, Fine-tuned SLMs, Retrieval-Augmented Models (RAG), and Hybrid Pipelines—along with their representative models. The hierarchical structure, depicted through a yellow-to-brown gradient, reveals both the breadth of approaches and the growing ecosystem of SLM-based forecasting solutions.

5.1 Prompt Engineering for Direct Time Series Forecasting

In the paradigmatic transition from classical statistical and deep learning forecasting models toward LLMs and SLMs, prompt engineering serves as the fundamental basis for translating time series data into the semantic and inferential space of language models [32]. In the novel approach of "direct forecasting," the time series prediction problem transforms from a mere numerical regression into a task of "text generation" or "sequence completion" [34]. Within this framework, historical data, covariates, and forecasting instructions must be structured within textual prompts in such a manner that the model can comprehend temporal dependencies and statistical patterns without requiring fundamental modifications to the Transformer architecture [35]. The success of this approach is heavily dependent on strategies employed in the domains of data representation and tokenization, covariate structuring, and reasoning instruction design [33, 36]. Building on this foundation, Figure 9 drills down into the prompt engineering paradigm, offering a granular decomposition of its core components: tokenization techniques, covariate integration strategies, multivariate handling methods,

Table 6 Overview of paradigms for empowering time series analysis with SLMs.

Category	Representative Models	Solved Challenge	Pros	Cons
Prompt Engineering	PromptCast, Chronos, ChatTime, TOKON, MultiCast.	Addressing modality gap between continuous data and discrete tokens; enabling zero-shot capabilities.	**Generalizability:** High on unseen data. **Flexibility:** No retraining needed.	**Sensitivity:** Degradation with bad prompts. **Overhead:** Increased sequence length.
Fine-Tuned SLMs	MOMENT, TsLLM, UniTime, Time-Prompt, TEMPO, GPT4TS.	Overcoming inability of general models to comprehend specific temporal dynamics.	**Accuracy:** High in specific domains. **Alignment:** Good text-time series adaptation.	**Cost:** Needs retraining resources. **Risk:** Potential overfitting on noise.
RAG	TimeRAG, TS-RAG, FinSrag, Agentic RAG.	Mitigating context window limits and recalling distant historical patterns.	**Memory:** Access to long history. **Interpretability:** References history.	**Noise:** Accuracy drop if retrieval fails. **Complexity:** Needs search mechanism.
Hybrid Pipelines	StreamTS, SMETimes, TimeCMA, TEST, MLTA.	Resolving imbalance between semantic reasoning (LLM) and numerical precision.	**Synergy:** Semantic and Numerical power. **Robustness:** Task decomposition helps.	**Complexity:** Syncing disparate modules. **Maintenance Cost:** Managing multiple models.

and instruction engineering approaches. This detailed taxonomy highlights the rich design space available to practitioners when adapting SLMs for time series tasks without modifying model weights.

5.1.1 Advanced Tokenization and Value Representation

One of the fundamental challenges in utilizing language models for direct forecasting is the misalignment between the continuous, infinite nature of time series data and the discrete, finite vocabulary-based nature of language models [35]. Standard text tokenizers (e.g., BPE or SentencePiece), when encountering decimal numbers, fragment them into arbitrary subword units or individual characters; this leads to an increased input sequence length and a reduction in the model's ability to extract mathematical patterns [33]. To address this limitation, recent literature proposes innovative solutions.

In one approach, the ChatTime model proposes that time series should be treated as a "foreign language" for the language model. In this method, continuous values

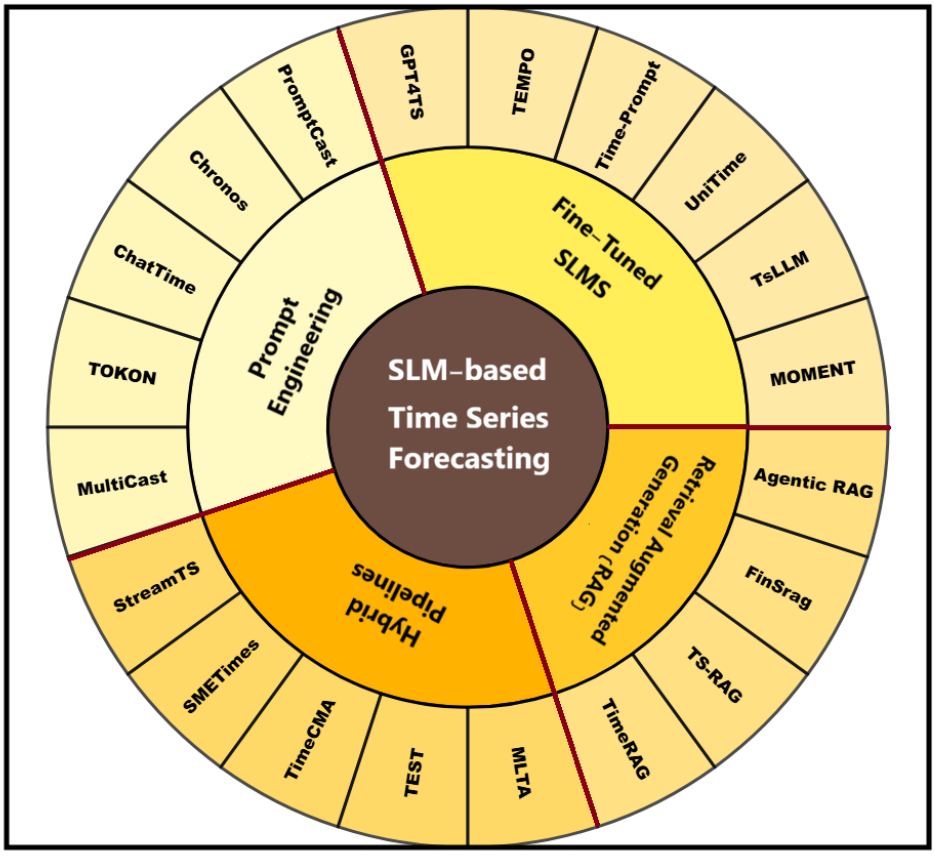


Fig. 8 Four Main Paradigms of SLM-based Time Series Forecasting with Representative Models. This sunburst diagram illustrates the hierarchical taxonomy of the four primary approaches for applying SLMs to time series prediction: (1) *Prompt Engineering* for direct prediction using models such as PromptCast, Chronos, ChatTime, TOKON, and MultiCast; (2) *Fine-Tuned SLMs* including MOMENT, TsLLM, UniTime, Time-Prompt, TEMPO, and GPT4TS; (3) *Retrieval-Augmented Models (RAG)* featuring TimeRAG, TS-RAG, FinSrag, and Agentic RAG; and (4) *Hybrid Pipelines* encompassing StreamTS, SMETimes, TimeCMA, TEST, and MLTA. The color gradient from light yellow (outer ring) to dark brown (center) represents the hierarchical structure, with each paradigm distinguished by its unique shade to facilitate visual differentiation.

are first converted into a finite set of discrete values through normalization and binning techniques. Subsequently, these values are encapsulated using special marker characters (e.g., `###`) and embedded as independent words within the prompt ((e.g., `###0.2835###`) so that the model can distinguish clearly between the textual context of the instruction and the numerical data. This technique ensures that each numerical value, regardless of its decimal precision, is represented by a single token, leading to a significant reduction in token consumption and computational costs [35].

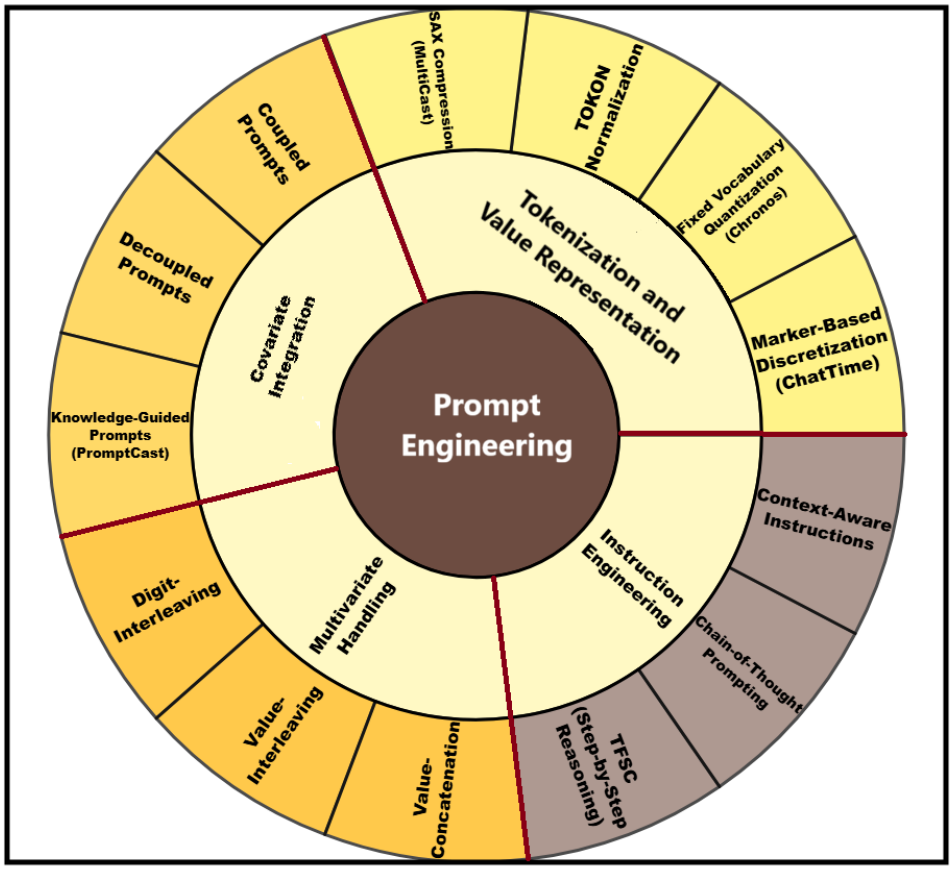


Fig. 9 Prompt Engineering Taxonomy for Time Series Forecasting. This comprehensive sunburst visualization decomposes the Prompt Engineering paradigm into four fundamental categories: (1) *Tokenization & Value Representation*, encompassing techniques such as Marker-Based Discretization (ChatTime), Fixed Vocabulary Quantization (Chronos), TOKON Normalization, and SAX Compression (MultiCast); (2) *Covariate Integration*, including Coupled Prompts, Decoupled Prompts, and Knowledge-Guided Prompts (PromptCast); (3) *Multivariate Handling*, featuring Digit-Interleaving, Value-Interleaving, and Value-Concatenation strategies; and (4) *Instruction Engineering*, comprising TFSC (Step-by-Step Reasoning), Chain-of-Thought Prompting, and Context-Aware Instructions. Each technique is color-coded to reflect its category, with the yellow-to-brown gradient indicating the depth of the hierarchical structure.

Another strategy, the Chronos framework, adopts a minimalist approach wherein time series values are converted into a set of "time series tokens" from a fixed vocabulary via mean scaling and uniform quantization. In this method, the language model is trained directly on these discrete tokens, and forecasting is performed as a classification problem over the vocabulary. The use of the cross-entropy loss function in this

structure allows the model to learn arbitrary probabilistic distributions, including multimodal ones, without relying on the parametric assumptions common in statistical models [34].

To further optimize the tokenization process within prompts, the TOKON method is proposed, aiming to map time series numbers directly to integers that exist precisely within the model's tokenizer dictionary. This method utilizes a scaling parameter optimized via a 1D search algorithm to convert each data element into a single unit token. Experimental results demonstrate that this method can reduce the number of tokens by 2 to 3 times and improve the RMSE error in multi-step forecasting by 7% to 18%, depending on the prompt type [33]. Finally, for scenarios with severe context window limitations, the MultiCast method suggests using the Symbolic Aggregate Approximation (SAX) technique; in this approach, normalized numerical values are quantized into alphabetic or digital symbols, allowing repetitive patterns to be presented to the model using fewer tokens and with lower complexity, which significantly increases inference speed [37].

5.1.2 Covariate Integration Mechanisms

Beyond the representation of numbers, the overall prompt architecture and the integration of historical data with covariates (e.g., calendar information, holidays, and related indicators) play a significant role in model performance. Research indicates that selecting a suboptimal prompt structure, even with rich information, can lead to reduced accuracy [36]. Three primary architectures have been identified and evaluated for this purpose as follows:

1. **Coupled Prompts:** In this architecture, each historical data point (y_t) is directly paired with its corresponding covariate (x_t) and placed in the prompt as a regular, connected sequence (e.g., format $x_1 : y_1, x_2 : y_2, \dots$). For future forecasting, only the future covariates ($x_{T+1}, \dots$) are appended to the end of the prompt. Due to its high compatibility with the autoregressive nature of language models, which perform learning as a continuous sequence, this structure has demonstrated more stable performance and lower error rates compared to other methods in multiple experiments on real-world data (e.g., influenza and air traffic data) [36].
2. **Decoupled Prompts:** In this approach, the list of target values and the list of covariates are presented as separate blocks within the prompt. The core idea is that explicit separation helps the model distinguish between input features and the forecasting target. However, empirical evidence suggests that this method often performs more poorly compared to coupled prompts, likely due to the model's difficulty in establishing correlations between separate lists over long context windows [36].
3. **Contextualized & Knowledge-Guided Prompts:** Methods such as PromptCast and knowledge-guided approaches attempt to enrich the prompt using natural language templates (e.g., "The temperature on date X was Y") or by adding domain descriptions (e.g., "This data relates to electricity consumption") [32, 36]. Although these methods can be beneficial in zero-shot scenarios and

for very large models, research indicates that adding excessive textual information can lead to noise or "data leakage," and in smaller models, it may cause a decrease in accuracy [36].

5.1.3 Multivariate Handling via Token Multiplexing

Most language models are inherently designed to process univariate text sequences, posing a significant challenge for direct multivariate time series forecasting. To overcome this limitation, the **MultiCast** method proposes "token multiplexing" strategies to serialize different dimensions into a single text string. In this approach, **Digit-Interleaving** places digits corresponding to different dimensions at each time step alternately side-by-side so the model can learn precise local correlations at the digit level. **Value-Interleaving** organizes the entire numerical value of each dimension sequentially for each time step in the prompt, which is more suitable for time series with varying scales. Additionally, **Value-Concatenation** concatenates the values of different dimensions one after another with a separator (such as a comma) to preserve the distinction between dimensions. Results show that using these methods in language models (such as LLaMA-2) enables the understanding of inter-dimensional correlations without the need for complex multi-channel architectures [37].

5.1.4 Instruction Engineering and Reasoning

The quality and stability of direct forecasting are heavily dependent on the clarity and detail of the "instruction" within the prompt. Despite their high capabilities, language models occasionally suffer from hallucinations or calculation errors when performing simple algebraic operations or understanding long-term trends. To address this challenge, the novel **"Time Series Forecasting with Care (TFSC)"** prompt has been developed. This prompt explicitly and detailedly requests the model to perform the analysis "step-by-step," focus on identifying seasonal patterns and trends, and execute calculation operations with precision and accuracy at every stage. Using phrases such as *"Analyze the time series step by step... ensuring precision and accuracy"* in the prompt compels the model to activate a Chain-of-Thought process. Empirical evidence suggests that using TFSC alongside appropriate normalization strategies (like TOKON) can reduce the RMSE error in multi-step forecasting by approximately 18% and ensure model stability across different time horizons [33]. Furthermore, preserving text Q&A capabilities during instruction fine-tuning is essential for maintaining the model's inferential ability, and its removal can lead to performance degradation in forecasting [35].

5.2 Fine-tuned SLMs on Time Series Datasets

With the emergence of large language models and small language models as powerful tools in time series analysis, adapting these models to accurately predict continuous values over long time horizons has become one of the main research challenges. Unlike typical linguistic tasks that focus on generating discrete tokens, time series forecasting requires a deep understanding of temporal dependencies, seasonal patterns, and changes in data distribution over time. Therefore, simply using pretrained models is

not enough, and it is necessary to employ targeted fine-tuning strategies to align the model representation space with the dynamic nature of time series. This adaptation focuses on solving two critical challenges:

5.2.1 Optimal Windowing Strategies

To encode temporal patterns in a form that is understandable to Transformers, efficient approaches for multi-step forecasting are required to preserve forecasting accuracy over longer prediction horizons. In the following, the impact of different data transformation methods and input structuring techniques on the performance of SLMs in this domain is analyzed [24, 95, 98].

5.2.2 Windowing Strategies

The primary hurdle in adapting language models for time series is the "modality gap" between continuous numerical data and the discrete token embeddings used by Transformers. To address this issue, optimal windowing strategies must extend beyond simple segmentation and incorporate robust temporal alignment techniques at the input stage [24]. In this context, a patch of a time series can be regarded as a natural analog to a token in language models, and **patching** has been shown to improve model performance. Moreover, this improves inference speed as the number of tokens being fed is reduced by a factor of the patch length. In language models **windowing** a time series means converting a historical sequence into a format that the model can process as input [98, 99]. In the realm of time series prediction modeling, the window size (w) is a critical hyperparameter that determines the number of time units included in each example provided to a learning model. This hyperparameter is crucial because it allows the learning model to recognize both long-term and short-term trends, as well as seasonal patterns, while reducing sensitivity to random noise. Choosing the right window size is critical to capture long- and short-term trends and seasonal patterns without making the model sensitive to random fluctuations or leading to overfitting. This technique converts a time series into a labeled dataset such that each window contains a set of past observations that are considered as input to predictive models. This approach involves dividing the time series data into smaller sections of fixed size (w). The term sliding refers to the process of moving the window along the series by a specified step (p) to the right to build a training data set. The observation immediately after the end of the window is defined as the target value that should be predicted based on the previous window. The results show that increasing the window size (w) generally improves the evaluation criterion values up to a point of stabilization, beyond which further increases do not have a significant impact on the prediction accuracy. The concept of an optimal window size is not absolute and its effect strongly depends on the characteristics of the dataset and the machine learning model used. It should be noted that the use of very large window sizes may not be justified by the increase in the input data size and, consequently, the increase in the training time of the models [42].

Fixed-Window Patching: To mitigate the computational bottlenecks of Transformers and accommodate the heterogeneity inherent in time series data, the MOMENT model employs a hybrid windowing strategy comprising **fixed input**

length and **non-overlapping patching**. **MOMENT model** enforces a strict constraint where the input is a univariate time series of a fixed length $T = 512$. This design choice is predicated on empirical evidence suggesting that a 512-point window is adequate for accurate long-horizon forecasting and encompasses the duration of most classification datasets. Instead of processing individual time points as distinct tokens—which would incur prohibitive computational costs—the 512-point window is segmented into N disjoint, non-overlapping patches. This patching mechanism reduces the Transformer's memory footprint and computational complexity quadratically, while simultaneously enabling the model to capture local dependencies within patches and global dependencies across them. To reconcile real-world variations in sequence length with this fixed-window constraint, MOMENT adopts specific adaptation strategies: sequences exceeding 512 points are sub-sampled, whereas shorter sequences are left-padded with zeros to reach the required length. For multivariate data, the model utilizes a channel-independent strategy, processing each variable separately along the batch dimension without explicit inter-channel dependencies at the input stage. Furthermore, prior to patching, the input window undergoes Reversible Instance Normalization (RevIN) to rescale and center the data, thereby handling distributional shifts effectively [95].

Sliding Window / Segmentation: This technique aims to achieve temporal alignment and optimize the model performance in both computational and semantic dimensions. In this approach, instead of processing the entire time sequence as a single vector, a **sliding window** operation is used to divide the time dimension into smaller **segments**. Suppose the input is a time series of length L (with dimensions $B \times L$ where B is the batch size). By applying this technique, the input matrix is restructured into a tensor of dimensions $B \times N \times K$, where N is the number of segments and K is the length of each segment. This structural transformation converts the raw data into a format that is more understandable for the input layers of the Transformer models and allows for the extraction of local features. One of the main obstacles in using LLMs for long time series is the computational complexity of the self-attention mechanism, which increases quadratically ($\mathcal{O}(L^2)$) with the input length. Using the segmentation strategy, the attention and feed-forward network computations are performed on smaller segments instead of the entire length L. This reduces the computational complexity to $\mathcal{O}(N^2K)$. This reduction in complexity enables the model to process long-term dependencies without encountering computational explosion or memory limitations, which is critical for accurate forecasts over long horizons. In addition to the computational advantages, this technique acts as a bridge for temporal alignment. In the input stage of language models, it is essential to transform raw data into structured formats to ensure that the LLM can process the semantic features and mathematical properties underlying the time series. The use of sliding windows allows for the generation of multiple samples of a single time series, allowing the model to learn dynamic patterns and distributional changes at different scales. This process reduces the modality gap between discrete linguistic structures and continuous temporal patterns, providing a more accurate semantic representation for subsequent layers of the model [98, 100].

Symmetric Windowing Strategy: Unlike many forecasting methods that keep the input look-back window length fixed (e.g., 96 time steps) and output different forecast horizons, this method uses a dynamic and symmetric strategy. The core of the windowing strategy in this study is that the length of the input window (denoted by L) is set exactly equal to the length of the forecast horizon (denoted by H). The authors state: "In our forecasting setting, the time lag is the same as forecasting horizon." This means that if the model is to forecast 7 days into the future, it receives exactly the past 7 days as input context. In this method, the model learns a function f that maps equal-sized windows of past data (including numerical time series X and text data Y) into the future:

$$[X_{t-L}, \ldots, X_{t-1}, Y_{t-L}, \ldots, Y_{t-1}] \xrightarrow{f} [\hat{X}_t, \ldots, \hat{X}_{t+H-1}, \hat{Y}_t, \ldots, \hat{Y}_{t+H-1}]$$

where L is the input time lag and H is the forecast horizon. In each time window I, the model processes two types of data:

1. Time series (C): where C is the number of variables.
2. Text (N): where N is the length of the tokenized text at each time step.

After being converted to Embedding, these two inputs are combined (concatenate) and entered into the model [91].

Flexible Token-Based Windowing: This approach fundamentally reimagines time series processing by employing a high-fidelity, patch-based encoding strategy designed to bridge the modality gap between numerical data and text. Unlike standard LLM tokenizers that inefficiently split numbers into multiple tokens, **TsLLM** segments the input time series into fixed-length patches (e.g., patch size $p = 32$). These patches are then projected into dense vector representations via a specialized encoder-decoder architecture pretrained with a β-VAE objective, effectively compressing the temporal signal while preserving fine-grained information These dense vectors are subsequently mapped to the LLMs embedding space using lightweight Multilayer Perceptron (MLP) adapters, allowing the model to treat time series segments as semantic "pseudo-tokens" seamlessly interleaved with text. A critical distinction of this strategy is its input flexibility. While models like MOMENT enforce a rigid Fixed-Window constraint (typically $T = 512$ time points) and rely on strategies like left-padding or sub-sampling to fit variable-length data into this fixed horizon TsLLM eliminates such constraints. In TsLLM, the sequence length is limited only by the LLM's maximum context window (e.g., 4096 tokens), enabling the processing of variable-length time series mixed with extensive textual instructions and context. Furthermore, to address the challenge of magnitude perception, TsLLM incorporates a Scale-Shape Decomposition technique. The input patches are first normalized via Z-score to extract the shape (waveform patterns) for stable encoding. Crucially, the normalization statistics (mean μ and standard deviation σ) are not discarded but are encoded separately as scale embeddings. These scale embeddings are then injected back into the model (via additive combination), granting the LLM the ability to comprehend absolute magnitudes and scale-dependent semantics [38, 90, 96].

Variable Look-back Window: UniTime uses a flexible patching strategy to handle input lengths of different domains. In this method, the patch length is fixed, but the stride is variable. The formula for the number of tokens is: $N_\tau = \lceil \frac{L_\tau - P}{S_\tau} \rceil + 1$. This means that if a domain requires a longer input window, UniTime increases the stride to control the number of tokens, rather than generating more tokens (which is computationally expensive). In contrast, TsLLM uses a fixed stride and patch. To handle variable lengths, TsLLM simply increases or decreases the number of tokens input to the language model (limited to the Context Window of the model). The TsLLM approach is closer to the structure of the language (more tokens for longer text), while the UniTime approach is more engineered to control the input dimensions. This model uses text as an aid to domain identification. The text input consists of short descriptions (e.g., "Exchange rate data with daily sampling rate") so that the model does not get confused and understands which domain the input data belongs to. In the former approach, the text plays a central role and is aligned with the time series. The text and the time series are input alternately, and the model can produce text output (descriptions) or numerical output (predictions). In UniTime, the text is only the input, and the output is always numerical. A specific problem in multi-domain learning is that some domains (with simple patterns) are learned early and overfit. To solve this problem, UniTime uses a masking strategy to force the model to work harder on simpler domains, balancing the speed of convergence [89].

Privileged Prompt-Based Windowing: The windowing and input processing approach in this model is designed to transfer knowledge from a LLM to a lightweight student model. This approach has two main aspects:

1. **Textual Windowing for Teacher:** In this model, the historical window of the time series is not given directly to the language model numerically the numerical values in the input window (X) are converted into a sequence of text strings (String Sequence) to adapt to the nature of the input to the language models. In addition to historical data, this text window is combined with Privileged Knowledge. This knowledge includes textual descriptions and information about the future (ground truth prompts) that are only available during training. This structure allows the teacher model to have a deeper understanding of the patterns in the time window.
2. **Numerical Windowing for Student:** The student model (which is to be used at inference time) uses a standard and efficient windowing approach.

   Purely historical input: Unlike the teacher, the student model receives only a numerical historical window (X) and does not have access to textual prompts or future information.

   Knowledge distillation goal: The main goal of the windowing approach in TimeKD is for the student model to learn how to produce representations from its limited, numerical window that resemble the rich, textual representations of the teacher model [101].

Causal Selection Windowing-Augur: In contrast to the previous approach that focused on representation enrichment, the Augur model uses LLM to "filter and select" the content of the input window.

1. **Textual Window for Causal Discovery:** In the first stage (teacher LLM), the goal is not to make predictions, but to discover cause-and-effect relationships between variables. The textual window is given to the LLM as part of a prompt, and the model analyzes the simultaneous changes in values in the text to extract the causal DAG.
2. **Causally-Filtered Window:** In the second stage (student/forecaster), the extracted graph is then used to optimize the input window. Instead of the historical window including all available covariates, only those variables that are identified as "parents" or direct causes of the target variable in the causal DAG are kept in the input window. This makes the input window of the predictive model pure and free of spurious correlations, which usually reduce accuracy in complex time series [92].

5.2.3 Multi-Step Forecasting

In the development of Time Series Foundation Models (TSFMs), adapting the outputs of LLM to the needs of numerical forecasting over long time horizons is a key challenge. The existing literature has addressed this issue through four main strategies:

1. modification of the output layers
2. auto-correlated formulation
3. efficient temporal adapters and fine-tuning
4. input optimization and knowledge distillation

Forecasting Head Tuning: In this strategy, it is assumed that the pre-trained LLM is capable of extracting complex features and only needs to map these features to the time series numerical space. The main body of the model is usually frozen and only one output layer or module is trained. One Fits All (GPT4TS) model uses a minimalist approach in which all the Transformer layers and embedding layers of the GPT-2 model are frozen. To perform the forecast, only one simple linear layer without an activation function is added at the end of the model. This linear layer is responsible for converting the feature vector extracted by the model (with dimensions D) directly into the length of the forecast horizon (H). Experimental results show that this simple layer is sufficient to preserve the general knowledge of the model and prevent overfitting in time series data [96]. MOMENT model uses a "Reconstruction Head" in the pre-training phase to recover the masked patches. In the fine-tuning phase for prediction, this head is replaced by a "Forecasting Head." This new head first transforms all output patches ($N \times D$) into a flattened vector and then maps it to the dimensions of the forecast horizon (H) through a linear projection layer. This method allows the model to process the global relationships between all input patches in one step [95]. UniTime model uses a Lightweight Transformer and a final

linear layer to solve the challenge of varying forecast lengths across domains (cross-domain). This decoder always produces a fixed and maximum length output (O) (e.g., 720 time points). At runtime, depending on the needs of the target domain (e.g., if a 96-hour forecast is needed), the generated output is truncated to fit the desired horizon exactly. This mechanism allows the model to handle different domains without changing the structure [89]. Hybrid-MMF model is designed to process text and time series simultaneously and uses a multi-head MLP module in the output layer. This module receives mixed representations (which include text and time series information) and projects them to future values of the time series. This design allows contextual information (such as news or reports) to directly influence the predicted values [91]. In the Time Series Foundation Model (TimesFM), "Input and Output Residual Blocks Tuning" strategy is used to fine-tune the model. This method is inspired by the GPT4TS model protocol. In this method, the main body of the model (the Transformer layers that contain the general knowledge of the time series) is usually kept frozen or modified at a much lower learning rate, and the main focus of training is on the following two parts:

1. Input Residual Block: This block is responsible for converting the raw time series slices into model dimension vectors for input to the Transformer layers. This block is actually a MLP with one hidden layer and one hopping connection that places the raw time series slices into the model embedding dimension. This block maps the hidden representations to the prediction horizon. By updating the weights in these specific blocks using our training data, we align the input processing and output generation of the model with the specific statistical distribution of our target data, while taking advantage of the strong inductive biases encoded in the frozen Transformer layers.
2. Output Residual Block: Similar to the input block, this is also an MLP with a residual connection. Its job is to invert the input operation; that is, it takes the vector with the dimensions of the model and converts it into a prediction vector with length h.

By fine-tuning these blocks, you actually change how "specific inputs to the new dataset are interpreted" and "how the final output is produced," while leaving the knowledge of temporal feature extraction (stored in the intermediate layers of the Transformer) intact [99].

Autoregressive forecasting: This approach, while preserving the original nature of language models, models forecasting as a sequential generation process. TsLLM model: In this model, time series forecasting is defined as a "next-token prediction" or natural continuation task. The time series is divided into patches of fixed length, and each patch is considered a token. At each time step, the model predicts the next patch and then appends this predicted patch to its input to produce the next patch. This process continues until the entire forecast horizon is covered, allowing the model to use the full capacity of its Context Window [38].

SE-LLM model (inference): This model uses an autoregressive inference scheme. Although its overall architecture includes an encoder and decoder, when generating

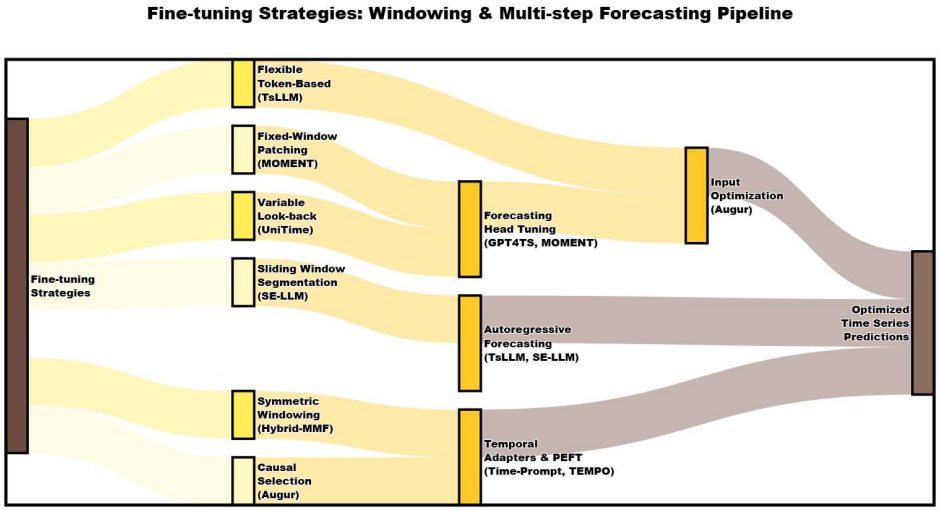


Fig. 10 Fine-tuning Strategies: Windowing and Multi-step Forecasting Pipeline. This Sankey diagram illustrates the methodological flow of fine-tuning strategies for SLMs in time series forecasting. The pipeline originates from two main branches: (1) *Windowing Strategies*, including Fixed-Window Patching (MOMENT), Sliding Window Segmentation (SE-LLM), Symmetric Windowing (Hybrid-MMF), Flexible Token-Based (TsLLM), Variable Look-back (UniTime), and Causal Selection (Augur); and (2) *Multi-step Forecasting Approaches*, comprising Forecasting Head Tuning (GPT4TS, MOMENT), Autoregressive Forecasting (TsLLM, SE-LLM), Temporal Adapters & PEFT (Time-Prompt, TEMPO), and Input Optimization (Augur). The flow converges toward optimized time series predictions, with node colors transitioning from light yellow (input strategies) through amber (intermediate processing) to dark brown (final output), representing the progression through the fine-tuning pipeline.

the output, the predicted values at previous time steps are recursively fed as new inputs. This method helps the model to correct the prediction error over time and track long-term dynamic trends more accurately [100].

Finally, Figure 10 employs a Sankey diagram to trace the methodological pipeline of fine-tuning approaches, from windowing strategies through multi-step forecasting techniques to final predictions. The flow visualization captures the sequential decision points and the interplay between input processing methods and forecasting architectures, with color transitions reflecting the progressive transformation of data through the fine-tuning pipeline.

Temporal adapters and efficient fine-tuning: These methods inject lightweight modules into the model architecture to learn the dynamic and specific features of the time series without changing the original weights.

Time-Prompt model: This model uses the LoRA technique to fine-tune the LLM parameters. Low-rank matrices are added to the attention layers to adapt the semantic space of the model to understand continuous values. Learnable "soft prompts" are also used to guide the model's behavior for specific time series tasks. This approach allows the model to perform well with less than one percent of the trainable parameters [93].

SE-LLM model (adapter): This model introduces an architectural module called Time-Adapter, which consists of parallel LSTM layers. This adapter is placed inside the attention blocks of the language model. The goal of this adapter is to learn short-term and long-term temporal dependencies that text Transformer models (designed for discrete sentences) may miss in continuous data. The output of this adapter is combined with the output of the attention layer to produce richer temporal features [100].

TEMPO model: This model is a combination of time series decomposition and LoRA. First, the time series is decomposed into "Trend," "Seasonal" and "Residual" components. Then, for each component, LoRA is used to adjust the parameters of the GPT-2 model so that the model can adapt to changes in the distribution of data (Distribution Shift). In addition, a Prompt Pool is used to select appropriate prompts based on the similarity of the input data and help the model [94].

Reasoning-Based Input Optimization Strategies: These methods focus on "intelligent input filtering" using the reasoning ability of LLM instead of changing the structure of the forecast model. A representative example in this category is the Augur model; This model uses a two-step approach to variable selection. In the first step, the time series data is converted to text and fed to a powerful LLM to discover the "causal relationships" between variables. The LLM identifies which variables actually influence the target variable (causal parents). In the second step (prediction), the input window of the model is filled with only these selected variables and irrelevant variables (which create noise) are removed. This "causal filtering" makes the model suffer less from cumulative error due to spurious correlations in multi-step predictions [92].

5.3 Retrieval-Augmented SLMs Combining Similar Historical Patterns

Extending the retrieval-augmented paradigms introduced in Section 4.4, this subsection examines concrete implementations and retrieval mechanisms specifically designed for time series data.

Leveraging similar historical patterns, beyond a simple database search mechanism, serves as a fundamental strategy for domain adaptation and context enrichment in SLMs. Traditional predictive models, and even standard language models, inherently possess "static knowledge," meaning that all information is encoded within the model weights. As a result, they often suffer from performance degradation when confronted with distribution shifts or the non-stationary dynamics commonly observed in time series data.

Modern retrieval-augmented approaches overcome this limitation by connecting the model to a "dynamic external memory," including knowledge bases of historical patterns. By retrieving and injecting similar patterns into the model context, without requiring parameter updates (i.e., in a zero-shot setting), these systems enhance the model's reasoning capability using real evidence and help bridge the gap between training and testing distributions.

5.3.1 Multi-Scale Pattern Matching Retrieval

In this approach, it is assumed that "history repeats itself," but this repetition may occur with phase shifts or on different time scales. Therefore, the system must be able to identify similar structural patterns. To overcome the limitations of language models in directly understanding numbers, **TimeRAG** framework proposes a two-step process: first, a knowledge base of historical subsequences is created. The main innovation of this model is the use of Dynamic Time Warping (DTW) for searching. Unlike Euclidean distance, which is sensitive to exact temporal alignment, DTW can accurately measure the similarity between two time series even if their rates of occurrence are different or they have phase shifts. After retrieving k similar subsequences (reference sequences), this data is transformed into a text prompt along with the original query. TimeRAG structures this information in such a way that the language model can infer the future path by comparing the current trend with successful historical examples. This method is especially effective in domains such as traffic or electricity consumption that have recurring patterns [43].

5.3.2 Adaptive and Learned Retrieval

In more complex scenarios, mere "appearance similarity" is not enough. A historical pattern may be similar to the current situation but have a different outcome due to different environmental conditions. These methods learn "which" information is useful and "how" it should be combined with the model's internal knowledge.

TS-RAG model: This model is designed to solve the problem of noise interference in RAG systems. If the retrieved data is irrelevant, it can degrade the model's performance. TS-RAG has two key components:

1. Pre-trained temporal encoder: Uses a time series model (such as TimesNet) to generate semantic embeddings of time windows so that the search is performed in vector space, not raw space.
2. Adaptive Retrieval Mixer (ARM): This innovative module dynamically weights the retrieved information. By simultaneously analyzing the "query embedding" and the "retrieved document embedding," ARM generates a Gating Score that determines how much the final output should be influenced by the "internal reasoning of the model" and how much by the "retrieved evidence." This mechanism allows the model to rely on its own knowledge when faced with noisy data [44].

FinSrag model: This model was developed in the domain of financial forecasting, where the data is highly random and non-stationary. The main challenge here is the "semantic gap" between mathematical similarity and behavioral market similarity. Instead of using generic encoders, the model uses a dedicated retriever called FinSeer, trained with a dual objective function:

1. Time series similarity
2. Market signals (such as price direction)

FinSeer: FinSeer refines its search strategy by receiving feedback from the language model (whether the retrieved documents led to the correct prediction). This allows the system to find patterns that may be different in appearance, but are similar in "market impact" (e.g., the pattern before the stock crash) [45].

5.3.3 In-Context Fine-Tuning

This approach takes the RAG concept beyond "memory aid" and uses it as a tool for on-the-fly training. The main idea of this research is that foundation models can adapt their behavior to the specific distribution of that data by observing a few relevant examples in the context window. For each target time series, the system first retrieves a set of relevant time series from the database. These examples include "history" and "ground truth." When these examples are given as prompts to the model, the model's self-attention mechanism adjusts its internal activations based on this evidence. This process is similar to the model performing a gradient descent step on the data of that specific domain, but without actually changing the weights. The results show that this method brings the zero-shot performance to the level of Fine-tuned models [46].

5.3.4 Hierarchical Agentic Frameworks

To handle complex tasks that require multi-step reasoning, single-model systems are not sufficient and agent-based architectures are needed.

Agentic RAG model: This model proposes a Multi-Agent System for time series analysis that consists of three levels:

1. Master Agent: This agent plays the role of a manager; it analyzes the user request (e.g., "prediction and anomaly analysis") and divides it into subtasks.
2. Specialized Sub-agents: Each sub-agent is trained and optimized for a specific task (such as short-term, long-term forecasting, or pattern recognition).
3. Shared Prompt Pool: Instead of searching raw data, sub-agents have access to a knowledge base of "optimized prompts". These prompts contain distilled knowledge about successful historical patterns and similar trends. Agents adapt their problem-solving strategy to the type of input data by retrieving the appropriate prompt [47].

5.4 Hybrid Pipelines: Synergy of Classical Models, Deep Learning, and SLM Embeddings in Time Series Forecasting

Building on the general neuro-symbolic principles outlined in Section 4.5, this section focuses on hybrid architectures tailored specifically for time series prediction. The development of hybrid pipelines represents a fundamental paradigm shift in time series forecasting. The primary objective of this approach is to bridge the gap between the abstract semantic reasoning capabilities of language models (LLMs/SLMs) and the numerical precision and temporal sensitivity of specialized models [102]. While language models excel in understanding macro patterns, contextual background, and generalization, classical statistical models (such as ARIMA) and specialized neural networks (such as LSTMs and lightweight Transformers) operate more efficiently in

modeling precise temporal dependencies and continuous fluctuations [49, 102]. This section reviews advanced strategies for integrating these models across four main categories.

5.4.1 Integration with Statistical Models and Decomposition Strategies

One of the most efficient hybrid methods involves decomposing time series into constituent components (such as trend and seasonality) and delegating the processing of each component to its specialized tool. This approach reduces data complexity and leverages the strengths of different models.

In the StreamTS framework, a streamlined forecasting strategy is employed wherein the time series is first decomposed into two main components: "trend" and "seasonal". In this architecture, the language model, due to its high capability in understanding long-term contexts and textual memory, assumes responsibility for forecasting the trend component [97]. In parallel, a lightweight linear model (similar to the approach in DLinear [103]) is used to forecast seasonal fluctuations and short-term variations that require high mathematical precision. A notable feature of this model is that in zero-shot forecasting scenarios, the linear module is replaced with the classical ARIMA model to enhance the model's generalizability when encountering new, unseen data [97].

The SMETimes model advances model integration by embedding the logic of statistical tools directly into the architecture via a "dynamic mixture-of-experts (dynamic MoE)" structure. Rather than relying on the ensemble of external model outputs, this system trains multiple internal expert layers to specialize in different temporal distincts (e.g., trend vs. seasonality). An intelligent adaptive gating mechanism functions as the integration core, performing optimal weighting between the semantic predictions (derived from the SLM) and the outputs of these statistical experts based on the input time series characteristics. This approach effectively internalizes the decomposition strategy, resulting in reduced forecasting error and increased inference efficiency by eliminating the need for multi-stage model execution [49].

In more complex applications such as global supply chain disruption forecasting, the Transformer-ARIMA hybrid model has been proposed to manage multimodal data. In this model, the ARIMA network is used to analyze structured time series data (such as shipment volumes) and extract linear trends, while the Transformer model (such as BERT) is tasked with processing unstructured textual data (such as news and text reports) and extracting contextual embeddings. The outputs of these two paths are integrated within a feature fusion layer to produce a final prediction with a comprehensive understanding of environmental conditions and numerical trends [48].

5.4.2 Dual-Modality Encoding and Cross-Modality Alignment

The second category of hybrid models focuses on resolving "modality imbalance" and "data entanglement" challenges by aligning the discrete representation space of language models with the continuous space of time series.

To activate the capability of LLMs through prototype alignment, the TEST model presents a data-centric approach to bridge the gap between time series and language models. This model first tokenizes the time series and converts it into vectors using an encoder. The core innovation of TEST lies in the use of instance-wise and feature-wise "contrastive learning" strategies to align the time series embedding space with the word embedding space of the language model. By utilizing aligned text prototypes as reference points, the model maps time series patterns to textual patterns (such as upward, downward, stable), enabling the language model to process them as a sequence of familiar patterns and thereby activating the LLM's capability as a "pattern machine" [104].

Building upon this prototype-alignment idea, the MLTA (Multi-Level Text Alignment) model enhances interpretability through a multi-level alignment framework. In this method, the time series is decomposed into three components: trend, seasonal, and residual, and each component is separately mapped to its specific text anchors. Specifically, the trend component is aligned with anchors describing movement direction (such as "increase" or "decrease"), while seasonal and residual components are associated with denser text prototypes. This design ensures that time series representations are not only more accurate but also more comprehensible and interpretable for the language model [105].

Beyond prototype-based alignment, the TimeCMA model introduces a dual-modality encoding architecture to explicitly address the data entanglement problem. This architecture comprises two branches: a time series encoder branch that extracts disentangled but weak embeddings, and an LLM-based branch that produces entangled but robust embeddings by converting time series into prompts. The key innovation of this model is the use of a "cross-modality alignment" mechanism, which extracts robust and relevant features from the prompt embedding space via channel-wise similarity retrieval and integrates them with time series embeddings [106].

Similarly addressing cross-modal imbalance, the BALM-TSF model specifically targets the issue of distributional and semantic imbalance between text and time series. This framework utilizes a lightweight two-branch approach: in one branch, descriptive statistics of the time series along with learnable prompts are fed into a language model to produce compact textual embeddings, while in the other branch, raw data is processed by a time series encoder. To align these two branches, BALM-TSF employs a scaling strategy and a contrastive objective to map textual embeddings precisely into the time series latent space, leading to significant performance improvements in long-term and few-shot forecasting [107].

Finally, the S^2IP-LLM model also builds upon time series decomposition but emphasizes alignment within the semantic space. This model first employs a specialized tokenization module that concatenates patches derived from time series decomposition (trend, seasonal, and residual). The resulting time series embeddings are then aligned with "semantic anchors" derived from the word embedding space of a pre-trained model [108]. These selected semantic anchors serve as textual prompts, providing the necessary context for the language model to better comprehend temporal dynamics.

5.4.3 Semantic Fusion and Knowledge Distillation

This category of methods focuses on transferring knowledge from large models to smaller ones or directly integrating high-level and low-level features.

The TimeKD model leverages "knowledge distillation" to address the slow inference issue in large models. In this method, a powerful teacher model, trained using "calibrated language models" and privileged information (such as ground truth prompts), transfers its knowledge to a lightweight, classical student model. This process involves correlation distillation and feature distillation, allowing the student model to mimic complex teacher behaviors and operate with high efficiency and faster inference speed [101].

The SemInf-TSF model is designed to combine high-level semantic representations from an LLM with low-level temporal information from a temporal Transformer. This model utilizes a gating mechanism to dynamically fuse features extracted from both models. The result of this fusion is a hybrid representation that encompasses both historical time series dynamics and semantic patterns of variations, leading to more accurate predictions [102].

5.4.4 Multi-Aspect and Federated Prompting Approaches

The Multi-Aspect Prompting (MAP4TS) model introduces a framework that directly integrates classical time series knowledge into prompt design. This model employs four specialized prompt types: global domain, local domain, statistical (based on classical analyses like STL), and temporal (including ACF and PACF analyses) [109]. These prompts are integrated with time series embeddings and processed through a cross-modality alignment module, enabling the model to perform joint reasoning on textual and numerical data.

Finally, the TIME-FFM model addresses privacy challenges and data silos by presenting the first Federated Foundation Model (FFM) for time series. By converting time series into text tokens and utilizing a "prompt adaption" module, this model dynamically generates domain-customized prompts. The personalized federated training strategy in this model allows for the learning of global encoders and local prediction heads [110]. Furthermore, models like AutoTimes, by redefining language models as autoregressive forecasters, enable arbitrary-length prediction and the use of time series as prompts (in-context forecasting), demonstrating the high flexibility of these hybrid approaches [111].

To conclude this section and provide a comprehensive comparative perspective on the SLMs and hybrid approaches discussed, their detailed specifications are consolidated in Table 7. This table organizes the models by "Category" and "Base Language Model" (Base LM), while explicitly indicating their capabilities in "zero/few-shot" scenarios. Furthermore, by highlighting the "Key Contribution" of each model and detailing the "Evaluation Context" (encompassing metrics, datasets, and application domains), it serves as an effective tool for researchers to select the most appropriate architecture for their specific computational and application requirements through a deeper understanding of the functional distinctions among existing models.

Table 7: Comprehensive comparison of SLM-based time series forecasting models.

Model	Category	Base LM	Zero/ Shot	Key Contribution	Evaluation Context (Metrics / Data / Domain)
ChatTime	Prompt Eng.	LLaMA-2-7B	✓	Modeling TS as a foreign language; Unified multimodal framework for forecasting & QA.	**Met:** MAE, Rank **Data:** Monash, TFB, ETT, Weather **Dom:** Energy, Finance, Climate
Chronos	Prompt Eng.	T5, GPT-2	✓	Minimalist framework converting forecasting into classification over fixed tokens.	**Met:** WQL, MASE **Data:** 42 datasets (Monash, M4) **Dom:** Energy, Retail, Health
MultiCast	Prompt Eng.	LLaMA-2, Phi-2	✓	Enabling zero-shot multivariate forecasting; Reducing token usage via SAX.	**Met:** RMSE **Data:** Gas Rate, ETDataset **Dom:** Energy, Weather, Industrial
PromptCast	Prompt Eng.	T5, BERT, GPT-2	✓	Pioneering prompt-based paradigm; Framing forecasting as text generation (PISA dataset).	**Met:** RMSE, MAE **Data:** PISA (Temp, Energy) **Dom:** Weather, Mobility
TOKON	Prompt Eng.	GPT-4o-mini	✓	Mapping numbers to single tokens; Reasoning-based instructions (TFSC).	**Met:** RMSE, MAE **Data:** AIHEPC, M4 Subset **Dom:** Energy, Economics
Informed Forecast	Prompt Eng.	GPT-4o-mini	✓	Integrating auxiliary knowledge (calendar/time) into prompts without retraining.	**Met:** MAE, MAPE **Data:** WHO FluNet, Call Center **Dom:** Health, Service
StreamTS	Hybrid	DeepSeek-7B	✓	Decoupling trend (SLM) and seasonal (Linear/ARIMA) forecasting.	**Met:** MSE, MAE **Data:** ETT, Weather, Traffic **Dom:** Energy, Traffic
SMETimes	Hybrid	LLaMA-3B, GPT-2	✓	Dynamic MoE structures encoding statistical dynamics (e.g., ARIMA-like behaviors).	**Met:** MSE, MAE **Data:** ETT, Weather, Solar **Dom:** Energy, Weather
Trans.-ARIMA	Hybrid	BERT	×	Integrating structured TS and unstructured text (news) for supply chains.	**Met:** Acc, F1 **Data:** Supply Chain Disruption **Dom:** Supply Chain
TimeCMA	Hybrid	GPT-2	×	Extracting robust TS embeddings via alignment with textual prompt embeddings.	**Met:** MSE, MAE **Data:** ETT, Exchange, FRED-MD **Dom:** Economics, Health
BALM-TSF	Hybrid	GPT-2	✓	Mapping textual embeddings to TS latent space to resolve modality imbalance.	**Met:** MSE, MAE **Data:** ETT, Exchange, Weather **Dom:** Energy, Economics
S^2IP-LLM	Hybrid	GPT-2	✓	Utilizing semantic anchors from word embeddings to enhance TS embeddings.	**Met:** MSE, MAE **Data:** ETT, Weather, Traffic **Dom:** Energy, Traffic
TimeKD	Hybrid	GPT-2, LLaMA	✓	Privileged Knowledge Distillation from LLM teacher to lightweight student.	**Met:** MSE, MAE **Data:** ETT, Exchange, PEMS **Dom:** Traffic, Energy
MAP4TS	Hybrid	GPT-2, LLaMA	✓	Integrating classical TS knowledge (STL, ACF) directly into prompt design.	**Met:** MSE, MAE **Data:** ETT, Climate, Health **Dom:** Energy, Climate, Health
TIME-FFM	Hybrid	GPT-2	✓	First FFM; privacy-preserving prompt generation.	**Met:** MSE, MAE **Data:** ETT, Weather, ILI **Dom:** Health, Energy
AutoTimes	Hybrid	LLaMA-7B, GPT-2	✓	Repurposing SLMs as autoregressive forecasters; using TS as prompts.	**Met:** MSE, MAE **Data:** ETT, Traffic, M4 **Dom:** Energy, Traffic, Finance
TEST	Hybrid	BERT, GPT-2	✓	Aligning tokenized TS with text prototypes (e.g., up/down).	**Met:** Acc, MSE **Data:** UCR, UEA, ETT **Dom:** Classification, Forecasting, Energy
GPT4TS	Fine-tuning	GPT-2	✓	Freezing LM body and training only one linear layer (One Fits All).	**Met:** MSE, MAE **Data:** ETT, Traffic, ILI **Dom:** Energy, Traffic, Healthcare
MOMENT	Fine-tuning	T5	✓	Open-source foundational models with Masked pretraining.	**Met:** MSE, MAE **Data:** The Time Series Pile **Dom:** Energy, Traffic, Finance, Healthcare
UniTime	Fine-tuning	GPT-2	✓	Using "domain instructions" and variable-step patching for cross-domain learning.	**Met:** MSE, MAE **Data:** ETT, Exchange, Illness **Dom:** Energy, Finance, Healthcare
Time-Prompt	Fine-tuning	GPT-2, LLaMA	✓	Combining soft (learnable) and hard (statistical) prompts with LoRA.	**Met:** MSE, MAE **Data:** ETT, Carbon Emissions **Dom:** Environment
SE-LLM	Fine-tuning	LLaMA, GPT-2	✓	Time-Adapter module (LSTM) within attention layers for dependencies.	**Met:** MSE, MAE **Data:** ETT, M4, Traffic **Dom:** Energy, Finance, Traffic
TEMPO	Fine-tuning	GPT-2	✓	Decomposing TS into trend/seasonal; using prompt pool for shifts.	**Met:** MSE, MAE **Data:** ETT, Weather, Traffic **Dom:** Energy
TsLLM	Fine-tuning	LLaMA, Qwen	✓	Patch-based encoding (VAE) and mixed text-number generation.	**Met:** MSE, MAE **Data:** GIFT-Eval, ECG-QA **Dom:** Healthcare
Augur	Fine-tuning	Generic LLMs	✓	Discovering causal relationships and filtering irrelevant variables.	**Met:** MSE, MAE **Data:** ETT, Weather, Exchange **Dom:** Energy, Weather, Finance
TimesFM	Fine-tuning	Decoder-only Transformer	✓	Decoder-only Transformer architecture with asymmetric patching (output patches longer than input).	**Met:** MAE, MSE **Data:** Google Trends, Wiki Pageviews, Synthetic, M4, Electricity **Dom:** Web Traffic, Retail, Energy
TimeRAG	RAG	General LLMs	✓	Retrieval of reference sequences using DTW to enhance prompts.	**Met:** MSE, MAE **Data:** PeMSD, ETT **Dom:** Traffic, Energy
TS-RAG	RAG	TSFMs	✓	Retrieving semantically relevant segments; fusion via adaptive retrieval mixer.	**Met:** MSE, MAE **Data:** ETT, Weather, Traffic, Exchange **Dom:** Energy, Traffic, Weather, Finance
FinSrag	RAG	StockLLM-8B	✓	Domain-specific retriever (FinSeer) refined by LLM feedback.	**Met:** ACC, MCC **Data:** ACL18, STOCK23 **Dom:** Finance
Agentic RAG	RAG	Llama-8B	✓	Hierarchical multi-agent architecture; master agent orchestrates prompt retrieval.	**Met:** Acc, Prec **Data:** PeMSD, METR-LA **Dom:** Traffic

6 Comparative Evaluation

Evaluating the performance of SLMs in time series forecasting necessitates a transition from a unidimensional approach to a comprehensive and multifaceted evaluation framework [32]. Unlike classical statistical models, which yield solely numerical outputs, modern LLM/SLM-based models confront complex challenges, including valid token generation, probabilistic uncertainty management, and semantic interpretability [34]. Consequently, the selection of appropriate metrics must be predicated on the data modality (continuous, discrete, or textual) and the ultimate forecasting objective (point accuracy, distributional coverage, or event detection) [65]. A critical analysis of these metrics follows, and Table 8 provides a comprehensive comparative overview of their distinct characteristics, optimal use cases, and limitations.

6.1 Metric-Based Evaluation Framework

6.1.1 Scale-Dependent Metrics

This category of metrics measures the deviation between predicted values ($\hat{y}_t$) and actual values (y_t) in the original units of the data and is vital for evaluating model performance on a specific dataset [112].

Among these, the Mean Squared Error (MSE) and its square root, the Root Mean Squared Error (RMSE), are defined, assuming T represents the number of samples, as follows:

$$MSE = \frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2 \tag{1}$$

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2} \tag{2}$$

The RMSE metric, defined as the square root of the Mean Squared Error, measures the average magnitude of prediction errors in the original units of the data and is preferred in applications where penalizing large errors is essential, as its squaring mechanism assigns disproportionately higher weight to larger deviations [112]. However, analyses indicate that this same squaring mechanism renders RMSE highly sensitive to outliers, and it can therefore provide misleading evaluations in time series containing abnormal noise [65]. Furthermore, this metric is entirely scale-dependent and is unsuitable for comparing models across multiple datasets with differing units [65].

In contrast, the Mean Absolute Error (MAE) calculates the mean absolute deviations and provides a linear and robust interpretation of model error:

$$MAE = \frac{1}{T}\sum_{t=1}^{T}|y_t - \hat{y}_t| \tag{3}$$

Compared to RMSE, MAE is more robust against outliers and penalizes severe fluctuations less, a feature that makes it desirable for financial markets [112]. On the other hand, due to insufficient sensitivity to extreme errors, MAE might not convey sufficient information in risk management applications where rare events are significant [65].

6.1.2 Scale-Independent and Relative Metrics

With the emergence of foundation models trained on dozens of diverse datasets, metrics are required that assess model performance regardless of the data measurement unit [34].

The Mean Absolute Percentage Error (MAPE) is popular in management reporting due to its percentage interpretability; however, it is undefined when facing zero values ($y_t = 0$) and applies an asymmetric penalty [65]. To mitigate this issue, the Symmetric MAPE (SMAPE) metric is proposed, utilizing the average of the absolute predicted and actual values in the denominator:

$$MAPE = \frac{100}{T} \sum_{t=1}^{T} \left| \frac{y_t - \hat{y}_t}{y_t} \right| \tag{4}$$

$$SMAPE = \frac{100}{T} \sum_{t=1}^{T} \frac{|y_t - \hat{y}_t|}{(|y_t| + |\hat{y}_t|)/2} \tag{5}$$

Although SMAPE is bounded within the 0 to 200% range, Hewamalage et al. [65] note that this metric remains not perfectly symmetric and is undefined if both actual and predicted values are zero. Furthermore, interpreting SMAPE is more difficult for non-expert stakeholders compared to MAPE [65].

To overcome the limitations of percentage-based metrics, the Mean Absolute Scaled Error (MASE) is proposed, which scales the forecast error relative to the in-sample error of a baseline method (such as the seasonal naïve forecast) computed on the training data [34, 65]. Formally, let N denote the number of training observations, T the number of test observations, and m the seasonal period of the time series (e.g., $m = 12$ for monthly data or $m = 7$ for daily data with weekly seasonality). MASE is then defined as:

$$MASE = \frac{\frac{1}{T} \sum_{t=1}^{T} |y_t - \hat{y}_t|}{\frac{1}{N-m} \sum_{i=m+1}^{N} |y_i - y_{i-m}|} \tag{6}$$

where the numerator is the MAE of the forecast model on the test set, and the denominator is the MAE of the seasonal naïve method on the training set–computed as the mean absolute difference between each observation and the observation m periods prior. Ansari et al. [34] demonstrated that MASE is an ideal metric for assessing zero-shot performance on heterogeneous datasets, as it is scale-invariant and avoids division-by-zero problems encountered by percentage-based metrics. However, its calculation requires a precise and consistent definition of the seasonal period (m) for the baseline, which may be ambiguous for datasets with multiple or irregular seasonal cycles [65].

6.1.3 Probabilistic Metrics

Since many SLMs, such as Chronos, model the joint distribution of future values, deterministic metrics are insufficient for their comprehensive evaluation [34]. The Weighted Quantile Loss (WQL) addresses this gap by assessing the compatibility between the predicted distribution and the ground truth at various quantile levels (τ):

$$WQL = \frac{\sum_{t,\tau} \max(\tau(y_t - \hat{y}_t^{(\tau)}), (1-\tau)(\hat{y}_t^{(\tau)} - y_t))}{\sum_t |y_t|} \tag{7}$$

This metric is vital for evaluating distribution heads in language models, as it assesses the model's ability to estimate confidence intervals and manage tail risks [34]. Qiu et al. [64] demonstrated that using WQL provides a more complete picture of model performance under uncertainty.

6.1.4 Classification Metrics

In applications where the forecasting objective involves detecting discrete events or market direction, the regression problem transforms into classification [48]. Wang [48] showed that using accuracy alone is misleading in imbalanced data and suggested using the F1-score, defined as the harmonic mean of precision and recall. For financial markets where detecting movement direction (up/down) is the goal, Xiao et al. [45] propose the Matthews Correlation Coefficient (MCC) metric, which considers all cells of the confusion matrix and provides a reliable score even in imbalanced data:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{8}$$

6.1.5 Generative and Textual Metrics

With the advent of multimodal models, evaluating the quality of textual and structural model outputs is essential [91]. The Missing Rate metric measures the percentage of responses where the language model failed to convert to a valid numerical format, indicating the model's inability regarding instruction following [32]:

$$Missing\ Rate = \frac{N_{failed}}{N_{total}} \times 100\% \tag{9}$$

To evaluate generated textual explanations, Kim et al. [91] employed cosine similarity, which captures semantic similarity in embedding space, ROUGE, which measures word overlap, and GPT Semantic Score metrics. In the GPT Score metric, a powerful language model (like GPT-4) acts as a judge to score the semantic consistency of the generated text against reality on a Likert scale (e.g., 1 to 10) via a specific prompt [91].

6.1.6 Ranking

In large-scale benchmarks, direct error averaging may be skewed by more difficult datasets; thus, the mean Rank of models is used [64]. Since Rank alone does not indicate the magnitude of performance difference, Hewamalage et al. [65] suggest using statistical tests such as the Friedman Test and Critical Difference Diagrams to determine the statistical significance of these ranks and confirm the true superiority of models.

Table 8 Comprehensive comparative evaluation metrics for SLM-based time series forecasting

Category	Metric	Primary Focus	Best Use Case	Key Limitation / Challenge
Scale-Dependent	RMSE	Penalizing large deviations (variance penalty)	General forecasting where error distribution is expected to be Gaussian [112].	Highly sensitive to outliers; unreliability for ranking across series [65].
	MAE	Linear magnitude assessment	Robust regression against noise and outliers (e.g., financial markets) [112].	Ignores the relative magnitude of extreme errors; may fail to capture tail risks [65].
Scale-Independent	SMAPE	Symmetric percentage error bounded [0-200%]	General competitions with diverse scales [65].	Asymmetric penalty for under/over-estimation; undefined if $y = \hat{y} = 0$ [65].
	MASE	Scaling error relative to a naïve baseline	Zero-shot evaluation and generalization across heterogeneous domains [34].	Requires a precise definition of seasonality for the benchmark; abstract interpretation [65].
Probabilistic	WQL	Assessing predicted distribution quality (quantile loss)	Generative models; quantifying uncertainty and confidence intervals [34].	Computationally intensive; requires multiple sampling or ground truth quantiles [64].
Structural Validity	Missing Rate	Instruction following capability	Prompt-based models; detecting hallucination or format failure [32].	Does not measure the numerical accuracy of the "valid" outputs [32].
Classification	MCC	Prediction quality correlation in imbalanced classes	Predicting market direction (up/down) in imbalanced financial data [45].	Indirect interpretation for financial profit/loss calculation [45].
	F1-score	Harmonic mean of precision and recall	Detecting discrete events (e.g., supply chain disruptions or anomalies) [48].	Loses information regarding the magnitude of the event severity [48].
Textual Quality	GPT Score	Semantic evaluation by a judge LLM	Multimodal forecasting; assessing reasoning correctness and semantic alignment [91].	High cost; dependence on the judge model's inherent biases [91].
	Cosine sim.	Vector similarity in embedding space	Assessing overall semantic proximity between generated and reference text [91].	May fail to distinguish subtle factual differences critical in forecasting [91].
Aggregate	Rank	Average performance rank across datasets	Large-scale benchmarks to mitigate the effect of outliers across datasets [64].	Ignores the performance margin between models; requires statistical significance tests [65].

6.2 Robustness Tests: Evaluation Protocols and Adaptability

The evaluation of SLMs in time series prediction should not be limited solely to accuracy on static test data. Given the dynamic nature of real-world environments,

assessing model robustness against input noise and distributional shifts (concept drift) is a critical criterion for gauging model reliability in final deployment [113]. In what follows, relevant evaluation protocols and adaptive strategies are reviewed.

6.2.1 Noise Tolerance Evaluation

In the context of SLMs, noise tolerance testing encompasses not only numerical noise (such as Gaussian noise) but also structural and linguistic noise [66, 67]. Recent studies suggest that models should be tested against varying levels of perturbations. Scaria et al. [67] demonstrated that evaluation must include the injection of character-level noise (character-flip) and transliteration noise, as these factors have a detrimental effect on the tokenization process and self-attention mechanisms in SLMs. The results of these tests indicate that models pre-trained on high-quality, synthetic data (such as Phi-2) maintain higher reconstruction rates and accuracy compared to models that are merely larger in size [67]. For quantitative assessment, it is recommended to compare model performance under "clean data" versus "noisy data" conditions. Furthermore, the employment of data augmentation strategies involving the injection of data-dependent Gaussian noise into historical samples has been introduced as a valid benchmark for mitigating the distribution gap between training and testing data [66].

6.2.2 Concept Drift Adaptation Tests

Concept drift refers to a change in the joint probability distribution of data $P_t(X, Y) \neq P_{t+1}(X, Y)$, which leads to the obsolescence of knowledge in static language models. Evaluating the capability of SLMs to manage this phenomenon requires specific tests. One evaluation protocol focuses on detection-based assessment, in which the model's ability to identify the exact onset of a change point is evaluated. Pawar et al. [114] propose the use of the PELT (Pruned Exact Linear Time) algorithm to identify change points within input features, with the success criterion being the extent of accuracy improvement following selective retraining on data after the change point. Results indicate that this approach can reduce the MAE by up to 44% in forecasting models [114]. Additionally, Zhang et al. [66] introduced the D^3A framework, where the evaluation metric is the detection of a statistically significant deviation in the model's loss distribution to prevent performance degradation over time.

For global models where full retraining might be computationally expensive, evaluation shifts to the model's capability for continuous adaptation through dynamic weighting. Liu et al. [115] proposed "Error Contribution Weighting (ECW)" and "Gradient Descent Weighting (GDW)" methods, with the evaluation criterion being the comparison of an ensemble model's performance against baseline models during sudden and gradual drift events. Empirical results on simulated data demonstrate that the GDW method offers superior stability and accuracy compared to traditional statistical methods and baseline models in sudden drift scenarios [115].

To provide a clear synthesis of these methodologies, Table 9 categorizes the primary strategies for concept drift adaptation relevant to SLMs.

Table 9 Comparative overview of concept drift adaptation strategies for SLMs in time series forecasting.

Strategy Category	Method	Core Mechanism	Adaptation Action	Performance Benchmark
Detection-based	PELT [114]	Statistical change point detection in input features	Selective Retraining: Retrain model only on post-drift data segments	Reduced MAE by ∼44% in real-world datasets
	D^3A [66]	Monitoring deviations in loss distribution	Aggressive Adaptation: Updates model using history-augmented data upon alarm	Outperformed SOTA (FSNet) by ∼33% in MSE
Continuous	ECW/GDW [115]	Dynamic Weighting: Gradient descent or error-based weight adjustment	Ensemble Fusion: Weighted average of global (stable) and local (fast) models	Superior stability in sudden and incremental drifts

6.3 Scalability Evaluation: Metrics and Framework

Scalability in SLMs designed for time series forecasting is a multi-dimensional concept that extends beyond parameter count to encompass inference latency, memory footprint, energy consumption, and throughput across diverse hardware platforms [72, 116]. Unlike classical models that produce straightforward numerical outputs, evaluating the deployment viability of language-model-based systems requires a distinct set of efficiency-oriented measures [72]. This subsection introduces six key metrics that together form a systematic framework for scalability assessment and analyzes the considerations associated with each. The architectural and system-level challenges that motivate the monitoring of these metrics are examined comprehensively in Section 8.4

6.3.1 Key Scalability Metrics

Inference latency measures the time required to produce a single prediction and directly affects the feasibility of deployment in real-time applications [23]. In language models, the inference process proceeds in two phasesa parallel prefill stage and an autoregressive decode stage—with first-token latency in the prefill phase being especially critical for time-sensitive tasks. Factors such as the number of layers, feed-forward network width, vocabulary size, and parameter sharing all influence latency independently of model size [72]. In practice, models that eschew quadratic self-attention in favor of lightweight architectures such as the MLP-Mixer can achieve competitive inference speeds even on CPU-only machines [117], indicating that architectural choices matter as much as parameter reduction in reducing latency.

Another critical dimension is memory footprint, which denotes the peak GPU or CPU memory consumed during inference and serves as a proxy for model compactness and the effectiveness of compression techniques [23, 72]. Memory footprint is not

strictly a linear function of parameter count; factors such as vocabulary size, context length, and KV cache configuration also exert substantial influence. In SLMs, runtime memory is primarily consumed by model parameters, the KV cache, and intermediate computation buffers, with the share of the cache and buffers growing significantly as context length increases [72]. Consequently, memory footprint depends not only on model size but also on how the context window is managed and how the model is architected internally.

Parameter count—the total number of trainable parameters—while not directly determining inference cost, correlates with model size, storage requirements, and memory bandwidth demands [72]. Time series forecasting models span a broad parametric spectrum, from Transformer-based architectures with several hundred million parameters to highly compact designs with only a few million [49, 117]. This diversity testifies to advances in efficient architecture design and demonstrates that limited model capacity need not preclude learning complex temporal patterns. However, parameter count must be interpreted alongside other efficiency metrics, as models with more parameters are not necessarily slower or more resource-intensive [72].

A complementary measure, Floating-Point Operations (FLOPs), quantifies hardware-independent computational complexity by counting floating-point operations per inference, providing an estimate of a model's computational demands [72, 116]. FLOPs are measured alongside financial cost, CO_2 emissions, and energy usage as part of the efficiency metric suite, and their value depends on the specific hardware configuration [116]. In language models, FLOP reduction is primarily achieved by replacing quadratic self-attention with lighter-weight alternatives [117]. Nevertheless, FLOPs alone do not paint a complete picture of efficiency, as a model with low FLOPs may still exhibit high latency due to memory bandwidth bottlenecks [72].

Energy per inference, together with financial cost and CO_2 emissions, is classified under sustainability and resource-consumption metrics [116]. The energy efficiency ratio characterizes how energy-intensive a model is in practice [23]. Although direct energy measurement remains uncommon in the time series forecasting literature, low-footprint designs—such as models that run exclusively on CPU or those with minimal active parameters—implicitly contribute to reduced energy usage [107, 117]. This dimension of scalability warrants greater attention in future research, particularly for battery-powered deployments and energy-constrained environments.

Finally, throughput—the number of samples processed per unit time—is critical for latency-sensitive applications and large-scale tasks [23]. This metric complements inference latency: while latency measures response time for a single request, throughput captures the overall system capacity for continuous data processing. In Transformer-based models, throughput is heavily influenced by the quadratic complexity of self-attention, and replacing it with more efficient architectures can substantially improve throughput [49, 117].

6.3.2 Key Considerations in Scalability Evaluation

Analysis of the above metrics reveals several guiding principles for scalability assessment in SLMs. First, the accuracy-efficiency trade-off is not absolute: intelligent architectural design can improve efficiency without significantly sacrificing accuracy

[49, 107, 117]. Second, hardware dependency is pervasive: latency, memory, and even FLOPs are heavily influenced by hardware configuration, making controlled benchmarking conditions essential for fair comparisons [72, 116]. Third, unidimensional evaluation is insufficient: judging a model's suitability for resource-constrained deployment requires examining the full efficiency profile—accuracy, speed, memory, energy, and throughput—rather than any single metric in isolation. The architectural and system-level challenges that necessitate such comprehensive evaluation, along with the optimization techniques designed to address them, are discussed in detail in Section 8.4.

6.4 Interpretability: Attention-based Explanations and Feature Attribution

In the transition from traditional statistical models to complex deep learning models, particularly SLMs and Transformers, the "black-box" nature of these models has become a significant barrier to establishing trust among users and decision-makers. Interpretability in time series forecasting is defined as the extraction of relevant knowledge from the model regarding relationships present in the data or patterns learned by the model [118]. Unlike NLP tasks where explanations can be textual, in time series, interpretability often rests on two primary pillars: **Attention-based explanations**, which focus on the importance of temporal steps, and **feature attribution**, which determines the contribution of each input variable to the final prediction [119].

6.4.1 Attention-based Explanations

Attention mechanisms, which form the core of Transformer architectures, offer a powerful tool for understanding temporal dependencies. The central idea is that attention weights can be interpreted as a proxy for the relative importance of past time steps in predicting the future. Research has demonstrated that analyzing attention weights in model layers can reveal recurring and seasonal patterns; for instance, in advanced models such as the Temporal Fusion Transformer (TFT), attention matrices in the self-attention layer are capable of modeling long-term dependencies, and by examining the distribution of attention weights across different forecast horizons, one can observe that the model automatically focuses on specific time intervals (such as similar days in preceding weeks), indicating the learning of seasonality without the need for manual feature engineering [119]. However, the raw utilization of attention weights does not always provide an accurate interpretation, as they may reflect positional biases rather than true temporal importance. Newer models like CrossScaleNet have attempted to enhance intrinsic interpretability by introducing multi-scale cross-attention mechanisms, in which temporal patches are compared with high-level representations instead of raw input signals, causing the model to focus on semantic patterns and main trends rather than random noise and thereby identifying "temporal saliency" with higher accuracy compared to post-hoc methods [120]. In addition to persistent patterns, attention mechanisms can be utilized to identify sudden changes in temporal dynamics or "regime shifts" by calculating the distance between the attention pattern at a specific time and the average attention pattern, which reveals moments where the

model fundamentally alters its behavior (e.g., during periods of high financial market volatility) [119].

6.4.2 Feature Attribution and Variable Selection

While attention mechanisms focus on "time," feature attribution methods answer the question of "which variable" (e.g., temperature, power consumption, or exchange rates) has had the greatest impact on the output. Some architectures are designed to perform feature selection intrinsically—an approach referred to as intrinsic methods—through the use of variable selection networks that allow the model to dynamically assign different weights to inputs at each time step, an approach that not only eliminates noisy inputs and improves performance but also provides precise insight into the global importance of each variable across the entire dataset or its local importance in a specific sample by examining the selector weights [119]. For models lacking intrinsic interpretability capabilities, post-hoc methods such as SHAP or integrated gradients are employed to estimate the contribution of each feature by perturbing inputs or analyzing gradients. However, recent studies indicate that these methods may face challenges such as high computational costs and instability when dealing with LLMs [118], and in multivariate time series, simple feature ablation methods might be insufficient for complex datasets and fail to accurately depict interactions between variables [118, 120].

6.4.3 Latent Space Visualization and Decision Boundaries

Beyond examining weights and gradients, novel interpretability methods have moved towards visualizing the model's embedding space. Tools like DeepView, utilizing discriminative dimensionality reduction methods, enable the 2D visualization of the decision space of language models. These methods allow researchers to observe how the model organizes data, the boundaries between classes, and regions of high uncertainty. Such visualizations are particularly vital for identifying adversarial examples and outliers that might disrupt the predictive model's performance [121].

Ultimately, the convergence of these methods suggests that for complete interpretability in language-based time series models, a combination of intrinsic interpretability (such as attention and variable selectors) and global visualization techniques is required to accurately clarify both temporal dynamics and feature importance for human users [119, 121].

7 Benchmark Datasets and Frameworks

7.1 Popular Time Series Corpora

Unlike the field of NLP, which relies on structured vocabulary and grammar, or Computer Vision (CV), which relies on standardized datasets, time series analysis faces the challenge of lacking a single, universal standard. Real-world time series data—from chaotic and trend-driven fluctuations in financial markets to seasonal and intermittent patterns in energy consumption—exhibit vastly different dynamical and environmental characteristics, making the development of foundation models more challenging

than NLP. With the emergence of language models, recent research has adopted a hybrid approach to overcome these challenges and ensure fair comparisons with deep learning methods. This approach involves using de facto standards in energy, traffic, and climate to measure the accuracy of spatiotemporal dependencies, as well as leveraging large-scale archives such as the **Monash Archive**(It points out that there is no comprehensive archive of time series datasets and therefore offers a complete time series repository. It offers 30 datasets for real-world data and competitive purposes, covering various domains such as energy, economics, nature, and sales.) and **M-Competitions** to assess model robustness to distribution shifts. Using these open datasets, often collected in real-time via sensors or transaction logs, allows researchers to objectively and convincingly demonstrate the contribution of their proposed models to benchmarks without engaging in complex data collection processes. In this section, we examine these corpora frameworks in detail [64]. To comprehensively evaluate language models in time series forecasting, recent research has utilized a collection of standard datasets, each challenging a specific aspect of model capabilities. Based on the systematic review conducted (Table 7), the most widely used of these corpora are:

7.1.1 Long-Term Multivariate Forecasting (LTSF) Benchmarks

This category comprises the **Electricity**, **Traffic**, and **Weather** datasets. Characterized by high dimensionality and intertwined inter-variable dependencies, these datasets serve as the primary touchstone for evaluating the long-term memory and modeling capacity of Transformer-based SLMs.

Electricity: This dataset contains hourly electricity consumption records of clients. The primary technical challenge lies in its composite patterns, requiring models to simultaneously capture short-term fluctuations (daily/nightly usage) and long-term seasonal trends (summer/winter). SLMs such as **UniTime** and **TEMPO** leverage this dataset to demonstrate their ability to preserve periodic patterns over extended forecast horizons [89, 94].

Traffic: this dataset records road occupancy rates via hundreds of sensors. Unlike the Electricity dataset where variables may be independent, traffic data exhibits strong spatial correlations, where congestion at one sensor influences adjacent ones. Hybrid and agentic models, such as **StreamTS** and **TimeKD**, utilize this data to test the model's capacity to comprehend complex spatiotemporal dynamics alongside temporal sequences [97, 101].

Weather: Comprising 21 meteorological indicators (e.g., temperature, humidity, pressure) recorded every 10 minutes, this dataset is characterized by rapid fluctuations and strong physical dependencies between variables. Models like **BALM-TSF** and **TimeCMA** use this benchmark to validate whether the alignment of textual space with time series embeddings successfully captures the underlying physical laws governing the data [106, 107].

7.1.2 Financial and Volatile Data

Evaluating robustness is critical for SLMs, particularly in domains where data does not follow fixed physical rules.

Exchange Rates: Unlike physical datasets, financial data lacks distinct seasonality and is subject to severe non-stationarity and market shocks (distribution shifts). Foundation models like **MOMENT** utilize this dataset to demonstrate that pretraining on massive corpora enables the model to remain robust against noise and sudden regime changes, where classical models often fail [95].

7.1.3 Large-Scale Archives for Generalization

The utilization of massive archives distinguishes modern foundation models from traditional deep learning approaches, shifting the focus from fine-tuning to zero-shot capabilities.

M4 & M5 Competitions: These collections contain thousands of diverse time series ranging from macroeconomics and finance to demographics and retail, with varying frequencies (yearly to hourly). In recent research involving models like **Chronos** and **TSLLM**, these archives are not merely used for training but are crucial for zero-shot evaluation. The objective is to assess whether the SLM has learned the universal language of time series and can generalize to unseen domains without any specific parameter updates [34].

7.2 Open-Source Toolkits

A detailed examination of the implementations of the models reviewed in this paper shows that developers are no longer relying solely on traditional time series libraries; instead they are leveraging a hybrid architecture. This new ecosystem includes specialized probabilistic modeling frameworks such as **GluonTS** for the Chronos model [34, 95], vast repositories of pretrained models such as **Hugging Face** for access to LLaMA and GPT weights [35, 93], and **classical tools** for signal decomposition in hybrid architectures [38, 49, 97]. In the following, these tools are categorized according to their key role in the development of SLMs.

7.2.1 Probabilistic Modeling Frameworks (GluonTS)

language models are inherently probabilistic. In time series, this property is crucial for estimating uncertainty. Originally built for deep learning models (such as DeepAR), the **GluonTS framework** has now become the main infrastructure for basic time series models. The **Chronos model** uses the GluonTS ecosystem for input/output management and evaluation. Since Chronos transforms the time series into discrete tokens, its raw output is a probability distribution over the vocabulary. GluonTS takes this distribution and transforms it into confidence intervals (Prediction Intervals) and standard probabilistic metrics such as **WQL**. Without GluonTS, it would be very difficult to convert the linguistic outputs of the model into statistically reliable measures [34].

7.2.2 Classical and Hybrid Tools (Prophet & Darts)

Despite the power of SLMs in reasoning, they sometimes perform worse than classical models in understanding precise mathematical patterns or simple linear trends. While

some hybrid architectures rely on external libraries like Prophet or Darts, the **SME-Times model** adopts a more efficient approach by internalizing statistical capabilities. Instead of executing separate algorithms, it employs a dynamic mixture-of-experts framework. In this structure, specific parameterized projection layers act as "internal experts," trained to mimic distinct temporal dynamics such as linear trends (resembling ARIMA) or seasonal periodicities (similar to Prophet). This design allows the model to leverage statistical inductive biases directly within the neural network, prioritizing between statistical regularity and semantic reasoning via a gating mechanism without the computational overhead of external tools [49]. For zero-shot scenarios and new domains, the StreamTS model uses simple linear or ARIMA models to avoid the problem of overfitting language models [97].

7.2.3 Language Model Substrates and Wrappers (Hugging Face)

Most of the models reviewed in this article use pre-trained weights from existing language models (such as GPT-2, LLaMA, or BERT). The **Hugging Face Transformers library** is the main repository of these weights, but these models were originally designed to process discrete textual tokens rather than continuous numerical data. Therefore, researchers create "shells" or wrappers that wrap around the main model. The wrapper mechanism is responsible for taking the time series numbers, segmenting them into fixed-length patches, and then projecting them into dense vector representations compatible with the language model's embedding space. In addition to the main model, tools such as PEFT (Parameter-Efficient Fine-Tuning) are used to implement methods such as LoRA. Models such as **Time-Prompt** or **TEMPO** use these tools to train only a small number of new parameters without changing the heavy weights of the main model (frozen backbone) [93, 94].

7.3 Standard Evaluation Protocols and Train/Test Splits

To ensure a fair, scientific, and reproducible comparison between SLMs and advanced deep learning models, it is crucial to adopt unified evaluation protocols. Unlike the common qualitative metrics in NLP, evaluation in time series requires adherence to quantitatively precise settings regarding "forecasting horizons" and "context windows" [65]. Our systematic review shows that leading studies in this field follow established standards in LTSF rather than inventing new metrics. This alignment in protocols allows for the measurement of model stability over short and long time frames, while avoiding challenges such as data leakage and overfitting.

7.3.1 Forecasting Horizons (H) and Long-Term Stability

For LTSF, the established industry standard requires evaluating models across four exponentially increasing prediction horizons: $H \in \{96, 192, 336, 720\}$ time steps [96]. This multi-horizon approach is crucial for assessing the model's stability; a robust SLM must maintain performance not only in the immediate future ($H = 96$) but also as temporal dependencies weaken over longer horizons ($H = 720$). Some studies on high-frequency data (like Traffic) also include shorter horizons (e.g., $H \in \{24, 48\}$) to test rapid responsiveness [90].

7.3.2 Input Context Strategies: Fixed vs. Tuned Windows

A key divergence from classical deep learning lies in the input context length (L). While traditional models often treat L as a dataset-specific hyperparameter, SLM-based approaches frequently adopt a fixed-window strategy. For instance, the **MOMENT** family imposes a strict input constraint of $L = 512$ time points [95]. This design choice leverages the SLM's pre-trained ability to process long sequences and capture global dependencies without requiring task-specific architectural modifications. Other models, such as **UniTime**, employ variable strides to accommodate diverse domain frequencies within a unified context window [89, 95].

7.3.3 Chronological Data Splitting

To strictly prevent temporal data leakage, random shuffling is prohibited. Datasets are partitioned in strict chronological order. The standard partitioning ratios employed in benchmarks such as ETT, Weather, and Traffic are commonly used. For zero-shot evaluation, the training set is not utilized for parameter updates, and the model is evaluated directly on the test split of the target domain to assess generalization performance [34, 89, 95].

7.3.4 Normalization via Reversible Instance Normalization (RevIN)

SLMs are inherently sensitive to the scale of input values and distribution shifts (non-stationarity). To mitigate this, **RevIN** is standard practice [95]. This technique involves a two-step process:

1. **Normalization:** Before patching, the input window is normalized using its local mean and standard deviation: $x' = (x - \text{mean}(x))/\text{std}(x)$.
2. **Denormalization:** After the model predicts the output, the reverse transformation is applied to restore the original scale: $\hat{y} = \hat{y}' \times \text{std}(x) + \text{mean}(x)$.

This ensures that the SLM processes a stationary distribution while the evaluation metrics are calculated on the original real-world scale [82, 95].

7.3.5 Rolling Forecast with Fixed Stride

Evaluation is never performed on a single static window. Instead, a rolling window strategy is employed where the input window slides across the test set with a fixed stride. This generates a comprehensive set of samples covering various temporal dynamics, ensuring that the reported metrics (MSE/MAE) are statistically robust and not biased by a specific starting point [65].

8 Key Challenges and Limitations

While SLMs have demonstrated transformative potential in extracting complex temporal patterns and performing textual reasoning, transitioning them from the domain

of NLP to time series forecasting entails significant structural and operational obstacles [98]. Unlike textual data, which possesses defined grammar and discrete vocabulary, time series data is inherently continuous, noisy, and it lacks explicit semantics. This fundamental disparity leads to challenges in aligning the representation space of language models with temporal dynamics, a pervasive issue in the literature often referred to as the "modality gap" or heterogeneity in intrinsic data patterns [16]. Furthermore, while LLMs benefit from high generalization capabilities due to training on massive corpora, SLMs often face limitations in learning capacity. Consequently, in scenarios characterized by data scarcity or high noise levels, They are prone to overfitting irrelevant patterns and often show reduced performance over long prediction horizons [24, 68].

Moreover, the non-stationary nature of real-world data and statistical distribution shifts over time exacerbate the adaptation challenge for these models, which typically require diverse and high-quality data for effective generalization [98]. Finally, given that a primary objective of developing SLMs is to facilitate deployment on edge devices, hardware constraints in these environments necessitate an intricate trade-off between model accuracy, inference latency, and energy efficiency [24]. In this section, we examine these challenges across four primary categories: data scarcity and irregular sampling, overfitting of small models to noisy signals, handling non-stationarity and extreme events, and energy and compute constraints on edge deployment.

8.1 Data Scarcity and Irregular Sampling

The development and deployment of SLMs for time series forecasting face fundamental obstacles stemming from the intrinsic nature of real-world data. In contrast to NLP and CV, which benefit from vast and standardized data repositories, time series modeling grapples with the dual challenge of "aligned data scarcity" and "structural heterogeneity in sampling." These limitations, particularly when attempting to transfer the generalization capabilities of language models to the temporal domain, necessitate novel architectural solutions [110].

8.1.1 Modality Gap and Few-Shot Learning Challenges

A primary obstacle in the application of SLMs is the "modality gap" between the continuous space of time series and the discrete space of textual tokens. When training data is limited, the alignment of these two spaces becomes increasingly difficult [49]. Early approaches, such as GPT4TS, demonstrated that by freezing the language model backbone and training lightweight adaptive layers, general model knowledge could be transferred to time series tasks with minimal computational cost [96]. However, more recent research, such as BALM-TSF, argues that focusing exclusively on a single modality (either text or time series) leads to an information imbalance. By introducing balanced multi-modal alignment mechanisms, they demonstrated that combining textual descriptive statistics with temporal encoders significantly enhances model performance in few-shot learning scenarios [107].

Beyond alignment, tokenization strategies also play a pivotal role. The Chronos model, by converting continuous values into a fixed, discrete vocabulary via quantization, reframes forecasting as a language modeling problem, thereby enabling zero-shot generalization on unseen datasets [34]. Conversely, to address data scarcity in specific domains, the UniTime model leverages cross-domain learning. By utilizing "domain instructions" and unifying feature spaces, this model allows the system to exploit knowledge from data-rich domains to improve forecasting in data-poor domains [89]. Furthermore, in scenarios characterized by isolated and dispersed data (data silos), the TIME-FFM framework simultaneously manages privacy concerns and local data scarcity through a federated learning approach and the use of adaptive prompts [110].

8.1.2 Handling the Complexities of Irregular Sampling

While standard benchmarks are often predicated on the assumption of regular sampling, real-world data (such as IoT sensors or medical events) are inherently irregular and event-driven. Chang et al. [122], in the TIME-IMM study, demonstrate that conventional models suffer severe performance degradation when confronting "structural irregularities". They classify these irregularities into three categories: (1) Trigger-based, where sampling depends on the occurrence of an event; (2) Constraint-based, stemming from system limitations; and (3) Artifact-based, resulting from recording errors [122].

To overcome this challenge in lightweight models, the Tiny Time Mixers (TTM) architecture has introduced the "resolution prefix tuning" approach. This method enables the model to adapt to varying temporal resolutions and variable history lengths without the need for extensive retraining, delivering stable performance even under conditions of sparse data [117]. Similarly, models such as SMETimes have provided greater flexibility in integrating asynchronous temporal data with textual information by replacing rigid tokenizers with "fusion embedding" modules [49].

8.2 Overfitting Small Models to Noisy Signals

One of the fundamental challenges in deploying SLMs and TSFMs is their inherent sensitivity to noise and a strong tendency to overfit on random signals [68]. Although these models have a strong ability to learn complex patterns, recent empirical studies show that their performance declines considerably on time series with high noise and long periodic patterns. In some cases, simpler approaches, such as linear autoregressive (AR) models and DLinear-based methods, achieve better accuracy [68, 103]. Unlike statistical methods that are often grounded in simplifying assumptions, Transformer-based models lack specific theoretical assumptions and, when encountering unseen data with arbitrary statistical properties, exhibit unpredictable behavior and high vulnerability to noise [68].

8.2.1 Structural Vulnerabilities and Spurious Channel-wise Correlations

One major cause of this overfitting originates from the tokenization design of Transformer models [123]. Lu et al. [123] showed that conventional temporal-wise tokenization, used in models such as Informer [124] and Autoformer [125], introduces a problem known as "timestep mixing." This issue causes the model to rely excessively on cross-variable interactions at individual time steps, leading to the learning of spurious correlations between channels [123]. As a result, when real-world data contain weak or noisy inter-variable relationships, the model may interpret noise as meaningful patterns, which increases the risk of overfitting [123]. Similarly, Zeng et al. [103] found that Transformer models often struggle to capture accurate temporal dependencies in long sequences and tend to overfit temporal noise instead of learning meaningful temporal patterns [103].

8.2.2 Data Redundancy in Rolling Window Training

Beyond model structure, data generation strategies also play a role in exacerbating overfitting. The use of rolling forecasting for data augmentation results in the generation of samples with high overlap, leading to the phenomenon of "data redundancy." Calculations indicate that in long-term forecasting, the similarity between two consecutive training samples can exceed 99%; this issue severely limits the diversity of training samples and reduces the model's generalization capability, as the model is driven towards memorizing repetitive data and their accompanying noise rather than learning generalized patterns [126].

8.2.3 Aliasing Distortion in Tokenization

In the paradigm of time series language models, the process of discretization and converting continuous signals into tokens (tokenization) serves as another source of noise introduction. If the token length is fixed and the sampling rate does not comply with the Nyquist theorem, high-frequency noise is reflected into lower frequency bands, a phenomenon known as aliasing distortion. This distortion causes the model to train on corrupted signals, which not only degrades signal quality but also intensifies the model's sensitivity to input noise [127].

8.2.4 Novel Strategies for Mitigating Overfitting

To address the aforementioned challenges, recent studies have introduced several effective strategies. One prominent direction focuses on adaptive spectral filtering. For example, the TSLANet model incorporates adaptive spectral blocks that employ Fourier analysis to transform data into the frequency domain and suppress high-frequency noise through adaptive thresholding [128]. This mechanism helps the model avoid overfitting to transient fluctuations while preserving the dominant temporal patterns. Similarly, the Apollo-Forecast framework introduces an Anti-Aliasing Quantization Module (AAQM), which applies Butterworth filters prior to tokenization to remove high-frequency noise and improve signal fidelity [127, 128].

Another line of research addresses data redundancy through curriculum learning strategies. The CLMFormer model adopts a progressive noise injection mechanism in which Bernoulli noise is gradually added to training samples with increasing dropout rates [126]. By progressively increasing the difficulty of the training process, the model is encouraged to learn robust sequential representations rather than relying on superficial similarities within the data.

Recent work has also explored token reformulation and adaptive pattern matching to improve robustness. Instead of representing tokens as raw temporal values, the ICTSP model defines tokens as forecasting tasks consisting of lookback–future pairs [123]. This reformulation enables the model to perform adaptive pattern matching through in-context learning rather than memorizing fixed temporal structures. Consequently, when processing noisy signals, the model can dynamically reduce its effective complexity and behave similarly to simpler linear predictors, such as DLinear [103], thereby reducing the risk of overfitting [123].

8.3 Handling Non-stationarity and Extreme Events

Addressing non-stationarity and forecasting extreme events remain fundamental and unresolved challenges in the application of SLMs and Transformers for time series analysis [129–131]. Unlike stationary data, where statistical properties remain invariant over time, real-world data are often subject to continuous joint distribution shifts and sudden fluctuations [129, 131]. These phenomena cause models trained on historical data to exhibit performance degradation when generalizing to future data [66].

8.3.1 The Stationarization Paradox and Distribution Adaptation

One prevalent strategy to mitigate distribution shifts involves the use of stationarization techniques, such as instance normalization or RevIN, which have been employed in foundation models like MOMENT. By eliminating local mean and variance, these methods transform the data into a standardized distribution to facilitate the model learning process [95]. However, Liu et al. [129] argue that this approach can lead to a detrimental phenomenon termed "over-stationarization." When the intrinsic statistical information of the time series is removed, Transformer models lose their ability to distinguish between different series and model specific temporal dependencies. To address this issue, they proposed the non-stationary Transformers framework, which utilizes a "de-stationary attention" mechanism to reincorporate the eliminated statistical information into the attention mechanism, allowing the model to simultaneously leverage distribution stability and the intrinsic dynamics of the data [129].

8.3.2 Adaptive Strategies and Concept Drift Detection

In addition to distribution shifts, the reliance on fixed time windows in traditional methods hinders the model's adaptation to concept drift [131]. Recent research indicates that language models must be equipped with adaptive mechanisms. Cao et al. [94] demonstrated with the TEMPO model that by decomposing the time series into trend and seasonal components and subsequently applying efficient parameter fine-tuning

(such as LoRA) to each component, the model can be adapted to data distribution shifts without the need for full retraining [94].

Furthermore, to counter sudden changes in data distribution, "detect-then-adapt" strategies have been proposed. Zhang et al. [66] introduced the D^3A framework, which, instead of indiscriminate adaptation, first detects the occurrence of drift in the data and subsequently updates the model to align with the shifting distribution [66]. In a similar approach, Pawar et al. [114] utilized the PELT algorithm to identify change points, ensuring the model is retrained only when necessary based on post-change data, thereby enhancing efficiency and accuracy in dynamic environments (Table 9) [114].

8.3.3 Managing Extreme Events and Severe Volatility

Forecasting extreme values that reside in the tails of the probability distribution (such as stock market crashes or air pollution peaks) presents a challenge where averaging-based models (e.g., those using MSE loss) exhibit shortcomings. By proposing a hybrid method based on Self-Organizing Maps (SOM), López-Gonzales et al. [130] demonstrated that time series behavior under extreme conditions is governed by distinct dynamics. By clustering temporal patterns and allocating specialized models (MLP) to each cluster, they were able to enhance prediction accuracy for rare events [130].

In the new generation of language models, the utilization of external knowledge and MoE architectures has been proposed to address this issue. Fan et al. [49] employed a hybrid architecture in the SMETimes model, utilizing a learnable gating mechanism to manage volatility. This mechanism dynamically adjusts the contribution of different internal modules—balancing between numerical projection experts (which capture stable statistical patterns like ARIMA) and semantic embeddings derived from the language model. Under volatile conditions or extreme events, the gating system adaptively shifts the model's focus to the most relevant representation, thereby maintaining accuracy where monolithic language models might otherwise fail due to modality misalignment [49].

Furthermore, Xiao et al. [45] demonstrated with the FinSrag model that language models can leverage a RAG approach to extract similar historical patterns (such as patterns preceding past financial crises) from a knowledge base. This approach enables the model to identify and forecast extreme events not as noise, but as recurrent patterns within specific contexts [45]. Additionally, Srinivas et al. [131] showed that integrating LLM-generated textual descriptions with numerical data can deepen the model's understanding of anomalies and provide more precise uncertainty estimation for decision-making in critical conditions [131].

8.4 Energy and Compute Constraints on Edge Deployment

The deployment of SLMs and time series forecasting models on edge devices, such as IoT sensors, drones, and embedded platforms, confronts fundamental challenges regarding energy and computational constraints [59–61, 132]. Unlike cloud infrastructures that possess virtually unlimited access to processing resources, edge environments face severe limitations in battery capacity, memory bandwidth, and

operational throughput [59–61, 132]. Recent research indicates that optimizing these models necessitates a multifaceted approach involving architectural redesign, advanced quantization, and collaborative inference strategies [49, 59, 60, 62, 132].

8.4.1 Computational Bottlenecks and Architectural Optimizations

The attention mechanism at the core of Transformer architectures poses a fundamental scalability challenge: its time and memory complexity grows quadratically with the sequence length, making long-horizon or high-frequency time series particularly expensive to process [81, 132]. A primary obstacle in deploying Transformer-based models is the computational complexity and memory consumption, which scales quadratically $\mathcal{O}(L^2)$ with the input sequence length [132]. This issue becomes a fundamental bottleneck in long-term forecasting, where both the look-back window length and the forecast horizon increase [101]. To mitigate this overhead, efficient Transformers employ sparse access patterns or low-rank approximations to reduce complexity to $\mathcal{O}(L \log L)$ or even $\mathcal{O}(L)$[81]. However, in edge device deployment, the primary limitation is often memory bandwidth rather than computational power [72, 76]. Research indicates that in large models, loading weights from memory to the processor accounts for a major portion of the latency [76]. This challenge becomes critical in real-time monitoring applications, such as marine environment monitoring requiring long-term forecasting, where resources are scarce [132]. To overcome this limitation, Teng et al. [132] introduced the MarineEdgeformer model. By replacing the standard attention mechanism with "linear attention" and removing start tokens from the decoder structure, they succeeded in reducing computational costs (FLOPs) by 34.5% and significantly decreasing inference time, without compromising prediction accuracy over long horizons (e.g., 288 time steps) [132].

While Transformer models offer superior modeling capabilities, they face significant competition from highly efficient linear architectures in severely resource-constrained environments. For instance, the non-Transformer model **SWIFT** has been proposed, which utilizes Discrete Wavelet Transform (DWT) and a sub-series mapping strategy to enable nearly lossless downsampling [133]. This approach allows the SWIFT model to demonstrate competitive or superior performance while possessing only 15% of the parameters of other lightweight models (such as FITS [134]), serving as an efficiency benchmark for deployment in highly constrained environments [133].

8.4.2 Quantization Strategies and Memory Management

Even after architectural optimizations, the memory footprint of Transformer-based models remains prohibitive for many edge devices, where storing and loading full-precision weights (FP16/FP32) dominates energy and latency budgets [76]. Quantization techniques play a vital role in reducing memory footprint and energy consumption [59]. Quantization reduces model size and memory traffic by lowering the precision of numerical representation (e.g., from FP16 to INT4) [77]. The GPTQ method has demonstrated that weights of very large models can be reduced to 3 or 4 bits using second-order derivative information (Hessian) without significant loss in model accuracy. This method enables the execution of models with billions of parameters on a single GPU, which is critical for edge scenarios [76].

In time series, extreme fluctuations in input data (outliers) can make the quantization of activation functions difficult. The SmoothQuant method, by shifting the quantization difficulty from activations to weights via a mathematically equivalent transformation, enables full W8A8 quantization (8-bit weights and activations), leading to an inference speedup of up to 1.56× [77].

Shakhadri et al. [59] developed the **Shakti-LLM** model (with 2.5 billion parameters), demonstrating that utilizing 4-bit quantization alongside innovations such as "Variable Grouped Query Attention" (VGQA) can drastically accelerate inference speeds on edge devices (like smartphones and IoT) [59].

However, uniform quantization is not always a sufficient solution for preserving accuracy [62]. In implementing Transformers on embedded FPGAs (such as Spartan-7), Ling et al. [62] found that complex models become undeployable due to inefficient use of Block RAM (BRAM) and saturation of distributed memory (DRAM). They developed a resource-aware mixed-precision quantization method. By dynamically assigning different bit-widths (4, 6, or 8 bits) to different model layers, they successfully deployed models that were previously impossible to implement due to hardware resource constraints [62].

8.4.3 Energy-Accuracy Trade-off Analysis

In critical scenarios such as urban sewer overflow management, where devices must operate autonomously on battery power, the balance between prediction accuracy and energy consumption is decisive [61]. Beyond the energy–accuracy trade-off, inference speed is another closely related challenge that limits the practical deployment of language models in time series forecasting. For instance, it has been reported that the inference of the Time-LLM model on the ETTm1 dataset is approximately 13 times slower than lightweight classical models such as iTransformer. To address this problem, frameworks like TimeKD utilize "Privileged Knowledge Distillation". In this method, a large teacher model, which leverages future data (as privileged information) during training, transfers its knowledge to a lightweight student model. The student model, possessing a simpler architecture, is capable of delivering prediction accuracy close to that of the teacher model during testing, but with significantly higher speed and lower memory consumption [101].

Through hardware-aware hyperparameter search, Ling et al. [61] showed that although 8-bit Transformer models offer higher accuracy (MSE of 0.0376) compared to LSTM models, their energy cost per inference (0.370 mJ) is over 40 times that of LSTM models (0.009 mJ). These findings suggest that in applications where battery life is the primary priority, simpler models (like LSTM) may still be preferred over SLMs despite lower accuracy (approximately 14.89% higher error) [61]. Additionally, in benchmarking CPU load prediction models on IoT gateways, Cajas et al. [63] demonstrated that offline ensemble methods (like XGBoost) and online learning methods (like ARF) exhibit distinct memory consumption and training time profiles, which must be selected according to the specific constraints of the edge device [63].

8.4.4 Collaborative and Specialized Inference Architectures

Executing complete SLMs on edge devices is often infeasible due to the combined constraints of memory, computation, and energy. This motivates architectures that distribute or specialize the inference workload [49, 60]. To overcome the inherent limitations of running full SLMs at the edge, novel architectures have shifted towards edge-cloud collaboration and the use of specialized models [60]. She et al. [60] introduced the token level routing system, wherein a small model (SLM) runs on the edge device, and only tokens with high uncertainty (approximately 7%) are sent to a large model in the cloud for processing. This collaborative architecture improves model accuracy in complex reasoning tasks by up to 60% while reducing latency and preserving privacy [60].

In another approach to enhance parameter efficiency, Fan et al. [49] developed the SMETimes model featuring a "dynamic MoE" structure. By selectively activating specialized modules for different temporal patterns and employing adaptive embeddings, this model achieved a 5.2x reduction in memory consumption compared to conventional LLMs while achieving superior performance in time series forecasting [49].

In edge computing scenarios, predictions are typically performed in real-time with a batch size of one. Under these conditions, traditional models suffer from memory bottlenecks. The "Parallel Scaling (ParScale)" approach, by performing computations across multiple parallel streams sharing common parameters, converts the memory bottleneck into a computational bottleneck, which is more favorable for modern processors [135]. Field research on mobile devices (such as Jetson Orin) indicates that for processing historical windows (prefill), edge GPUs exhibit significantly higher performance compared to CPUs; however, in long forecast horizons, mobile CPUs suffer severe performance degradation due to thermal throttling [72]. Therefore, employing hybrid strategies such as ParScale or distilled models offers a sustainable solution for maintaining long-term efficiency [135].

9 Ethical, Privacy & Reliability Considerations

The application of language models in time series analysis has opened up new frontiers in automated decision-making; however, this development is accompanied by profound ethical and security complexities. In sensitive domains such as economics and health, model outputs directly affect the welfare of society and the health of individuals, and any malfunction in performance can have unintended consequences [136]. The main challenge here is the opaque or so-called "black box" nature of deep models, which prevents human users from understanding the underlying logic of the predictions. On the other hand, the process of fine-tuning these models on sensitive data is always associated with the risk of private information leakage and inference attacks [69]. Also, the inherent vulnerability of Transformers to adversarial attacks can severely impair the reliability of the system in the face of manipulated inputs [70]. In addition, the transfer of statistical and cultural biases from large models to smaller models during the training process poses a serious challenge to fairness in the distribution of resources and opportunities [71].

9.1 Bias and Fairness when Forecasting Sensitive Metrics

One prominent approach to detecting bias involves counterfactual fairness testing. In this approach, sensitive input features (such as demographic identifiers in credit profiles) are subjected to controlled perturbations, while other causal variables are held constant. If the model makes a different prediction in response to a change in a sensitive variable (such as changing the gender of the input without changing the transaction history), a fairness violation is identified. This approach allows developers to test the robustness of the model in the protected feature space.

Unlike model-driven approaches that focus on modifying the architecture, the data-driven strategy emphasizes improving the quality and balance of the training dataset. This approach is particularly critical in financial forecasting, where data often suffers from a "representation gap." The process includes techniques such as re-weighting minority-group samples and generating fair synthetic data to fill statistical gaps. By correcting the data distribution before training, biases in estimating sensitive metrics such as default risk can be reduced, thereby mitigating the reinforcement of historical stereotypes [137].

One of the innovations introduced in 2025 is the **ThinkSLM Benchmark**, which aims to move beyond evaluating only the final output and instead examine the internal reasoning process of small language models. By forcing the model to generate a Chain-of-Thought, this benchmark makes the intermediate layers and logical reasoning steps of SLMs more transparent. This transparency allows researchers to identify the stage of information processing (for example, feature extraction or knowledge synthesis) at which bias is introduced and consequently mitigate the logical roots of bias before they influence the final prediction [138].

Another emerging direction from information theory evaluates models through the lens of lossless compression as a tool for measuring both mathematical accuracy and distributive fairness. This criterion is based on the assumption that a fair and accurate model should correctly capture the statistical complexities of all parts of the distribution, including low-frequency groups in economic data. Using bit-rate as a measure of predictive capability reveals anomalies that traditional metrics such as MSE may fail to detect. This form of mathematical evaluation helps uncover hidden inequalities in socio-economic forecasting and improves the reliability of the model [138].

9.2 Data Privacy in Edge/IoT Scenarios

Deploying SLM on edge devices and IoT sensors inherently reduces the risk of sensitive information disclosure by eliminating the need to transfer raw data to the cloud. However, SLM nodes in autonomous agent networks are highly vulnerable to "on-device tampering" and adversarial attacks on lightweight inference paths, which can lead to the propagation of corrupted signals throughout the network. To address this crisis, **FFMs** such as **TIME-FFM** have been proposed, which allow time series models to be trained collaboratively, without sensitive time data ever leaving their original source [49, 139].

Additionally, in scenarios where model personalization is required for specific IoT users, the **PriME framework** minimizes the risk of data leakage among data-sharing users by utilizing evolutionary integration of models and gradient-free methods, creating an optimal balance between prediction accuracy and privacy budget. Finally, with increasingly stringent privacy and AI governance frameworks such as the GDPR and the EU AI Act, it is imperative to implement robust security protocols at the edge layer to prevent the risks of "Shadow AI" (i.e., the unauthorized use of models without IT oversight) in IoT ecosystems, thereby ensuring that sensitive time series data remains secure and properly controlled [71].

9.3 Ensuring Reliable Alarms and Avoiding "Hallucinated" Forecasts

One of the fundamental challenges in deploying SLM for time series forecasting is the phenomenon of "numerical hallucination," a situation in which the model produces numbers that, despite their logical appearance, contradict physical laws or historical patterns in the data. This challenge can lead to disastrous decisions, especially in sensitive systems that operate on an alarm basis. To combat hallucinations, advanced prompt engineering approaches such as TFSC have been developed. In contrast to direct forecasting, this protocol forces the model to use a "Chain-of-Thought." In this process, the model must first analyze and describe the time series components, including trend, seasonality, and remainder, in a textual manner. This intermediate step acts as a "logical anchor" and ensures that the final predicted numbers are derived from a structural understanding of the signal behavior and not simply random sequences of tokens [97]. A point forecast alone is not sufficient for warning systems, as it increases the risk of "false alarms." Advanced models such as Chronos have addressed this shortcoming by shifting the paradigm from single-number forecasting to distributed forecasting. Using weighted quantile loss functions, these models generate confidence intervals. In operational scenarios, the warning system is only activated when the "lower bound" or "upper bound" of the probability distribution crosses a critical threshold. This probabilistic approach ensures the reliability of the system in the face of environmental noise and avoids the illusion of uncertainty [34]. Attackers can purposefully alter the model output by applying very small perturbations that would appear as noise in conventional statistical analysis. These attacks can create two dangerous scenarios: 1. missed alarms, in which a real critical situation is ignored by the model; and 2. targeted false alarms, which force the system to take costly and unnecessary actions. Identifying these vulnerabilities is essential for securing critical IoT infrastructure [70]. For the final validation of the alarms, the model must provide explanations that are understandable to human experts. The **Domain-Adaptive Explanation Mechanism (DAEM)** works by aligning the model logic with the metadata of the specialized domain (such as laws of physics or medical protocols). If an SLM issues an alert but cannot provide a logical cause consistent with domain knowledge in its interpretable layer (such as attention maps or patterns extracted by DAEM), that prediction is labeled as "illusion" and removed from the decision cycle. This process ensures the robustness of alerts even in resource-constrained environments by using distilled models as observers [69, 136].

10 Case Studies & Real-World Deployments

The transition of SLMs and compact foundation models for time series from purely research-oriented environments to operational systems is gradually reshaping the design of forecasting and monitoring frameworks [117]. While massive models exhibit high generalization capabilities, they often come with significant computational costs, high latency, and dependency on cloud infrastructure. In contrast, many emerging compact models prioritize computational efficiency, targeting deployment in resource-constrained environments [117, 140]. This section examines empirical evidence across three application domains—retail and e-commerce, industrial IoT, and smart healthcare—demonstrating how compact deep learning models and architectures inspired by language models are being utilized in practice.

10.1 Demand Forecasting in Retail and E-Commerce

Time series data in retail typically involve a vast number of items with heterogeneous statistical patterns and varying time horizons, characteristics that render demand forecasting a challenging problem. This heterogeneity has led many operational systems to rely on classical models, such as Gradient Boosted Decision Trees (GBDT), due to their scalability and acceptable performance in industrial scenarios [54]. However, these models generally lack explicit mechanisms for cross-domain knowledge transfer or the leverage of textual and event-driven information [89].

In response to this limitation, some research has shifted towards unified architectures that jointly model temporal representations and high-level cues. **UniTime** is an example of this approach, utilizing tokens and domain-based conditioning mechanisms to enable the transfer of knowledge learned in data-rich domains to forecasting tasks in data-scarce domains. Reported results indicate that this approach can improve forecasting performance in zero-shot scenarios under certain settings, although the extent of improvement depends on the data type and domain [89].

In situations where demand fluctuations are influenced by complex external factors such as supply chain disruptions or economic shocks, some studies have adopted the concurrent use of structured and unstructured data. In this context, hybrid models have been proposed that employ **Transformers** to process textual data for contextual understanding of disruptions, and **ARIMA** to model linear temporal trends in structured data such as trade volume and shipment delays. Empirical results suggest that this combination can enhance forecasting accuracy in certain supply chain-related scenarios compared to using either component in isolation [48].

In the retail and e-commerce domain, accurate forecasting during anomalous periods poses a fundamental challenge. Addressing this, a study by Kalifa et al. [53] demonstrates that integrating textual embeddings extracted from **world events** and news content, alongside actual sales data, can maintain demand forecasting stability during periods of high volatility. Reported empirical results indicate that this approach improves forecasting accuracy, particularly when sudden shocks cause deviations from historical patterns [53].

In the context of multivariate time series forecasting for supply chain inventory management, Wolff et al. [56] introduce a compact model that utilizes a **multivariate**

patching strategy to facilitate transfer learning across heterogeneous datasets. Experiments conducted on real-world supply chain data show that this approach can deliver performance comparable to more advanced models in strategic decision-making, though the degree of improvement is dataset-dependent [56].

From the perspective of noise robustness, some research focuses on simplifying input signals. The **StreamTS** model attempts to mitigate the impact of severe fluctuations by decomposing time series into trend and seasonal components and applying smoothing filters, thereby providing a more stable and suitable input for deep learning models. Reported results indicate that this approach can improve downstream model performance, particularly in real-world datasets characterized by high noise and irregular fluctuations [97].

Finally, the **Chronos** model adopts a distinct approach by mapping time series values to discrete symbols and formulating the forecasting problem as a classification task. This framework enables the extraction of forecast distributions and uncertainty estimation, with competitive performance reported across a range of decision-oriented forecasting tasks and benchmarks involving demand and sales time series. However, Chronos is not a general-purpose language model and should be classified as a specialized time series foundation model [34].

10.2 Predictive Maintenance and Industrial IoT

In environments with severe computational and energy constraints, often observed in industrial monitoring systems and the IoT, real-time equipment monitoring requires efficient models. To this end, compact models such as **Tiny Time Mixers (TTM)** have been developed. With a low parameter count and architectures based on channel and temporal mixing, they enable multivariate forecasting at a low computational cost. Empirical results show that these models offer competitive performance on certain time series forecasting benchmarks and are suitable for execution on CPUs or edge hardware [117].

For applications where energy consumption is critical, such as environmental or marine sensor networks, architectures with linear time complexity have been proposed. The **MarineEdgeformer** model reduces computational complexity by replacing the standard Transformer attention mechanism with **linear attention** (reducing complexity from $\mathcal{O}(L^2)$ to $\mathcal{O}(L)$). Reported experiments demonstrate a favorable balance between forecasting accuracy and energy consumption. These results suggest that architectural modifications can play a significant role in enhancing model deployability in edge environments, although long-term field evaluations remain limited [132].

In critical industries such as oil and gas, anomaly prediction is directly linked to reducing operational losses. In this domain, some studies have utilized compact time series foundation models that employ language model architectures and principles as a backbone to forecast discrepancy values in **custody transfer** systems. Reports indicate that these approaches outperform classical statistical methods in predicting such discrepancies [58]. Furthermore, case studies on industrial equipment data, such as compressors, show that approaches based on generative language models for time series reconstruction and modeling can achieve high accuracy in anomaly detection

under laboratory conditions, although results are heavily dependent on data type and experimental settings [57].

10.3 Smart Healthcare and Privacy Considerations

In smart healthcare applications, the analysis of physiological time series faces dual challenges: the high sensitivity of data necessitating privacy preservation, and the need for real-time processing which mandates edge deployment. These constraints limit the applicability of massive cloud-dependent models and have shifted research focus towards developing lightweight, on-device models [140].

In this regard, Lee et al. [140] demonstrate that efficient representations of PPG signals can be extracted using **self-supervised learning** approaches and optimized **Convolutional Neural Network (CNN)** architectures. Empirical results suggest that these compact models, despite having significantly fewer parameters than prevalent Transformer models, can provide acceptable performance in clinical classification tasks and serve as suitable options for computationally resource-constrained environments [140].

Additionally, integrating the semantic capabilities of language models with biosignal analysis has garnered attention. The **ECG-LLM** model leverages feature alignment techniques to bridge the gap between textual representations and Electrocardiogram (ECG) signals. Reports indicate that this approach shows promising potential in zero-shot disease diagnosis scenarios with limited access to labeled data [141].

In the realm of generative modeling, the **Auto-TTE** model offers a framework for synthetic data generation where ECG signals are synthesized based on clinical text descriptions (**text-to-ECG**). Although primarily proposed for enriching training data and preserving privacy, its ability to model complex relationships between text and signal waveforms represents a step forward in medical generative models [142].

Finally, models such as **MedTsLLM** have focused on integrating time series analysis with textual explanations to enhance interpretability. These research models demonstrate how multimodal architectures can provide relevant textual insights alongside anomaly detection to assist the clinical decision-making process [55].

In summary, existing evidence suggests that specialized, compact models—ranging from lightweight on-device models to generative and interpretable frameworks—can offer practical and reliable solutions for medical time series analysis in privacy-sensitive environments, narrowing the gap between research and real-world deployment in healthcare systems.

10.4 Selected Practical Insights

The transition from laboratory-scale experimentation to industrial operational deployment exposes challenges that extend well beyond incremental improvements in predictive accuracy. Although foundation and language models have demonstrated considerable promise in time series forecasting, many existing approaches remain inadequate for addressing the complexities associated with large-scale datasets and the operational demands of industrial environments. Recent studies further indicate that

even state-of-the-art models in critical sectors such as energy and finance are susceptible to systematic and predictable errors that cannot be reliably mitigated without external oversight or corrective intervention [143, 144].

Moreover, a substantial portion of current forecasting methodologies fails to effectively incorporate the rich semantic information and domain-specific knowledge embedded in textual documents, technical reports, and expert analyses. At the same time, the inherent limitations of machine learning systems imply that knowledge extracted automatically by models cannot fully substitute for human expertise and contextual understanding. In parallel, the substantial computational and financial costs associated with large-scale language models have intensified the demand for cost-efficient deployment strategies capable of justifying adoption in commercial and industrial settings [145, 146].

Consequently, this case study demonstrates that the successful deployment of SLM-based forecasting systems in real-world domains, including retail and manufacturing, depends not solely on advances in model architecture, but rather on the integration of three fundamental pillars: **(i) seamless compatibility with legacy infrastructure and resource-constrained hardware, (ii) the establishment of human-in-the-loop correction and monitoring mechanisms to ensure long-term reliability and sustainability, and (iii) the rigorous definition of return-on-investment (ROI) metrics grounded in operational cost and efficiency considerations** [147].

10.4.1 Integration with Legacy Systems

The transition of SLM-based forecasting systems from controlled research settings to industrial operational environments introduces substantial challenges related to compatibility with legacy infrastructure. In many industrial domains, forecasting systems must operate alongside decades-old platforms characterized by limited memory bandwidth, constrained computational capacity, FPGA- or CPU-based embedded architectures, and strict energy budgets. Consequently, the practical deployment of modern forecasting models depends not only on predictive capability, but also on their ability to adapt to highly resource-constrained environments.

Recent studies have demonstrated that conventional model compression techniques are often insufficient for embedded industrial systems. In FPGA-based infrastructures with limited on-chip memory, uniform quantization strategies may lead to distributed memory saturation and unstable inference behavior. To address this limitation, resource-aware mixed-precision quantization techniques have been proposed, allowing different model layers to operate at heterogeneous precision levels (e.g., 4-, 6-, or 8-bit representations) according to hardware constraints and workload characteristics [62]. Such strategies enable SLMs to fit within the memory limitations of legacy hardware while maintaining acceptable forecasting accuracy.

The integration process also requires careful consideration of the trade-off between predictive performance and energy efficiency. In energy-sensitive edge applications, such as industrial monitoring or wastewater management systems, replacing lightweight recurrent architectures with Transformer-based models may not always be

operationally justified. Empirical evidence indicates that although quantized Transformer models may achieve lower prediction errors than LSTM-based alternatives, their energy consumption can be several orders of magnitude higher [62]. As a result, hybrid deployment strategies are often preferred, whereby lightweight recurrent models remain responsible for continuous low-power monitoring tasks, while more sophisticated SLMs are selectively activated for complex forecasting scenarios.

Another important aspect of industrial integration involves maintaining compatibility with long-established statistical forecasting pipelines. In sectors such as logistics and supply-chain management, classical models including ARIMA and Exponential Smoothing have been calibrated and optimized over many years of operational use. Rather than replacing these systems entirely, recent approaches increasingly emphasize algorithmic symbiosis between statistical forecasting methods and language-model-based architectures. For example, the **StreamTS** framework decomposes time series into trend and seasonal components, assigning complex trend modeling to SLMs while preserving seasonal estimation through lightweight linear or ARIMA-based modules [97]. Similarly, hybrid architectures such as the **Transformer-ARIMA** model combine structured numerical forecasting with textual reasoning by integrating ARIMA-based temporal analysis with Transformer-based processing of unstructured textual data, including reports and disruption-related news [48]. The **Autoformer** architecture further advances this direction by embedding decomposition mechanisms directly within the model structure, thereby reproducing the interpretability and periodic decomposition logic of traditional statistical approaches while benefiting from modern deep learning capabilities [125].

Operational deployment in legacy environments additionally necessitates efficient management of computational and bandwidth resources. Continuous execution of large forecasting models is often infeasible in industrial infrastructures with limited processing capacity. To address this challenge, recent studies have proposed hierarchical inference strategies based on model cascading. Within the **FrugalGPT** framework, lightweight local models act as first-stage responders, while computationally expensive cloud-based models are invoked only when prediction uncertainty exceeds predefined thresholds or when more sophisticated reasoning is required [147]. This layered architecture substantially reduces inference costs, alleviates pressure on legacy infrastructure, and improves operational scalability.

Finally, seamless migration from traditional forecasting pipelines to intelligent forecasting systems requires minimizing disruption to existing industrial workflows. Complete replacement of legacy platforms is often impractical because of proprietary communication protocols, tightly coupled software ecosystems, and substantial infrastructure costs. Consequently, recent research has explored reprogramming strategies that adapt foundation models to existing operational systems without requiring architectural replacement. Techniques such as dynamic prompting and grouped-query multi-head attention enable language models to process spatio-temporal industrial data while remaining compatible with consumer-grade hardware and pre-existing enterprise infrastructures [143].

10.4.2 Human-in-the-Loop Corrections

Although SLMs are capable of extracting highly complex temporal dependencies, they remain vulnerable to abrupt environmental changes, non-stationary dynamics, and out-of-distribution events frequently encountered in operational environments. As a result, recent research has increasingly emphasized the importance of integrating human expertise into the forecasting process. Contemporary human-in-the-loop (HITL) systems extend beyond traditional manual correction mechanisms and instead establish interactive and adaptive collaboration frameworks between human experts and forecasting models.

A major development in this direction is the emergence of natural-language-driven adaptive optimization systems. These approaches enable domain experts to modify forecasting behavior through textual instructions without requiring model retraining or architectural modifications. In model-agnostic frameworks, human feedback expressed in natural language is translated into mathematical optimization actions through language agents and subsequently incorporated into forecasting pipelines using reinforcement learning or contextual bandit mechanisms [144]. This paradigm allows organizations to incorporate real-time expert knowledge into operational forecasting systems while avoiding costly downtime and retraining procedures.

Recent HITL architectures have also introduced iterative prediction-reflection-refinement cycles designed to continuously improve forecasting reliability. The **TimeXL** framework exemplifies this direction through its LLM-in-the-loop paradigm, in which a dedicated reflection model evaluates forecasting outputs by comparing predicted values against observed environmental conditions [145]. Potential inconsistencies and anomalous predictions are subsequently reviewed by human operators, whose feedback guides selective refinement and targeted retraining procedures. Over time, this closed-loop interaction enables the system to better distinguish genuine temporal signals from environmental noise and transient disturbances.

The effectiveness of human intervention further depends on the interpretability and transparency of forecasting systems. Modern HITL frameworks increasingly employ case-based reasoning and prototype-driven explanations to provide operators with historical examples analogous to current forecasting scenarios. Such interpretability mechanisms allow human experts to determine whether forecasting errors originate from deficiencies in the training data or from structural changes in the underlying environment, such as concept drift [145]. Consequently, human corrections become more targeted, reliable, and operationally meaningful.

At industrial scale, however, continuous human supervision may itself become prohibitively expensive and cognitively demanding. To mitigate expert fatigue and reduce operational overhead, recent HITL systems selectively request human intervention only under conditions of elevated predictive uncertainty or strategic risk. This selective interaction strategy ensures that human expertise is reserved for high-impact situations, including financial crises, energy instability events, or critical equipment failures, where the consequences of forecasting errors are particularly severe [146, 147].

10.4.3 ROI Metrics (Return on Investment)

Evaluating the return on investment (ROI) of SLM-based forecasting systems requires a broader perspective than traditional predictive accuracy metrics alone. In industrial deployments, profitability and operational viability are determined not only by forecasting precision, but also by inference cost, energy consumption, infrastructure compatibility, maintenance overhead, and the strategic value of human intervention mechanisms.

One of the primary dimensions of ROI involves achieving an optimal balance between forecasting performance and operational expenditure. Recent research has shown that model cascading frameworks can substantially reduce inference costs by dynamically allocating forecasting requests between lightweight local models and computationally expensive cloud-based systems [147]. Such adaptive deployment strategies enable organizations to reserve high-cost computational resources for only the most complex forecasting tasks, thereby accelerating the financial viability of large-scale industrial deployments.

Energy efficiency constitutes another critical component of industrial ROI, particularly in edge-computing and embedded forecasting environments. Techniques such as mixed-precision quantization significantly reduce computational load and energy consumption while extending the operational lifetime of legacy hardware components [62]. Beyond lowering electricity costs, reduced thermal stress and computational overhead can delay hardware replacement cycles, thereby contributing to long-term infrastructure sustainability and improving the overall economic efficiency of forecasting systems.

In addition to direct computational savings, a substantial portion of ROI derives from mitigating the operational risks associated with forecasting errors. In sensitive domains such as finance, logistics, and energy management, inaccurate predictions may result in severe economic losses, resource misallocation, or infrastructure instability. The integration of human-in-the-loop correction mechanisms and interpretable forecasting frameworks can significantly reduce the frequency and impact of such failures [145]. In many industrial scenarios, preventing a single large-scale operational error may offset the cumulative costs associated with model development, deployment, and maintenance, thereby transforming forecasting reliability itself into a major source of economic value.

11 Future Directions and Open Research Questions

Despite the significant advances in SLMs and their proven effectiveness in modeling complex temporal patterns, their deployment in real-world operational environments represents the beginning of a long research journey, not its end. In real-world scenarios, the traditional assumption of **stationarity** rarely holds; real-world environments are highly dynamic, and data distributions, seasonal patterns, and variable relationships are constantly evolving due to external shocks and environmental changes [114]. Current static models that rely solely on knowledge fixed at the time of training quickly suffer from performance degradation and knowledge obsolescence when faced with

these inevitable dynamics [114]. Therefore, the future prospects for time series forecasting using SLMs are not limited to further compression or parameter optimization of current architectures. The future research frontier requires a **fundamental paradigm shift** from **static, unimodal, and human-involved** models to **dynamic, multimodal, and autonomous** systems [123, 148]. This fundamental transition requires new research horizons, moving away from "prediction in isolated environments" and toward the creation of lifelong learning systems. The next generation of basic models must be able to detect changes in data distributions in real time, adapt to new conditions in real time and at the time of inference, and break the boundaries of purely numerical data and combine sensor signals with texts and metadata to gain a deeper understanding of the causes of fluctuations [123, 148]. Beyond improving the cognitive capabilities of models, the widespread deployment of these technologies on edge devices and the IoT poses significant challenges in scalability, privacy of sensitive data, and overreliance on manual engineering. Overcoming these obstacles requires innovation in collaborative learning frameworks and securing data flows.

11.1 Continual and Online Learning for Evolving Time Series

Maintaining the predictive accuracy of SLMs in dynamic environments necessitates continuous algorithmic adaptation. The core technical challenge in this domain is navigating the Stability-Plasticity Dilemma: an online forecasting system must possess sufficient plasticity to rapidly integrate emerging temporal patterns and distribution shifts [114], while retaining enough stability to prevent the catastrophic forgetting of previously consolidated knowledge. To achieve this equilibrium without the prohibitive computational cost of full-parameter retraining, contemporary frameworks have shifted towards highly localized and resource-efficient adaptation mechanisms. These targeted strategies include actively monitoring error streams for drift detection, updating Neural Memory states via instantaneous gradient signals during inference, and leveraging attention mechanisms for adaptive in-context pattern retrieval [123, 148].

11.2 Theoretical Frontiers

The Stability-Plasticity Dilemma In the context of evolving time series, the fundamental challenge for any online learning system is the Stability-Plasticity Dilemma. A robust forecasting model must possess sufficient Plasticity to integrate novel patterns arising from non-stationary environments (e.g., sudden concept drifts or new seasonalities) while maintaining Stability to prevent the erasure of previously consolidated knowledge. Failure to balance these opposing forces results in Catastrophic Forgetting, a phenomenon where the optimization of parameters for a new data distribution D_{t+1} causally degrades the performance on the historical distribution D_t [149]. To mitigate this, literature proposes three primary architectural mechanisms:

1. Weight Regularization Strategies: Standard fine-tuning on new data streams typically leads to overfitting on the most recent buffer (recency bias). Regularization-based approaches, such as Elastic Weight Consolidation and Synaptic Intelligence,

address this by constraining updates to the models parameters. These methods estimate the importance of each weight for solving previous tasks (e.g., via the Fisher Information Matrix) and add a quadratic penalty term to the loss function, effectively freezing critical weights while allowing less important parameters to adapt to new drifts [150].

2. Experience and Generative Replay: Unlike regularization which fixes weights, Replay mechanisms maintain a physical buffer of historical exemplars (Experience Replay) to interleave old data with incoming streams during training. This ensures that the model's decision boundary remains robust across temporal shifts by approximating the joint distribution of past and present data. Furthermore, to circumvent storage limitations in resource-constrained edge devices, Generative Replay (or Pseudo-Rehearsal) is proposed, where a separate generative model learns to synthesize "fake" historical data to reinforce the forecaster's memory without storing raw sensitive samples [149].
3. Parameter Isolation and Expansion: A third paradigm involves dynamically allocating specific neural resources to distinct temporal regimes. Parameter Isolation methods dedicate separate model branches or "heads" for different tasks or data distributions, ensuring that updates for a new concept drift do not interfere with the weights optimized for stable historical periods. This modularity effectively isolates plasticity (in new branches) from stability (in frozen branches), albeit at the cost of increasing architectural complexity [149].

11.3 Active Drift Management

Rather than relying on passive adaptation mechanisms which may suffer from lagged responses, robust forecasting systems must actively monitor the stochastic volatility of data streams to trigger timely interventions [114]. This involves a transition from static model assumptions to dynamic frameworks capable of identifying and reacting to concept drift the phenomenon where the joint probability distribution of the input and target variables shifts over time [115].

11.3.1 Hard Adaptation via Structural Change Detection (PELT)

For environments exhibiting abrupt regime shifts, a binary Detect-then-Adapt strategy is essential to prevent model contamination from obsolete historical correlations.

11.3.2 Optimal Change Point Detection

The framework utilizes the PELT algorithm to rigorously identify structural breaks in the time series. Unlike heuristic thresholding, PELT operates as an optimization problem, minimizing a regularized cost function, where represents the likelihood of fit within a segment, is the number of change points, and is a penalty term to preventing overfitting (segmentation noise). This allows for the precise temporal localization of drifts in linear computational time.

11.3.3 Selective Retraining Protocol

Upon the detection of a significant drift point, the system triggers a Selective Retraining protocol. In contrast to standard sliding Window approaches that discard data based on a fixed arbitrary horizon, Selective Retraining dynamically sets the training boundary at. The model is effectively reset and retrained exclusively on the post-drift data segment, ensuring that the optimized weights reflect only the current prevailing statistical distribution while minimizing computational waste on irrelevant historical epochs.

11.4 Test-Time Training (TTT) and Neural Memory

TTT and Neural Memory Architectures For scenarios requiring rapid and deep adaptation to high-frequency non-stationarity, traditional periodic retraining is computationally prohibitive, especially when deploying large TSFMs. Consequently, architectures are evolving towards TTT, a paradigm where the model continuously learns and updates its internal representations during the inference pass itself, leveraging the sequentially arriving ground-truth data (online feedback) without requiring a complete backward pass through the massive pre-trained backbone.

11.4.1 Lightweight Online Adaptation and Ensemble Weighting

A primary challenge with deploying massive foundation models (e.g., Chronos, MOIRAI) is their static nature post-deployment. The Efficient Lightweight Forecasting (ELF) framework addresses this by introducing a decoupled, dual-stream architecture designed specifically for online adaptation. Rather than updating the frozen foundation models, ELF introduces a highly efficient, parallel online learner composed of two critical modules:

The ELF-Forecaster: A lightweight auxiliary model (often a simple linear or shallow recurrent network) that is continuously trained online using the latest arriving data points. Its sole purpose is to capture the immediate, rapidly shifting distribution that the static foundation models fails to recognize.

The ELF-Weighter: A dynamic gating mechanism that continuously evaluates the real-time performance of both the frozen foundation models and the online ELF-Forecaster. By computing a weighted combination of the static forecast and the adaptive forecast, the system can swiftly pivot to rely on the online learner during sudden structural breaks or anomalous spikes, thereby preventing the catastrophic failure of the frozen backbone in volatile regimes [148].

11.4.2 Neural Memory and Surprise-Based Gradient Updates

Moving beyond auxiliary linear forecasters, intrinsic model architectures are being redesigned to incorporate continuous learning directly into the attention mechanism. The **Titans framework** introduces a parameterized Neural Memory module that serves as a long-term, evolvable contextual storage. Unlike standard self-attention mechanisms which treat the context window as a static buffer, Neural Memory stores a compressed representation of the historical trajectory. This memory matrix is updated

dynamically at each time step t using a **surprise-based** learning objective. As new data arrives, the model computes the local gradient of the loss function. This gradient acts as a measure of surprise (the divergence between expected and actual observations). Instead of standard backpropagation, this gradient signal is used to instantaneously update the memory weights via a localized update rule. This grants the model effectively infinite context length and the ability to integrate novel patterns instantly [151].

11.4.3 Test-Time Normalization and Prompt Tuning

In addition to weight updates, recent advancements demonstrate that TTT can be executed entirely in the embedding or normalization space, preserving the zero-shot integrity of LLMs. Before feeding data into the LLM, the time series undergoes **RevIN**, where the mean and variance of the current test instance are removed. The model generates the forecast in a normalized, domain-agnostic space, and the instance-specific statistics are re-applied to the output. This acts as a zero-parameter Test-Time Adaptation, instantly aligning the model with the current local distribution scale. **Test-Time Prompt Tuning (TTPT)**: When utilizing LLMs for forecasting, updating the Transformer blocks online is infeasible. Instead, TTPT techniques append a set of learnable soft Prompts (continuous embeddings) to the input sequence. During inference, a lightweight gradient descent step is performed only on these soft prompts using the most recent observation window. This effectively steers the LLM's reasoning pathways to match the newly evolved distribution without altering a single weight within the billions of parameters of the core language model [152].

11.5 Multimodal SLMs Combining Text, Sensor Data, and Metadata

Despite the remarkable success of language models in time series prediction, the sole reliance on unimodal numerical data is a fundamental weakness. In real-world scenarios, sensor data fluctuations do not occur in a vacuum, but are always surrounded by metadata, textual reports, event news, and even visual patterns. Being limited to numbers severely reduces the model's capacity to understand context and find the root causes of anomalies. Therefore, the forefront of future research is the transition from purely numerical processors to multimodal SLMs that can integrate texts, signals, and metadata into a common semantic space [106, 153].

11.5.1 Aligning Historical Descriptive Texts with Predictive Insights into the Future

Most existing multimodal time series models rely only on textual information from the past (such as reports recorded up to the present moment) and neglect the value of texts that contain predictions or expectations for the future. Frameworks such as Dual-Forecaster are able to simultaneously combine historical descriptive texts with predictive textual insights and extract the complex relationships between these two textual streams and numerical data through cross-modality alignment techniques. Furthermore, for real-world scalability, the process of extracting these texts needs to be

automated. The development of prompt-based frameworks such as GPT4MTS shows how to directly connect to large news databases (such as GDELT) to extract contextual information related to time series fluctuations and feed it to language models in real time. This path could create a new standard for impact forecasting [104, 105, 107, 153].

11.5.2 Context-Aware Probabilistic Modeling

One of the major theoretical obstacles to integrating language models with time series is the conflict between the deterministic numerical decoding and the generative and probabilistic nature of these models. When language models are tuned solely to produce deterministic numbers, their vital capabilities in modeling probability distributions and contextual awareness are suppressed. Future research should focus on multimodal probabilistic modeling. The CAPTime framework is a pioneering example in this field, which extracts temporal patterns by an encoder and aligns them with the context through **learnable interactions** By using the **Mixture of Distribution Experts technique** alongside frozen SLMs, the model can provide probabilistic forecasting that is fully aware of context conditions, without losing the model's inherent linguistic ability [48, 53, 90, 154].

11.5.3 Event-Based Reasoning and Augmented Generation with Retrieval (Domain-Specific RAG)

In domains such as financial markets or energy management, time series are heavily influenced by news and social events. A future approach to solving this problem is to use LLM-based Agents that can filter through a large volume of news and analyze the impact of unexpected events on the time series with a **human-like reasoning**. These agents continuously modify their news selection logic using a reflection mechanism to keep the impact of events in sync with the fluctuations of the signals. However, supporting such reasoning requires RAG systems. Since general textual retrievers fail in the face of high noise in time series data and financial metadata, the development of domain-specific retrievers has become a necessity. Architectures such as FinSeer (in the form of the FinSrag framework), using the LLM feedback-based candidate selection mechanism and similarity-driven learning objectives, are able to extract hidden and highly influential patterns from historical news and significantly increase forecast accuracy in volatile environments [45, 141, 155].

11.5.4 Cross-Domain Unification via Language Metadata

The next generation of multi-modal SLMs should be able to cross the boundaries between different domains for example, using the knowledge learned from weather forecasts to predict traffic. The main challenge here is the differences in the inherent properties of the data (such as differences in the number of variables or convergence rate). Studies such as UniTime demonstrate that language can act as a powerful bridge to align different domains into a common semantic space. Future research should move towards unified models that adapt to different sensor data formats using language instructions. This capability allows a single, unified SLM to act as a universal forecaster for different domains without the need for costly retraining [36, 89].

11.5.5 Vision-Language Integration for Complex Feature Extraction

The most innovative and revolutionary research direction is to introduce the machine vision dimension into time series language models. Texts, however rich, are limited in expressing fine-grained temporal dynamics and microsecond fluctuations. By transforming sensor data into information images, the power of basic visual-language models (VLMs) can be harnessed. Frameworks such as Time-VLM produce unique multimodal embeddings by utilizing a three-tier architecture consisting of a visual learner (to understand the visual patterns of the signal), a textual learner (to understand the context), and a memory bank (to understand the retrieval-based learner). One of the most important future agendas is the development of large, multi-dimensional datasets that include time series, textual annotations, and visual representations. Pre-training models on such datasets will open up entirely new horizons in few-shot and zero-shot predictions [156].

11.6 Federated and Privacy-Preserving Model Sharing

The deployment of language models at the network edge and within the IoT signifies a shift toward distributed intelligence architectures in edge environments [157]. Furthermore, efforts to deploy foundation models for time series forecasting have introduced novel technical challenges compared to the fine-tuning of traditional models [110]. However, the practical realization of this vision is hindered by constraints arising from the inability to directly share local data, particularly in domains such as healthcare that are governed by stringent regulations like HIPAA [110, 158]. In this context, federated learning has been introduced as a framework for collaborative model training while keeping the training data decentralized on user devices [158]. Despite this, deploying language or foundation models within a federated ecosystem (especially for time series forecasting applications) is accompanied by challenges such as cross-domain data heterogeneity and the complexities of knowledge sharing, which necessitate the design of highly specialized mechanisms at the architectural and protocol levels [110].

The primary challenge in this convergence lies in the development of FFMs aimed at facilitating knowledge sharing across heterogeneous domains [110]. Unlike conventional approaches that train a dedicated model for each specific forecasting task, recent studies have shifted toward frameworks that enable the utilization of shared knowledge across diverse domains [110, 159]. For instance, the TIME-FFM model, by converting continuous time series into discrete representations, seeks to harness the capacity of language models to extract temporal patterns [110]. Nevertheless, statistical heterogeneity and discrepancies in local data distributions across clients remain major impediments to the convergence of federated models in real-world scenarios [159]. In this regard, PEFT techniques (which update only a small fraction of the model's parameters) have been proposed as a viable strategy to reduce communication overhead and preserve the model's foundational knowledge [110, 159].

Another critical challenge in achieving global model convergence is the severe statistical heterogeneity inherent in time series data. Unlike general image or text data, real-world time series data frequently exhibit highly Non-Independent and Identically

Distributed (Non-IID) characteristics, meaning that each client observes only a partial subset of the system's temporal regimes and behavioral patterns. For example, financial market data can exhibit highly heterogeneous dynamics across different regions or institutions. Empirical results demonstrate that although this heterogeneity leads to a drop in performance compared to centralized training, classical federated algorithms like FedAvg remain stable and significantly outperform isolated local models [160]. Nonetheless, these findings suggest that a single global model does not invariably guarantee optimal performance for all clients. Consequently, a recent study has gravitated toward clustered federated learning, wherein clients are partitioned into distinct clusters prior to training using K-means clustering, and a separate global model is trained for each respective cluster [159]. Additionally, Noseda et al. [160] have explored personalized federated learning as a complementary approach to address data heterogeneity.

Beyond algorithmic challenges, computational and energy constraints within the IoT ecosystem render the direct execution of language models on edge devices highly problematic [161]. The computational overhead of these models, on one hand, and the communication costs associated with parameter exchange, on the other, impede their efficient deployment in distributed environments. To alleviate this systemic strain, recent studies have shifted toward integrating federated learning with split learning (split federated learning). Within this framework, the model is partitioned between clients and the edge server, enabling collaborative training [157, 161]. Methods such as CARD, by dynamically selecting the cut layer based on channel conditions and device resources, have reported significant reductions in both training latency and energy consumption [157]. Furthermore, the utilization of PEFT methods like LoRA, which update only a small subset of the model's parameters, has been proposed as a strategy to minimize communication overhead when tuning language models in federated environments [161]. Nevertheless, synchronizing training amidst device heterogeneity and network instability remains an open challenge within these hybrid architectures [161].

Finally, it must be emphasized that merely withholding raw data in federated learning does not guarantee absolute security. Studies have demonstrated that shared parameters or gradients can lead to the leakage of sensitive information, exposing the system to vulnerabilities such as backdoor injections or model poisoning by malicious clients [158, 161]. The new generation of federated systems has evolved beyond fundamental mechanisms, leaning toward the integration of distributed technologies like blockchain to enhance the traceability and integrity of the training process. By recording the hashes of model updates on immutable ledgers and utilizing systems like IPFS, malicious alterations can be effectively identified [162]. Concurrently, striking an optimal balance between privacy preservation and model utility remains a central challenge; employing mechanisms such as differential privacy requires sophisticated designs that limit information leakage without causing severe performance degradation [161]. Moreover, generating synthetic, privacy-preserving versions of data has been proposed as a complementary approach to reduce reliance on real-world data. Frameworks such as PFDS exemplify this trajectory, enabling the creation and sharing of privatized datasets [163].

11.7 Automated Discovery of Prompt Templates and Adapter Placements

Recent research on deploying language models for time series prediction indicates that the success of these models is heavily influenced by how numerical data is transformed into linguistic representations [164]. Although initial frameworks, such as the one introduced by Xue and Salim [32], relied on fixed and pre-defined textual templates, the dependence on human expertise to design these prompts has created a fundamental limitation in model generalizability. Indeed, the high sensitivity of SLMs to input sentence structure amplifies the necessity of transitioning towards automated discovery systems. In this regard, novel "prompt mining" approaches strive to dynamically execute the prompt generation and refinement process using information-theoretic metrics such as entropy, ensuring the generated content provides maximum informational richness for the model [164]. This automation process not only prevents human biases but also enables the rapid adaptation of the model to the specific characteristics of each data domain [32, 164].

Beyond textual aspects, the challenge of optimal adapter placement and soft prompt tuning within the model's vector space has emerged as a new research frontier. Given the asynchronous nature of many real-world time series, traditional fine-tuning methods may fail to extract latent patterns. To this end, the use of stochastic mechanisms in soft prompt design, such as the approach presented by Gupta et al. [165], has demonstrated that injecting learnable prefixes with a multi-level structure can yield more robust representations compared to conventional methods like QLoRA. This research direction is premised on the assumption that the automated and intelligent placement of adaptive layers within the language model architecture is key to achieving high efficiency while maintaining parameter constraints [165].

However, automating prompts without considering the semantic alignment between numerical and textual modalities can lead to misleading results. Results from extensive empirical analyses indicate that in many cases, adding manual descriptive texts not only fails to aid model interpretability but may also degrade performance due to semantic misalignment [166, 167]. Therefore, the trend of future research is towards using automated statistical prompts that directly transfer the global features of the time series into the model's semantic space. Comprehensive benchmarks such as LTSM-Bundle have proven that when the selection of these features and their integration into prompts are performed automatically based on the nature of the dataset, the model's prediction accuracy improves significantly [168]. Ultimately, establishing an AutoML framework capable of simultaneously optimizing prompt structure, the type of extracted statistical features, and the precise location of adapter insertion remains one of the open and critical questions for the next generation of SLMs in time series analysis [164, 168].

12 Concluding Remarks

The emergence of SLMs represents a paradigm shift in time series forecasting, democratizing access to powerful predictive capabilities while addressing the computational,

privacy, and cost constraints that have limited the practical deployment of larger models. The rapid growth in research activity–with over 65% of reviewed studies published in 2024–2025 alone–underscores the vitality and momentum of this field.

As SLM architectures continue to mature, training methodologies become more efficient, and deployment frameworks more sophisticated, we anticipate that compact language models will become the default choice for a wide range of forecasting applications, particularly those requiring edge deployment, real-time inference, or privacy preservation. The convergence of efficient attention mechanisms, semantic tokenization strategies, retrieval-augmented architectures, and neuro-symbolic reasoning promises to further narrow the performance gap with larger models while maintaining the deployment advantages that make SLMs uniquely suited for practical applications.

This survey has aimed to provide researchers and practitioners with a comprehensive roadmap for navigating this rapidly evolving landscape. By synthesizing the current state of knowledge, identifying critical challenges, and highlighting promising research directions, we hope to accelerate progress toward the next generation of efficient, accurate, and deployable time series forecasting systems powered by SLMs.

Declarations

This research received no external funding.

References

[1] Box, G.E., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: Time Series Analysis: Forecasting and Control. John Wiley & Sons, ??? (2015)

[2] Chatfield, C.: The Analysis of Time Series: Theory and Practice. Springer, ??? (2013)

[3] Kim, J., Kim, H., Kim, H., Lee, D., Yoon, S.: A comprehensive survey of deep learning for time series forecasting: architectural diversity and open challenges. Artificial Intelligence Review **58**(7), 216 (2025)

[4] Mariano-Hernández, D., Hernández-Callejo, L., García, F.S., Duque-Perez, O., Zorita-Lamadrid, A.L.: A review of energy consumption forecasting in smart buildings: Methods, input variables, forecasting horizon and metrics. Applied Sciences **10**(23) (2020) https://doi.org/10.3390/app10238323

[5] Asiri, M.M., Aldehim, G., Alotaibi, F.A., Alnfiai, M.M., Assiri, M., Mahmud, A.: Short-term load forecasting in smart grids using hybrid deep learning. IEEE Access **12**, 23504–23513 (2024) https://doi.org/10.1109/ACCESS.2024.3358182

[6] Rajkomar, A., Dean, J., Kohane, I.: Machine learning in medicine. New England Journal of Medicine **380**, 1347–1358 (2019)

[7] Kwak, G.H., Hui, P.: Deephealth: Deep learning for health informatics. CoRR **abs/1909.00384** (2019) 1909.00384

[8] Shi, X., Xue, S., Wang, K., Zhou, F., Zhang, J., Zhou, J., Tan, C., Mei, H.: Language models can improve event prediction by few-shot abductive reasoning. In: Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S. (eds.) Advances in Neural Information Processing Systems, vol. 36, pp. 29532–29557. Curran Associates, Inc., ??? (2023). https://proceedings.neurips.cc/paper_files/paper/2023/file/5e5fd18f863cbe6d8ae392a93fd271c9-Paper-Conference.pdf

[9] Amiri, Z., Heidari, A., Navimipour, N.J., Esmaeilpour, M., Yazdani, Y.: The deep learning applications in iot-based bio-and medical informatics: a systematic literature review. Neural Computing and Applications **36**(11), 5757–5797 (2024)

[10] Shahid, F., Zameer, A., Muneeb, M.: Predictions for covid-19 with deep learning models of lstm, gru and bi-lstm. Chaos, Solitons & Fractals **140**, 110212 (2020) https://doi.org/10.1016/j.chaos.2020.110212

[11] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., *et al.*: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020)

[12] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G.: LLaMA: Open and Efficient Foundation Language Models (2023). https://arxiv.org/abs/2302.13971

[13] Jin, M., Wang, S., Ma, L., Chu, Z., Zhang, J., Shi, X., Chen, P.-Y., Liang, Y., Li, Y.-F., Pan, S., *et al.*: Time-llm: Time series forecasting by reprogramming large language models. In: International Conference on Learning Representations, vol. 2024, pp. 23857–23880 (2024)

[14] 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)

[15] Liu, X., Liu, J., Woo, G., Aksu, T., Liang, Y., Zimmermann, R., Liu, C., Savarese, S., Xiong, C., Sahoo, D.: Moirai-moe: Empowering time series foundation models with sparse mixture of experts. arXiv preprint arXiv:2410.10469 (2024)

[16] Tang, H., Zhang, C., Jin, M., Yu, Q., Wang, Z., Jin, X., Zhang, Y., Du, M.: Time series forecasting with llms: Understanding and enhancing model capabilities. ACM SIGKDD Explorations Newsletter **26**(2), 109–118 (2025)

[17] Zhang, X., Chowdhury, R.R., Gupta, R.K., Shang, J.: Large language models for time series: A survey. arXiv preprint arXiv:2402.01801 (2024)

[18] Zhang, P., Zeng, G., Wang, T., Lu, W.: TinyLlama: An Open-Source Small Language Model (2024). https://arxiv.org/abs/2401.02385

[19] Zhou, B., Hu, Y., Weng, X., Jia, J., Luo, J., Liu, X., Wu, J., Huang, L.: Tinyllava: A framework of small-scale large multimodal models. arXiv preprint arXiv:2402.14289 (2024)

[20] Pham, T.M., Nguyen, P.T., Yoon, S., Lai, V.D., Dernoncourt, F., Bui, T.: Slimlm: An efficient small language model for on-device document assistance. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pp. 436–447 (2025)

[21] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)

[22] Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D.d.L., Hendricks, L.A., Welbl, J., Clark, A., et al.: Training compute-optimal large language models. arXiv preprint arXiv:2203.15556 (2022)

[23] Nguyen, C.V., Shen, X., Aponte, R., Xia, Y., Basu, S., Hu, Z., Chen, J., Parmar, M., Kunapuli, S., Barrow, J., Wu, J., Singh, A., Wang, Y., Gu, J., Dernoncourt, F., Ahmed, N.K., Lipka, N., Zhang, R., Chen, X., Yu, T., Kim, S., Deilamsalehy, H., Park, N., Rimer, M., Zhang, Z., Yang, H., Rossi, R.A., Nguyen, T.H.: A Survey of Small Language Models (2024). https://arxiv.org/pdf/2410.20011

[24] Wang, F., Zhang, Z., Zhang, X., Wu, Z., Mo, T., Lu, Q., Wang, W., Li, R., Xu, J., Tang, X., et al.: A comprehensive survey of small language models in the era of large language models: Techniques, enhancements, applications, collaboration with llms, and trustworthiness. ACM Transactions on Intelligent Systems and Technology (2024)

[25] Kandala, S.V., Medaranga, P., Varshney, A.: Tinyllm: A framework for training and deploying language models at the edge computers. arXiv preprint arXiv:2412.15304 (2024)

[26] Popov, R.O., Karpenko, N.V., Gerasimov, V.V.: Overview of small language models in practice. In: CS&SE@ SW, pp. 164–182 (2024)

[27] Ahn, K., Cheng, X., Song, M., Yun, C., Jadbabaie, A., Sra, S.: Linear attention is (maybe) all you need (to understand transformer optimization). In: International Conference on Learning Representations, vol. 2024, pp. 16193–16205 (2024)

[28] Wang, S., Li, B.Z., Khabsa, M., Fang, H., Ma, H.: Linformer: Self-attention with linear complexity. arXiv preprint arXiv:2006.04768 (2020)

[29] Zhang, M., Bhatia, K., Kumbong, H., Ré, C.: Hedgehog: Expressive linear

attention via softmax mimicry. arXiv preprint (2024)

[30] Kitaev, N., Kaiser, Ł., Levskaya, A.: Reformer: The efficient transformer. arXiv preprint arXiv:2001.04451 (2020)

[31] Zhu, C., Ping, W., Xiao, C., Shoeybi, M., Goldstein, T., Anandkumar, A., Catanzaro, B.: Long-short transformer: Efficient transformers for language and vision. Advances in neural information processing systems **34**, 17723–17736 (2021)

[32] Xue, H., Salim, F.D.: Promptcast: A new prompt-based learning paradigm for time series forecasting. IEEE Transactions on Knowledge and Data Engineering **36**(11), 6851–6864 (2023)

[33] Yang, J.: Tokon: Tokenization-optimized normalization for time series analysis with a large language model. arXiv preprint arXiv:2502.05701 (2025)

[34] Ansari, A.F., Stella, L., Turkmen, 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). Expert Certification

[35] Wang, C., Qi, Q., Wang, J., Sun, H., Zhuang, Z., Wu, J., Zhang, L., Liao, J.: Chattime: A unified multimodal time series foundation model bridging numerical and textual data. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 39, pp. 12694–12702 (2025)

[36] Ghasemloo, M., Moradi, A.: Informed forecasting: Leveraging auxiliary knowledge to boost llm performance on time series forecasting. arXiv preprint arXiv:2505.10213 (2025)

[37] Chatzigeorgakidis, G., Lentzos, K., Skoutas, D.: Multicast: Zero-shot multivariate time series forecasting using llms. In: 2024 IEEE 40th International Conference on Data Engineering Workshops (ICDEW), pp. 119–127 (2024). IEEE

[38] Parker, F., Chan, N., Zhang, C., Ghobadi, K.: Augmenting llms for general time series understanding and prediction. arXiv preprint arXiv:2510.01111 (2025)

[39] Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., *et al.*: Lora: Low-rank adaptation of large language models. ICLR **1**(2), 3 (2022)

[40] Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S.: Parameter-efficient transfer learning for nlp. In: International Conference on Machine Learning, pp. 2790–2799 (2019). PMLR

[41] Li, X.L., Liang, P.: Prefix-tuning: Optimizing continuous prompts for generation. arXiv preprint arXiv:2101.00190 (2021)

[42] Freitas, J.D., Ponte, C., Bomfim, R., Caminha, C.: Evaluating window size effects on univariate time series forecasting with machine learning. Journal of Information and Data Management **16**(1) (2025) https://doi.org/10.5753/jidm.2025.4668

[43] Yang, S., Wang, D., Zheng, H., Jin, R.: Timerag: Boosting llm time series forecasting via retrieval-augmented generation. In: ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1–5 (2025). IEEE

[44] Ning, K., Pan, Z., Liu, Y., Jiang, Y., Zhang, J.Y., Rasul, K., Schneider, A., Ma, L., Nevmyvaka, Y., Song, D.: Ts-rag: Retrieval-augmented generation based time series foundation models are stronger zero-shot forecaster. arXiv preprint arXiv:2503.07649 (2025)

[45] Xiao, M., Chen, Z., Qian, L., Jiang, Z., He, Y., Xu, Y., Jiang, Y., Li, D., Weng, R.-L., Huang, J., Peng, M., Ananiadou, S., Nie, J.-Y., Xie, Q.: Retrieval-augmented large language models for financial time series forecasting. arXiv preprint arXiv:2502.05878 (2025)

[46] Das, A., Faw, M., Sen, R., Zhou, Y.: In-context fine-tuning for time-series foundation models. arXiv preprint arXiv:2410.24087 (2024)

[47] Ravuru, C., Sakhinana, S.S.: Agentic retrieval-augmented generation for time series analysis. arXiv preprint arXiv:2408.14484 (2024)

[48] Wang, Q.: A hybrid transformer-arima model for forecasting global supply chain disruptions using multimodal data. International Journal of Advanced Computer Science & Applications **16**(1) (2025)

[49] Fan, H., Li, B., Weng, Y., *et al.*: Small but mighty: enhancing time series forecasting with lightweight llms. The Journal of Supercomputing **81**, 985 (2025) https://doi.org/10.1007/s11227-025-07491-5

[50] He, Q., Yu, J., Wang, W.: Large language model-enhanced symbolic reasoning for knowledge base completion. IEEE Transactions on Audio, Speech and Language Processing (2026)

[51] Fang, M., Deng, S., Zhang, Y., Shi, Z., Chen, L., Pechenizkiy, M., Wang, J.: Large language models are neurosymbolic reasoners. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, pp. 17985–17993 (2024)

[52] Liu, J., Song, X., Zhang, D., Thomale, J., He, D., Hong, L.: A hybrid framework for subject analysis: Integrating embedding-based regression models with large

language models. Proceedings of the Association for Information Science and Technology **62**(1), 445–457 (2025)

[53] Kalifa, D., Singer, U., Guy, I., Rosin, G.D., Radinsky, K.: Leveraging world events to predict e-commerce consumer demand under anomaly. In: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, pp. 430–438 (2022)

[54] Long, X., Bui, Q., Oktavian, G., Schmidt, D.F., Bergmeir, C., Godahewa, R., Lee, S.P., Zhao, K., Condylis, P.: Scalable probabilistic forecasting in retail with gradient boosted trees: A practitioner's approach. International Journal of Production Economics **279**, 109449 (2025)

[55] Chan, N., Parker, F., Bennett, W., Wu, T., Jia, M.Y., Fackler, J., Ghobadi, K.: Medtsllm: Leveraging llms for multimodal medical time series analysis. arXiv preprint arXiv:2408.07773 (2024)

[56] Wolff, M.L., Yang, S., Torkkola, K., Mahoney, M.W.: Using pre-trained llms for multivariate time series forecasting. arXiv preprint arXiv:2501.06386 (2025)

[57] Palma, G., Cecchi, G., Rizzo, A.: Large language models for predictive maintenance in the leather tanning industry: Multimodal anomaly detection in compressors. Electronics **14**(10), 2061 (2025)

[58] Hidayat, F., Nasution, A.H., Ambia, F., Putra, D.F., et al.: Leveraging large language models for discrepancy value prediction in custody transfer systems: A comparative analysis of probabilistic and point forecasting approaches. IEEE Access (2025)

[59] Shakhadri, S.A.G., KR, D.K., Aralimatti, R.: Shakti: A 2.5 billion parameter small language model optimized for edge ai and low-resource environments. In: IFIP International Conference on Artificial Intelligence Applications and Innovations, pp. 434–447 (2025). Springer

[60] She, J., Zheng, W., Liu, Z., Wang, H., Xing, E., Yao, H., Ho, Q.: Token level routing inference system for edge devices. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pp. 159–166 (2025)

[61] Ling, T., Singh, V., Qian, C., Biessmann, F., Schiele, G.: Automated energy-aware time-series model deployment on embedded fpgas for resilient combined sewer overflow management. arXiv preprint arXiv:2508.13905 (2025)

[62] Ling, T., Qian, C., Schiele, G.: Resource-aware mixed-precision quantization for enhancing deployability of transformers for time-series forecasting on embedded fpgas. arXiv preprint arXiv:2410.03294 (2024). Preprint, arXiv:2410.03294v3

[63] Ordóñez, S.A.C., Samanta, J., Suárez-Cetrulo, A.L., Carbajo, R.S.: Adaptive machine learning for resource-constrained environments. In: International Workshop on Discovering Drift Phenomena in Evolving Landscapes, pp. 3–19 (2024). Springer

[64] Qiu, X., Hu, J., Zhou, L., Wu, X., Du, J., Zhang, B., Guo, C., Zhou, A., Jensen, C.S., Sheng, Z., et al.: Tfb: Towards comprehensive and fair benchmarking of time series forecasting methods. arXiv preprint arXiv:2403.20150 (2024)

[65] Hewamalage, H., Ackermann, K., Bergmeir, C.: Forecast evaluation for data scientists: common pitfalls and best practices. Data Mining and Knowledge Discovery **37**(2), 788–832 (2023)

[66] Zhang, Y., Chen, W., Zhu, Z., Qin, D., Sun, L., Wang, X., Wen, Q., Zhang, Z., Wang, L., Jin, R.: Addressing concept shift in online time series forecasting: Detect-then-adapt. arXiv preprint arXiv:2403.14949 (2024)

[67] Scaria, N., Kennedy, S.J.J., Subramani, D.: Can small language models learn, unlearn, and retain noise patterns? arXiv preprint arXiv:2407.00996 (2024)

[68] Gupta, S.D.: Evaluating time series foundation models on noisy periodic time series. arXiv preprint arXiv:2501.00889 (2025)

[69] Karim, F., Majumdar, S., Darabi, H.: Adversarial attacks on time series. IEEE transactions on pattern analysis and machine intelligence **43**(10), 3309–3320 (2020)

[70] Liu, F., Jiang, S., Miranda-Moreno, L., Choi, S., Sun, L.: Adversarial vulnerabilities in large language models for time series forecasting. In: International Conference on Artificial Intelligence and Statistics, pp. 4672–4680 (2025). PMLR

[71] Kim, K., Shin, J., Kim, J.: Personalized language models via privacy-preserving evolutionary model merging. In: Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pp. 34441–34456 (2025)

[72] Lu, Z., Li, X., Cai, D., Yi, R., Liu, F., Zhang, X., Lane, N.D., Xu, M.: Small Language Models: Survey, Measurements, and Insights (2025). https://arxiv.org/abs/2409.15790

[73] Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)

[74] Sanh, V., Debut, L., Chaumond, J., Wolf, T.: Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108 (2019)

[75] Frantar, E., Alistarh, D.: Sparsegpt: Massive language models can be accurately pruned in one-shot. In: International Conference on Machine Learning, pp. 10323–10337 (2023). PMLR

[76] Frantar, E., Ashkboos, S., Hoefler, T., Alistarh, D.: Gptq: Accurate post-training quantization for generative pre-trained transformers. In: The Eleventh International Conference on Learning Representations (2023). OpenReview

[77] Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., Han, S.: Smoothquant: Accurate and efficient post-training quantization for large language models. In: International Conference on Machine Learning, pp. 38087–38099 (2023). PMLR

[78] Ma, S., Wang, H., Ma, L., Wang, L., Wang, W., Huang, S., Dong, L., Wang, R., Xue, J., Wei, F.: The era of 1-bit llms: All large language models are in 1.58 bits. arXiv preprint arXiv:2402.17764 (2024)

[79] Lester, B., Al-Rfou, R., Constant, N.: The power of scale for parameter-efficient prompt tuning. arXiv preprint arXiv:2104.08691 (2021)

[80] Gu, A., Dao, T.: Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752 (2023)

[81] Tay, Y., Dehghani, M., Bahri, D., Metzler, D.: Efficient transformers: A survey. ACM Comput. Surv. **55**(6) (2022) https://doi.org/10.1145/3530811

[82] Nie, Y., Nguyen, N.H., Sinthong, P., Kalagnanam, J.: A time series is worth 64 words: Long-term forecasting with transformers. In: Proceedings of the International Conference on Learning Representations (ICLR) (2023). ICLR

[83] Chiu, C.-Y., Hong, D.-Y., Liu, P., Wu, J.-J.: Effective compression of language models by combining pruning and knowledge distillation. In: 2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 429–438 (2024). IEEE

[84] Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., Liu, Q.: Tinybert: Distilling bert for natural language understanding. In: Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 4163–4174 (2020)

[85] Sun, Y., Li, Z., Zhang, Y., Pan, T., Dong, B., Guo, Y., Wang, J.: Efficient attention mechanisms for large language models: A survey. Journal of Efficient AI Systems **1**(1), 1–22 (2025)

[86] Lou, C., Jia, Z., Zheng, Z., Tu, K.: Sparser is faster and less is more: Efficient sparse attention for long-range transformers. Preprint / arXiv (2024)

[87] Xu, M., Xiong, Y., Chen, H., Li, X., Xia, W., Tu, Z., Soatto, S.: Long short-term transformer for online action detection. Advances in Neural Information

Processing Systems **34**, 1086–1099 (2021)

[88] Hou, Z.: Neural-symbolic reasoning: Towards the integration of logical reasoning with large language models. Authorea Preprints (2025)

[89] Liu, X., Hu, J., Li, Y., Diao, S., Liang, Y., Hooi, B., Zimmermann, R.: Unitime: A language-empowered unified model for cross-domain time series forecasting. In: Proceedings of the ACM Web Conference 2024, pp. 4095–4106 (2024)

[90] Jia, F., Wang, K., Zheng, Y., Cao, D., Liu, Y.: Gpt4mts: Prompt-based large language model for multimodal time-series forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, pp. 23343–23351 (2024)

[91] Kim, K., Tsai, H., Sen, R., Das, A., Zhou, Z., Tanpure, A., Luo, M., Yu, R.: Multi-modal forecaster: Jointly predicting time series and textual data. arXiv preprint arXiv:2411.06735 (2024)

[92] Cui, Z., Wang, B., Liu, Q., Wang, Y., Zhou, Z., Liang, Y., Wang, Y.: Augur: Modeling covariate causal associations in time series via large language models. arXiv preprint arXiv:2510.07858 (2025)

[93] Wang, Z., Lan, L., Li, Y.: Time-prompt: Integrated heterogeneous prompts for unlocking llms in time series forecasting. arXiv preprint arXiv:2506.17631 (2025)

[94] Cao, D., Jia, F., Arık, S.Ö., Pfister, T., Zheng, Y., Ye, W., Liu, Y.: Tempo: Prompt-based generative pre-trained transformer for time series forecasting. In: The Twelfth International Conference on Learning Representations (2024)

[95] 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 (2024). PMLR

[96] Zhou, T., Niu, P., Wang, X., Sun, L., Jin, R.: One fits all: Power general time series analysis by pretrained lm. In: Advances in Neural Information Processing Systems, vol. 36 (2023)

[97] Song, W., Fang, Y., Gu, X., Zhang, W., Liu, Z., Cheng, Y., Di Mauro, M.: Streamts: A streamline solution towards zero-shot time series forecasting with large language models. Electronics **14**(20), 4088 (2025)

[98] Shi, F., Yin, X., Wang, K., Tu, W., Sun, Q., Ning, H.: Large language models for time series analysis: Techniques, applications, and challenges. arXiv preprint arXiv:2506.11040 (2025)

[99] Das, A., Kong, W., Sen, R., Zhou, Y.: A decoder-only foundation model for time-series forecasting. arXiv preprint arXiv:2310.10688 (2023)

[100] Liu, H., Yang, C., Xiaoxing, Z., Zhu, X.: Semantic-enhanced time-series forecasting via large language models. arXiv preprint arXiv:2508.07697 (2025)

[101] Liu, C., Miao, H., Xu, Q., Zhou, S., Long, C., Zhao, Y., Li, Z., Zhao, R.: Efficient multivariate time series forecasting via calibrated language models with privileged knowledge distillation. In: 2025 IEEE 41st International Conference on Data Engineering (ICDE), pp. 3165–3178 (2025). IEEE

[102] Su, C., Tian, Y., Liu, Q., Zhang, J., Song, Y.: Fusing large language models with temporal transformers for time series forecasting. arXiv preprint arXiv:2507.10098 (2025)

[103] 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)

[104] Sun, C., Li, H., Li, Y., Hong, S.: Test: Text prototype aligned embedding to activate llm's ability for time series. arXiv preprint arXiv:2308.08241 (2023)

[105] Zhao, T., Chen, X., Sun, M.: Enhancing time series forecasting via multi-level text alignment with llms. arXiv preprint arXiv:2504.07360 (2025)

[106] Liu, C., Xu, Q., Miao, H., Yang, S., Zhang, L., Long, C., Li, Z., Zhao, R.: Timecma: Towards llm-empowered multivariate time series forecasting via cross-modality alignment. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 39, pp. 18780–18788 (2025)

[107] Zhou, S., Schöner, H., Lyu, H., Fouché, E., Wang, S.: Balm-tsf: Balanced multimodal alignment for llm-based time series forecasting. In: Proceedings of the 34th ACM International Conference on Information and Knowledge Management, pp. 4498–4508 (2025)

[108] Pan, Z., Jiang, Y., Garg, S., Schneider, A., Nevmyvaka, Y., Song, D.: sip-llm: Semantic space informed prompt learning with llm for time series forecasting. In: Forty-first International Conference on Machine Learning (2024)

[109] Lee, S., Choi, J., Lee, S., Song, M., Jang, B.-G., Yu, H., Han, S.C.: Map4ts: A multi-aspect prompting framework for time-series forecasting with large language models. arXiv preprint arXiv:2510.23090 (2025)

[110] Liu, Q., Liu, X., Liu, C., Wen, Q., Liang, Y.: Time-ffm: Towards lm-empowered federated foundation model for time series forecasting. Advances in Neural Information Processing Systems **37**, 94512–94538 (2024)

[111] Liu, Y., Qin, G., Huang, X., Wang, J., Long, M.: Autotimes: Autoregressive time series forecasters via large language models. Advances in Neural Information Processing Systems **37**, 122154–122184 (2024)

[112] Chai, T., Draxler, R.R.: Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature. Geoscientific model development **7**(3), 1247–1250 (2014)

[113] Mehmood, H., Kostakos, P., Cortes, M., Anagnostopoulos, T., Pirttikangas, S., Gilman, E.: Concept drift adaptation techniques in distributed environment for real-world data streams. Smart Cities **4**(1), 349–371 (2021)

[114] Pawar, N., Hollweg, G.V., Hussain, A., Su, W., Bui, V.-H.: Enhancing forecasting accuracy in dynamic environments via pelt-driven drift detection and model adaptation. arXiv preprint arXiv:2506.14133 (2025)

[115] Liu, Z., Godahewa, R., Bandara, K., Bergmeir, C.: Handling concept drift in global time series forecasting. In: Forecasting with Artificial Intelligence: Theory and Applications, pp. 163–189. Springer, ??? (2023)

[116] Pham, N.T., Kieu, T., Nguyen, D.M., Xuan, S.H., Duong-Trung, N., Le-Phuoc, D.: Slm-bench: A comprehensive benchmark of small language models on environmental impacts. In: Findings of the Association for Computational Linguistics: EMNLP 2025, pp. 21369–21392 (2025)

[117] Ekambaram, V., Jati, A., Dayama, P., Mukherjee, S., Nguyen, N., Gifford, W.M., Reddy, C., Kalagnanam, J.: Tiny time mixers (ttms): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series. Advances in Neural Information Processing Systems **37**, 74147–74181 (2024)

[118] Singh, C., Inala, J.P., Galley, M., Caruana, R., Gao, J.: Rethinking interpretability in the era of large language models. arXiv preprint arXiv:2402.01761 (2024)

[119] Lim, B., Arık, S.Ö., Loeff, N., Pfister, T.: Temporal fusion transformers for interpretable multi-horizon time series forecasting. International journal of forecasting **37**(4), 1748–1764 (2021)

[120] Delibasoglu, I., Heintz, F.: Learning temporal saliency for time series forecasting with cross-scale attention. arXiv preprint arXiv:2509.22839 (2025)

[121] Roberts, I., Schulz, A., Hermes, L., Hammer, B.: Targeted visualization of the backbone of encoder llms. arXiv preprint arXiv:2403.18872 (2024)

[122] Chang, C., Hwang, J., Shi, Y., Wang, H., Peng, W.-C., Chen, T.-F., Wang, W.: Time-imm: A dataset and benchmark for irregular multimodal multivariate time series. arXiv preprint arXiv:2506.10412 (2025)

[123] Lu, J., Sun, Y., Yang, S.: In-context time series predictor. arXiv preprint arXiv:2405.14982 (2024)

[124] 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)

[125] Wu, H., Xu, J., Wang, J., Long, M.: Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. Advances in neural information processing systems **34**, 22419–22430 (2021)

[126] Li, M., Liu, R., Shi, G., Han, M., Li, C., Yao, L., Chang, X., Chen, L.: Mitigating data redundancy to revitalize transformer-based long-term time series forecasting system. ACM Transactions on Intelligent Systems and Technology **17**(3), 1–21 (2026)

[127] Yin, T., Wang, J., Ma, Y., Wang, H., Wang, C., Zhao, Y., Liu, M., Shen, W.: Apollo-forecast: Overcoming aliasing and inference speed challenges in language models for time series forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 39, pp. 22173–22181 (2025)

[128] Eldele, E., Ragab, M., Chen, Z., Wu, M., Li, X.: Tslanet: Rethinking transformers for time series representation learning. arXiv preprint arXiv:2404.08472 (2024)

[129] Liu, Y., Wu, H., Wang, J., Long, M.: Non-stationary transformers: Exploring the stationarity in time series forecasting. Advances in neural information processing systems **35**, 9881–9893 (2022)

[130] López-Gonzales, J.L., Gómez Lamus, A.M., Torres, R., Canas Rodrigues, P., Salas, R.: Self-organizing topological multilayer perceptron: A hybrid method to improve the forecasting of extreme pollution values. Stats **6**(4), 1241–1259 (2023)

[131] Sakhinana, S.S., Sannidhi, G., Ravuru, C., Runkana, V.: Advancing enterprise spatio-temporal forecasting applications: Data mining meets instruction tuning of language models for multi-modal time series analysis in low-resource settings. arXiv preprint arXiv:2408.13622 (2024)

[132] Teng, J., Qin, R., Xu, C.: Time series forecasting in marine environment monitoring sensor networks based on edge computing. IEEE Access (2025)

[133] Xie, W., Cao, F.: Swift: Mapping sub-series with wavelet decomposition improves time series forecasting. arXiv preprint arXiv:2501.16178 (2025)

[134] Xu, Z., Zeng, A., Xu, Q.: Fits: Modeling time series with $10k$ parameters. arXiv preprint arXiv:2307.03756 (2023)

[135] Chen, M., Hui, B., Cui, Z., Yang, J., Liu, D., Sun, J., Lin, J., Liu, Z.: Parallel

scaling law for language models. arXiv preprint arXiv:2505.10475 (2025)

[136] Yang, Y., Wen, L., Li, L.: Explainable ai for time series prediction in economic mental health analysis. Frontiers in Medicine **12**, 1591793 (2025) https://doi.org/10.3389/fmed.2025.1591793

[137] Chu, Z., Wang, Z., Zhang, W.: Fairness in large language models: A taxonomic survey. ACM SIGKDD explorations newsletter **26**(1), 34–48 (2024)

[138] Srivastava, G., Cao, S., Wang, X.: Thinkslm: Towards reasoning in small language models. In: Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pp. 32600–32650 (2025)

[139] Micheal, L.: Security and robustness challenges of small language models in autonomous agent networks (2025). Available at ResearchGate

[140] Lee, S.A., Tanade, C., Zhou, H., Lee, J., Thukral, M., Lu, B., Desai, S.A.: Towards on-device foundation models for raw wearable signals. In: NeurIPS 2025 Workshop on Learning from Time Series for Health (2025)

[141] Qiu, J., Han, W., Zhu, J., Xu, M., Rosenberg, M., Liu, E., Weber, D., Zhao, D.: Transfer knowledge from natural language to electrocardiography: Can we detect cardiovascular disease through language models? In: Findings of the Association for Computational Linguistics: EACL 2023, pp. 442–453 (2023)

[142] Chung, H., Kim, J., Kwon, J.-M., Jeon, K.-H., Lee, M.S., Choi, E.: Text-to-ecg: 12-lead electrocardiogram synthesis conditioned on clinical text reports. In: ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1–5 (2023). IEEE

[143] Srinivas, S.S., Ravuru, C., Sannidhi, G., Runkana, V.: Reprogramming foundational large language models (llms) for enterprise adoption for spatio-temporal forecasting applications: Unveiling a new era in copilot-guided cross-modal time series representation learning. arXiv preprint arXiv:2408.14387 (2024)

[144] Tiomoko, M., Paolo, G., Zhang, Y., Keli, Z., Cherkaoui, H., Meng, Y., Ali, H.T.: Human-in-the-loop adaptive optimization for improved time series forecasting. arXiv preprint arXiv:2505.15354 (2025)

[145] Jiang, Y., Yu, W., Lee, G., Song, D., Shin, K., Cheng, W., Liu, Y., Chen, H.: Timexl: Explainable multi-modal time series prediction with llm-in-the-loop. In: Advances in Neural Information Processing Systems (NeurIPS) (2025)

[146] Wu, X., Xiao, L., Sun, Y., Zhang, J., Ma, T., He, L.: A survey of human-in-the-loop for machine learning. Future Generation Computer Systems **135**, 364–381 (2022)

[147] Chen, L., Zaharia, M., Zou, J.: Frugalgpt: How to use large language models while reducing cost and improving performance. arXiv preprint arXiv:2305.05176 (2023)

[148] Lee, T.L., Toner, W., Singh, R., Joosen, A.: Lightweight online adaption for time series foundation model forecasts. arXiv preprint arXiv:2502.12920 (2025)

[149] Besnard, Q., Ragot, N.: Continual learning for time series forecasting: A first survey. Engineering Proceedings **68**(1), 49 (2024)

[150] Ao, S.-I., Fayek, H.: Continual deep learning for time series modeling. Sensors **23**(16), 7167 (2023)

[151] Behrouz, A., Zhong, P., Mirrokni, V.: Titans: Learning to memorize at test time. Advances in Neural Information Processing Systems **38**, 113506–113543 (2026)

[152] Kim, T., Kim, J., Tae, Y., Park, C., Choi, J.-H., Choo, J.: Reversible instance normalization for accurate time-series forecasting against distribution shift. In: International Conference on Learning Representations (ICLR) 2022, pp. 1–25 (2022)

[153] Wu, W., Zhang, G., Tan, Z., Wang, Y., Qi, H.: Dual-forecaster: A multimodal time series model integrating descriptive and predictive texts. arXiv preprint arXiv:2505.01135 (2025)

[154] Yao, Y., Li, J., Dai, X., Zhang, M., Gong, X., Wang, F.-Y., Lv, Y.: Context-aware probabilistic modeling with llm for multimodal time series forecasting. arXiv preprint arXiv:2505.10774 (2025)

[155] Wang, X., Feng, M., Qiu, J., Gu, J., Zhao, J.: From news to forecast: Integrating event analysis in llm-based time series forecasting with reflection. Advances in Neural Information Processing Systems **37**, 58118–58153 (2024)

[156] Zhong, S., Ruan, W., Jin, M., Li, H., Wen, Q., Liang, Y.: Time-vlm: Exploring multimodal vision-language models for augmented time series forecasting. arXiv preprint arXiv:2502.04395 (2025)

[157] Li, Z., Wu, S., Li, L., Zhang, S.: Energy-efficient split learning for fine-tuning large language models in edge networks. IEEE Networking Letters **7**(3), 176–180 (2025)

[158] Kairouz, P., McMahan, H.B.: Advances and open problems in federated learning. Foundations and trends in machine learning **14**(1-2), 1–210 (2021)

[159] Abdel-Sater, R., Ben Hamza, A.: A federated large language model for long-term time series forecasting. In: ECAI 2024: 27th European Conference on Artificial Intelligence, 19–24 October 2024, Santiago de Compostela, Spain–Including 13th

Conference on Prestigious Applications of Intelligent Systems (PAIS 2024), pp. 2452–2459 (2024). SAGE Publications Pvt. Ltd 1 Oliver's Yard, 55 City Road, London, EC1Y 1SP

[160] Noseda, M., De Luca, A., Von Briel, L., Lacour, N.: Federated learning for financial forecasting. arXiv preprint arXiv:2509.16393 (2025)

[161] Yang, H., Liu, H., Yuan, X., Wu, K., Ni, W., Zhang, J.A., Liu, R.P.: Synergizing intelligence and privacy: A review of integrating internet of things, large language models, and federated learning in advanced networked systems. Applied Sciences **15**(12), 6587 (2025)

[162] Chen, J., Xue, J., Wang, Y., Huang, L., Baker, T., Zhou, Z.: Privacy-preserving and traceable federated learning for data sharing in industrial iot applications. Expert Systems with Applications **213**, 119036 (2023)

[163] Fioretto, F., Van Hentenryck, P.: Privacy-preserving federated data sharing. In: AAMAS, pp. 638–646 (2019)

[164] Xue, H., Tang, T., Payani, A., Salim, F.D.: Prompt mining for language-based human mobility forecasting. arXiv preprint arXiv:2403.03544 (2024)

[165] Gupta, S., Durand, T., Taylor, G., et al.: Last stop for modeling asynchronous time series. arXiv preprint arXiv:2502.01922 (2025)

[166] Sun, Z., Qian, H., Tsang, I.: Exploring the effectiveness and interpretability of texts in llm-based time series models. arXiv preprint arXiv:2504.08808 (2025)

[167] Zhang, X., Feng, S., Li, X.: From text to time? rethinking the effectiveness of the large language model for time series forecasting. arXiv preprint arXiv:2504.08818 (2025)

[168] Chuang, Y.-N., Li, S., Yuan, J., Wang, G., Lai, K.-H., Sui, S., Yu, L., Ding, S., Chang, C.-Y., Costilla Reyes, A., *et al.*: Ltsm-bundle: A toolbox and benchmark on large language models for time series forecasting. ACM SIGKDD Explorations Newsletter **27**(2), 43–61 (2025)