VQ-LLMTS: VECTOR QUANTIZATION-BASED LARGE LANGUAGE MODELS FOR TIME SERIES FORECASTING AND UNDERSTANDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) excel at reasoning but struggle with time-series data due to the modality gap between discrete text tokens and continuous numerical signals. In this paper, we propose VQ-LLMTS, a multimodal framework that addresses this gap by discretizing continuous time series into a specialized temporal vocabulary. Unlike many alignment methods that directly bridge text and time-series embeddings, we employ Residual Vector Quantization (RVQ) to decompose complex temporal patterns into a hierarchical, discrete latent space. This expands the LLM's vocabulary with learned temporal tokens, enabling unified cross-modal reasoning while preserving the original backbone and single-head autoregressive architecture. To capture diverse temporal patterns, we develop a generalized hybrid data generator for RVQ codebook training. We then align the temporal vocabulary with the language model using synthetic examples for general capabilities and time-series understanding, together with real-world univariate time series. VQ-LLMTS achieves strong time-series understanding performance and competitive short-horizon forecasting performance using qualitative descriptions derived from the target windows. On the CiK benchmark, VQ-LLMTS also achieves the lowest context-conditioned RCRPS among the compared methods. These results support the effectiveness of an explicit temporal vocabulary for LLM-based time-series understanding and forecasting.

1 INTRODUCTION

Time series analysis is pivotal across domains such as energy (Wolde-Rufael, 2006), healthcare (Morid et al., 2023), traffic (Fang et al., 2022; Li et al., 2015), retail (Qi et al., 2023), and finance (Sezer et al., 2020; Xu & Cohen, 2018). Its methodology has evolved from classical deep learning (Qiu et al., 2026) and specialized foundation models (Ansari et al., 2024; Liu et al., 2024c; Das et al., 2024; Woo et al., 2024; Rasul et al., 2023) to LLM-based multimodal frameworks that integrate textual context to emulate human expert reasoning (Liu et al., 2025a;b). However, a fundamental challenge persists: the inherent modality gap between the continuous nature of time-series data and the discrete token vocabularies of LLMs hinders seamless fusion of numerical and semantic knowledge (Merrill et al., 2024). As discussed in Section 2, prompt-based and continuous-embedding methods still face challenges in preserving numerical precision or aligning time-series representations with language tokens. To resolve this tension, we propose discretizing continuous time-series data into a specialized temporal vocabulary, an approach inspired by successful vector-quantization paradigms in image processing (Razavi et al., 2019; Ramesh et al., 2021) and audio processing (Borsos et al., 2023). By expanding the LLM's lexical space with these discrete temporal tokens, we enable the model to perform unified cross-modal reasoning, effectively aligning high-level linguistic semantics with numerical patterns for superior forecasting and understanding.

Specifically, we propose a two-stage framework, VQ-LLMTS (Vector Quantization-based Large Language Models for Time Series Forecasting and Understanding). The first stage uses residual vector quantization (RVQ) to learn codebooks of representative time-series patterns; these codebooks serve as a temporal vocabulary. This approach expands the native vocabulary of the pretrained LLM with discrete temporal tokens. In the second stage (Modality Alignment), we propose a unified multimodal autoregressive generation architecture where both natural language and temporal token embeddings

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

are projected into a common dimension, enabling the LLM to handle them interchangeably over the expanded vocabulary space. By introducing special sentinel tokens (<TS_start> and <TS_end>), we use a single output-head architecture that naturally supports the interleaved generation of text and numerical tokens from a shared vocabulary. This approach allows samples with time-series data and textual context prompts to be jointly trained under a standard causal language modeling paradigm, achieving cross-modal semantic alignment. Such design enables a compact LLM to process textual task descriptions and temporal patterns within a unified causal generation framework. Furthermore, VQ-LLMTS can be adapted to forecasting tasks by replacing the output head with a regression head and updating the LoRA weights, temporal projector, and regression head on task-specific data.

To train the Stage I RVQ codebooks, we design a generalized hybrid data generator covering eight temporal pattern families. This exposes the learned vocabulary to a diverse range of temporal patterns. The codebooks are optimized using reconstruction and commitment losses on these synthetic data. Subsequent reconstruction experiments on real-world datasets assess whether the learned representation transfers beyond synthetic signals. For the second stage, we curate a multimodal corpus covering both general capabilities and time-series understanding, following the paradigms established by (Hollmann et al., 2025; Yang et al., 2025; Xie et al., 2025). This multimodal corpus, together with the Unified Time Series Dataset (UTSD-1G), a large-scale real-world univariate time-series corpus introduced by Timer (Liu et al., 2024c), is used for modality-alignment training. These diverse datasets bridge numerical sequences and semantic reasoning, enhancing the model's capacity for cross-modal analysis and time-series forecasting.

We evaluate VQ-LLMTS from three perspectives: time-series understanding, univariate forecasting, and context-aware forecasting. Evaluation uses samples that do not overlap with the alignment training data; the time-series understanding test uses the same synthetic generator and annotation scheme as the training data for that task. For time-series understanding, VQ-LLMTS achieves top scores across all five metrics, outperforming general-purpose LLMs (e.g., GPT-5.4) and domain-specific models such as ChatTS-14B. For univariate forecasting, VQ-LLMTS and its fine-tuned variant (VQ-LLMTS-Reg) demonstrate competitive performance against LLM backbones, pre-trained foundation models, and task-specific baselines in the reported conditional setting, which supplies qualitative descriptions derived from the observed target window to some models but not to all baselines. Under shared pretraining sources and target-derived textual context, VQ-LLMTS outperforms direct SFT of the base LLM in Experiment A and improves weighted-average MASE and OWA on M4, though not the weighted-average sMAPE; these comparisons support discrete temporal tokenization and cross-modal alignment without establishing history-only forecasting superiority. Moreover, VQ-LLMTS-Reg achieves further gains after adapting the regression head, LoRA weights, and temporal projector. For context-aware forecasting, VQ-LLMTS obtains a lower primary RCRPS score than Chronos and the compared general-purpose LLMs on the CiK benchmark. Collectively, these results demonstrate the strength of our temporal tokenization and cross-modal alignment framework across diverse time-series tasks. Our code is available at https://anonymous.4open.science/r/VQ-LLMTS-7948/README.md.

2 Related Work

Multimodal time-series forecasting, which leverages contextual insights from natural language descriptions, has attracted growing interest (Xu & Cohen, 2018; Liu et al., 2025b). Early approaches simply concatenated text embeddings and numerical features into a unified encoder. More recently, the semantic reasoning capabilities of large language models (LLMs) have motivated further research. To enable LLMs to process continuous time-series data, recent work has explored two common paradigms. The first adapts LLMs through prompt engineering, input reprogramming, or lightweight fine-tuning without modifying the core token vocabulary. LLMTime (Gruver et al., 2023) demonstrates that frozen LLMs can serve as effective zero-shot forecasters via careful numerical tokenization. Time-LLM (Jin et al., 2024) reprograms time-series patches into soft text-prototype embeddings via cross-attention alignment. TEMPO (Cao et al., 2024) decomposes time series with a learnable prompt pool, and TimeXL (Jiang et al., 2026) employs a closed-loop multi-LLM refinement process.

The second paradigm projects raw time series into continuous dense embeddings for input to pre-trained LLMs, e.g., GPT4TS (Zhou et al., 2023), AutoTimes (Liu et al., 2024b), S^2IP-LLM (Pan et al., 2024), Meta-Transformer (Zhang et al., 2023), UniTime (Liu et al., 2024a), and Time-VLM (Zhong

et al., 2025). Representative approaches freeze most LLM layers while fine-tuning projections (Zhou et al., 2023), align semantic spaces with time-series embeddings via cosine similarity (Pan et al., 2024), or unify multiple modalities with a shared encoder (Zhang et al., 2023). Despite their practical success, these approaches continue to face challenges in aligning temporal representations with language tokens. As identified in recent surveys (Liu et al., 2025a; Zhang et al., 2024; Jiang et al., 2024), two persistent bottlenecks remain: text-based encoding can lose numerical precision, while continuous-embedding projection requires explicit alignment with the token embeddings of autoregressive LLMs (Kou et al., 2025).

Vector Quantization (VQ) maps continuous signals into discrete codebook entries while preserving reconstruction fidelity (Gray, 1984). VQ-VAE (van den Oord et al., 2017) demonstrated that discrete latent representations can be learned end-to-end, a paradigm successfully scaled to image generation (VQ-VAE-2, Razavi et al. (2019), DALL-E, Ramesh et al. (2021)) and audio compression (SoundStream, Zeghidour et al. (2022), EnCodec, Défossez et al. (2023)), with a recent survey (Li et al., 2026) providing a comprehensive overview of discrete tokenization in multimodal systems. In the time series domain, TOTEM (Talukder et al., 2024) applies VQ-VAE to learn domain-agnostic forecasting tokens; however, these tokens are fed into task-specific prediction heads rather than integrated into a shared LLM vocabulary, limiting the use of semantic knowledge and autoregressive reasoning capabilities of pre-trained LLMs.

VQ-LLMTS differs from existing work in three key aspects:

- **Discrete Temporal Representation.** Unlike continuous embedding projection methods, we quantize time series into discrete tokens that natively inhabit the LLM's vocabulary space, reducing continuous-discrete incompatibility and enabling lexical alignment.
- **Plug-in Temporal Vocabulary Expansion.** Unlike the task-specific prediction-head setup reported for TOTEM, temporal tokens are integrated into the LLM's vocabulary, enabling joint training under a single causal language modeling objective with a shared generation head.
- **Unified End-to-End Autoregressive Framework.** Unlike prompt-based methods that treat LLMs as feature extractors or re-programmers, our framework supports end-to-end joint training of text and temporal tokens within a single autoregressive pipeline for forecasting and instruction-following analysis.

3 VQ-LLMTS AND TRAINING STRATEGY

VQ-LLMTS is built on a two-stage methodological framework. In the first stage, **vector-quantized codebook training**, a residual vector-quantized variational autoencoder (RVQ-VAE) learns codebooks that quantize continuous time-series signals into compact, ordered discrete codeword representations. This is illustrated in Figure 1. The second stage, **modality alignment** (Figure 2), processes text and time-series inputs through their respective tokenizers — the native LLM tokenizer and the RVQ tokenizer from the pre-trained codebooks. The temporal tokens from the RVQ tokenizer are then projected into the LLM's latent space via a modality projector. The interleaved token sequence is fed into an augmented LLM, enabling context-aware forecasting by exploiting the LLM's inherent reasoning and sequence modeling capabilities within a unified token space.

3.1 RESIDUAL VECTOR QUANTIZED CODEBOOK TRAINING STRATEGY

The architecture features a symmetric hierarchical encoder–decoder structure, a multi-stage residual quantizer, and a boundary-preserving padding and cropping mechanism to mitigate edge effects.

Symmetric 1D Feature Encoder-Decoder. The **Encoder** E projects the input signal $x \in \mathbb{R}^{1\times L}$ into a latent representation $z_e \in \mathbb{R}^{D_{TS}\times \frac{L}{P}}$, where D_{TS} is the embedding dimension and P denotes the effective temporal downsampling factor. To effectively capture multi-scale temporal correlations, we employ a hierarchical convolutional architecture. Each processing unit integrates a feature expansion layer to project features into a higher-dimensional manifold, a downsampling layer to reduce temporal redundancy and expand the receptive field, and a Residual Block (ResBlock1D) to ensure stable optimization and mitigate gradient degradation. In the reported configuration, the product of the

layer strides gives $P = 1$, so each time step corresponds to one latent patch (Appendix A.2.3). The **Decoder** D maps the quantized vector z_q back to the original signal space $\hat{x} \in \mathbb{R}^{1\times L}$.

Multi-stage Residual Vector Quantization (RVQ). In contrast to single-codebook approaches—which often suffer from limited expressiveness— we employ a multi-stage recursive quantization strategy to decompose the latent feature vector z_e in a "coarse-to-fine" manner. To achieve hierarchical discretization, the RVQ layer comprises I codebooks $\mathcal{Q}_i = \{q_{i,1}, \ldots, q_{i,K}\} \subset \mathbb{R}^{D_{TS}}$ for $i \in \{1, \ldots, I\}$, each with K entries of embedding dimension D_{TS}. The first stage maps z_e to the nearest code vector $z_{q,1}$ in $\mathcal{Q}_1$, capturing coarse temporal patterns. At stage i, the residual is $r_i = z_e - \sum_{j=1}^{i-1} z_{q,j}$ (the empty sum is zero for $i = 1$), and $z_{q,i} = q_{i,k_i}$ is the nearest entry to r_i in $\mathcal{Q}_i$, with $k_i \in \{1, \ldots, K\}$. The final quantized representation is $z_q = \sum_{i=1}^{I} z_{q,i}$. These operations apply independently at each latent patch.

To ensure high-fidelity reconstruction, we use symmetric boundary **replicate padding**. This padding makes the temporal dimension compatible with the downsampling requirements of the hierarchical layers. During decoding, we enforce strict temporal alignment by **center-cropping** the output to the original length L and discarding the distorted padding, as defined in equation (1).

$$\hat{x} = \mathrm{Crop}_{[P:P+L]}(D(z_q(E(\mathrm{Pad}_P(x))))) \tag{1}$$

The quantization process reduces the temporal resolution by a factor of P and maps a length-L unpadded input to $(L/P)I$ codeword indices when P divides L (in our configuration, $P = 1$); each selected codeword is D_{TS}-dimensional. The resulting token count scales linearly with the input length.

Loss Function. The model is optimized using a weighted sum of the reconstruction loss, $\|x - \hat{x}\|_2^2$, and the commitment loss. For each batch B, the total loss is defined as follows:

$$\mathcal{L}_{total} = \frac{1}{B}\sum_{i=1}^{B} \|x_i - \hat{x}_i\|_2^2 + \beta \cdot \frac{1}{B}\sum_{i=1}^{B} \|z_e(x_i) - \mathrm{sg}[z_q(x_i)]\|_2^2 \tag{2}$$

where z_e is the continuous encoder output, z_q is the aggregated discrete vector, $\mathrm{sg}[\cdot]$ denotes the stop-gradient operation, and β is the commitment-loss weight. Its value and further RVQ training details are given in Appendix A.2.

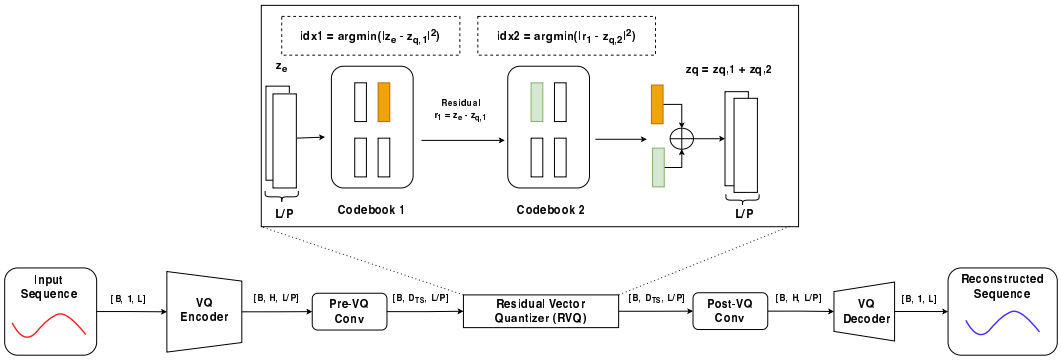

Figure 1: Feature extraction and reconstruction with the residual vector-quantized variational autoencoder (RVQ-VAE).

3.2 Modality Alignment Strategy

We address the inherent modality discrepancy that LLMs face when dealing with time-series analysis by expanding the LLM's tokenizer to include the discrete temporal tokens derived from the VQ codebooks described above. In addition, we design a prompt template that interleaves these newly introduced temporal tokens with natural language instructions to facilitate alignment training, as illustrated in Figure 2.

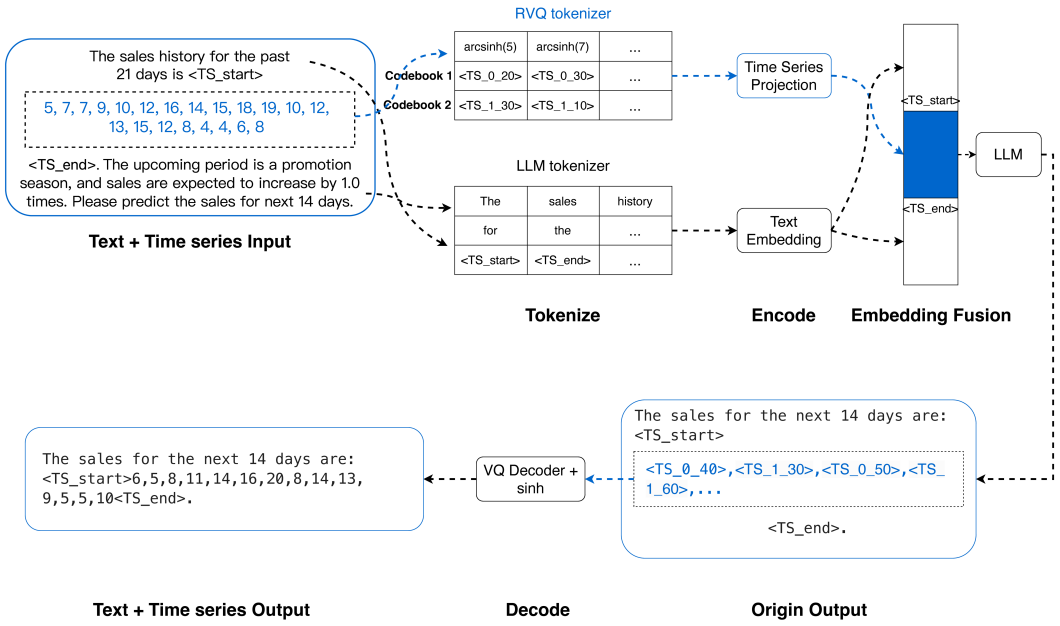

Figure 2: Overview of the modality-alignment process. The TS-Projector aligns the embedding space of discrete temporal tokens with the LLM's text token space.

Tokenization. Numerical variables are converted into tokens formatted as `<TS_i_n>`, where $i \in \{1, \ldots, I\}$ denotes the codebook level and $n \in \{1, \ldots, K\}$ denotes an entry within that codebook. For each patch in chronological order, we append the indices $k_1, \ldots, k_I$ from the first through the I-th quantizer; the sequence then proceeds to the next patch. Although this strategy increases the total sequence length by a factor of I, it yields a highly efficient representation space: by employing only I tokens per patch, the model is capable of characterizing up to K^I unique quantized states per patch (per time step when $P = 1$). This approach significantly reduces the overhead of expanding the codebook vocabulary while maintaining the expressive power required for high-fidelity signal representation. In addition, all time-series data tokens are encapsulated by special markers `<TS_start>` and `<TS_end>` to clearly delimit the time-series context for the LLM.

TS-Projector Layer. We introduce a trainable projection layer, where W_{proj} denotes the TS-projection matrix in equation (3), to bridge the dimensional and semantic gap between the VQ codebook space and the LLM embedding space.

Hybrid Embedding Layer. After tokenization, for an input token index v, the embedding h is computed by selecting the corresponding representation from either the original or the temporal vocabulary:

$$h = \begin{cases} E_{\text{orig}}[v] & \text{if } v \in \mathcal{V}_{\text{orig}} \\ W_{\text{proj}} \cdot E_{\text{ts}}[v - s_{\text{ts}}] & \text{if } v \in \mathcal{V}_{\text{ts}} \end{cases} \tag{3}$$

where $\mathcal{V}_{\text{orig}}$ and $\mathcal{V}_{\text{ts}}$ denote the original LLM vocabulary and the specialized temporal vocabulary, respectively. s_{ts} represents the starting index of the temporal tokens in the expanded vocabulary space. E_{orig} refers to the original embedding layer of the LLM, while E_{ts} stores the learned temporal token embeddings, aligned via the projection matrix W_{proj}.

Output Head Expansion. The LLM's prediction head is expanded to include the VQ codebooks, enabling the model to directly generate future temporal tokens autoregressively. Stage II transforms a pre-trained LLM (e.g., Qwen2.5-1.5B-Instruct) into a hybrid model capable of simultaneously processing raw text and compressed temporal tokens. The model receives an interleaved sequence of historical temporal data, date sequences, and descriptive textual context, autoregressively generating future trajectories. We utilize Parameter-Efficient Fine-Tuning (PEFT) via LoRA for the LLM

backbone, while the TS-Projector and TS-specific LLM prediction heads are fully updated. The entire architecture is optimized using a standard next-token prediction (NTP) loss. During inference, the tokens generated between the <TS_start> and <TS_end> markers are decoded back into numerical values using the VQ decoder.

4 Datasets, Experiments and Evaluation

We train the first stage VQ-VAE on synthetic time-series data, and perform modality-alignment pretraining using both another 300,000-sample synthetic modality-alignment corpus and the Unified Time Series Dataset (UTSD-1G) in the second stage. Our evaluation covers three catogories: **time-series understanding**, **univariate forecasting**, and **context-aware forecasting**. For each task, we conduct experiments on datasets distinct from those used during training, thereby assessing the model's ability to generalize to unseen data. To disentangle the contributions of different components to univariate forecasting, we compare VQ-LLMTS, direct SFT of the same LLM, and VQ-LLMTS with a regression head using the same training and evaluation datasets. This allows us to isolate the roles of temporal tokenization and prediction-head design. Experiments were run on a server with four NVIDIA A100-80GB GPUs, 64 CPU cores, and 512 GB of memory (detailed in appendix A.1.4).

4.1 Training Dataset Description

Hybrid Time-Series Data Generation for VQ Training. The generator combines eight temporal modes linearly or nonlinearly and adds high-frequency and Gaussian noise (Appendix A.1.1). Before encoding, we compress large-magnitude signals using $x_{\text{in}} = \operatorname{arsinh}(x)$ and restore their scale with $\sinh$ after decoding. Through extensive experiments, we identified that the setup of 2 Codebooks + 512 Tokens achieves the optimal balance between accuracy and computational efficiency. The learned RVQ codebooks reconstruct the evaluated short and long sequences with low error (Appendix A.2).

Alignment Dataset Part I: Synthetic Modality Alignment Corpus. We construct $300{,}000$ synthetic text–time-series pairs covering two Types. Generation procedures and examples are provided in Appendix A.1.2. **Type I: General Capability** contains $100{,}000$ synthetic samples for pattern recognition and $100{,}000$ samples encoding directed $X \to Y$ causal dependencies via graph-based generative networks. **Type II: Time-Series Understanding** comprises $100{,}000$ samples that describe time-series patterns in natural language across four attributes: periodicity, trend, local features, and noise (Xie et al., 2025).

Alignment Dataset Part II: Unified Time Series Dataset (UTSD-1G). We use a subset of real-world UTSD-1G Dataset introduced by Liu et al. (2024c), spanning energy, environment, health, IoT, nature, transportation, and web domains. Each univariate series is segmented into non-overlapping samples with a fixed history length of 96 and a prediction horizon randomly sampled between 7 and 28 steps, yielding $1{,}048{,}517$ training samples. Each prompt pairs history-derived textual descriptions (trend, variability, autocorrelation, and level) and target-derived text (direction, fluctuations, and extrema locations).See Appendix A.1.3 for detailed generation.

4.2 Evaluation: Time-Series Understanding

The time-series understanding evaluation follows the same data-generation scheme as Type II: Time-Series Understanding in the Synthetic Modality Alignment Corpus (Part I of the Alignment Dataset), while using independently generated time-series values that do not overlap with those used for alignment pretraining. This is a held-out-sample evaluation under the same generator and annotation scheme, not an out-of-distribution test across data sources. We evaluate the correspondence between generated descriptions and time-series characteristics using **BLEU** (with and without numerical tokens), **Regex Keyword** and **Regex Num** (qualitative and numerical attribute accuracy), and **LLM-as-a-Judge** (a 100-point assessment of trend, periodicity, local characteristics, noise, statistics, and numerical precision); the scoring rules are given in Appendix A.4. As shown in Table 1, VQ-LLMTS achieves the highest score on all five reported metrics, and these results indicate that the learned temporal representation supports both structural descriptions and numerically grounded explanations.

Table 1: Performance Comparison for Time-Series Understanding

Method	BLEU	BLEU (w/o num)	Regex Keyword	Regex Num	LLM-Judge (GPT-5.4)
Qwen2.5-1.5B	0.3844	0.5433	0.6865	0.2701	18.45
Qwen2.5-72B	0.4989	0.5645	0.4612	0.2436	27.77
DeepSeek-V3	0.6079	0.6941	0.5630	0.2657	33.26
GPT-4.1	0.5896	0.6468	0.6081	0.2753	34.03
GPT-5.4	0.6201	0.6701	0.5702	0.2568	39.99
ChatTS-14B	0.4844	0.6026	0.5145	0.2711	52.09
VQ-LLMTS	**0.8879**	**0.9676**	**0.9418**	**0.8821**	**77.14**

4.3 Evaluation: Univariate Forecasting

Evaluation protocol. We evaluate short-horizon forecasting (Experiment A) on eight public datasets: ETTh1, ETTh2, ETTm1, ETTm2, Electricity, ILI, Traffic, and Weather. Each dataset has separate training, validation, and test partitions. We further evaluate forecasting performance on the M4 competition dataset (Experiment B) across six frequencies: Yearly, Quarterly, Monthly, Weekly, Daily, and Hourly. These two groups of dataset statistics and experimental configurations are provided in Appendix Table 9 and Table 10 respectively. Both are excluded from the training corpus.

We consider three groups of forecasting benchmarks: language-model-based backbones, such as Time-LLM (Jin et al., 2024); pretrained time-series foundation models, including Chronos (Ansari et al., 2024) and Timer (Liu et al., 2024c); and task-specific time-series forecasting models, such as PatchTST (Nie et al., 2023) and Informer (Zhou et al., 2021).

In addition, we compare VQ-LLMTS, direct SFT of the same Qwen2.5-1.5B-Instruct LLM without vector quantization (LLM-SFT), and VQ-LLMTS with a regression head (VQ-LLMTS-Reg). VQ-LLMTS-Reg is built on the pretrained VQ-LLMTS by replacing the next-token generation head with a regression head and adapting the LoRA weights, temporal projector, and regression head on task-specific training data. All three models are pretrained on the same two alignment datasets (the Synthetic Modality Alignment Corpus and UTSD-1G) and followed by task-specific tuning, providing a controlled comparison of representation and prediction-head design. At test time, VQ-LLMTS, LLM-SFT, VQ-LLMTS-Reg, and Time-LLM receive the same qualitative text prompt describing the forecast period, without access to the target values. The remaining 14 baselines do not receive these text prompts. Thus, comparisons with Time-LLM are conducted under the same contextual input, while comparisons across benchmark groups also reflect differences in input information.

Experiment A: Short-Horizon Forecasting. We evaluate VQ-LLMTS, LLM-SFT, and VQ-LLMTS-Reg alongside 15 baseline models on the eight public datasets mentioned above, using an input history of 96 steps and a fixed output horizon of 28 steps. We report MAE and MSE in standardized space, with metric definitions provided in Appendix A.5.

Table 2 compares forecasting performance on the eight datasets. VQ-LLMTS and VQ-LLMTS-Reg achieve competitive forecasting accuracy in terms of both MAE and MSE. In particular, VQ-LLMTS-Reg achieves the lowest MAE and MSE on ETTh1, ETTh2, ETTm1, and Weather. Compared with Time-LLM under the same target-derived qualitative context, VQ-LLMTS achieves lower MAE and MSE on ETTh1, ETTh2, ILI, Traffic, and Weather. On ETTh1, with the pretraining alignment data and contextual input held fixed, VQ-LLMTS reduces MAE by 25.9% and MSE by 46.2% relative to LLM-SFT. This comparison suggests that quantizing time-series data into discrete tokens and aligning temporal and textual tokens in a shared semantic space can substantially improve forecasting accuracy over directly applying SFT to the underlying LLM. VQ-LLMTS-Reg provides a lightweight adaptation of VQ-LLMTS and further improves forecasting accuracy, suggesting that jointly adapting the regression head, LoRA weights, and temporal projector can yield more precise numerical predictions for scenario-specific forecasting tasks.

Experiment B: M4 Forecasting. We follow the standard M4 evaluation protocol, treating each sample as an independent univariate series and evaluating performance using sMAPE, MASE, and the overall weighted average (OWA), with metric definitions provided in Appendix A.5. Table 3 reports the performance of all models on the M4 dataset. The reported metrics show that VQ-LLMTS-Reg performs competitively across the M4 frequency groups. In the reported results, VQ-LLMTS-Reg

Table 2: Experiment A: short-horizon forecasting results. MAE / MSE in standardized space (lower is better). Red and blue-underlined values indicate the best and second-best distinct scores per dataset and metric; ties at the reported precision receive the same marking.

Method Metric	ETTh1 MAE / MSE	ETTh2 MAE / MSE	ETTm1 MAE / MSE	ETTm2 MAE / MSE	Electricity MAE / MSE	ILI MAE / MSE	Traffic MAE / MSE	Weather MAE / MSE
Language-model backbones								
Time-LLM	0.1228 / 0.0258	0.2009 / 0.0717	0.0713 / 0.0099	0.0987 / 0.0209	0.2341 / 0.0982	2.2189 / 8.5250	0.2724 / 0.1730	0.0159 / 0.0005
GPT4TS	0.1158 / 0.0235	0.1814 / 0.0646	0.0676 / 0.0102	0.0784 / 0.0153	0.2062 / 0.0811	1.1460 / 3.4438	0.2199 / 0.1330	0.0182 / 0.0007
S^2IP-LLM	0.1165 / 0.0237	0.1905 / 0.0667	0.0675 / 0.0091	0.0757 / 0.0149	0.2055 / 0.0786	1.3009 / 3.8330	0.2204 / 0.1386	0.0185 / 0.0007
Pretrained time-series models								
Chronos	0.1464 / 0.0375	0.1994 / 0.0782	0.0921 / 0.0168	0.0885 / 0.0223	0.2197 / 0.0977	1.4715 / 5.2068	0.1328 / 0.1008	0.0127 / 0.0004
Timer	0.1692 / 0.0465	0.3234 / 0.1711	0.1204 / 0.0247	0.2823 / 0.1305	1.0026 / 1.2426	2.2685 / 8.5178	0.8237 / 0.9830	0.0235 / 0.0010
Forecasting-specific models								
DLinear	0.1209 / 0.0254	0.1867 / 0.0646	0.0799 / 0.0127	0.0886 / 0.0191	0.2114 / 0.0864	1.5081 / 4.5915	0.3166 / 0.2300	0.0255 / 0.0013
PatchTST	0.1187 / 0.0247	0.1855 / 0.0643	0.0680 / 0.0092	0.0828 / 0.0167	0.2373 / 0.0950	1.1683 / 3.3853	0.2119 / 0.1310	0.0165 / 0.0006
TimesNet	0.1216 / 0.0257	0.1876 / 0.0638	0.0693 / 0.0095	0.0823 / 0.0181	0.2013 / 0.0762	1.1708 / 3.6122	0.1683 / 0.1158	0.0188 / 0.0007
FEDformer	0.1323 / 0.0287	0.2036 / 0.0707	0.0733 / 0.0102	0.0988 / 0.0200	0.2352 / 0.1007	1.3730 / 4.5006	0.2749 / 0.1864	0.0341 / 0.0020
Autoformer	0.1764 / 0.0504	0.2703 / 0.1206	0.2309 / 0.0919	0.2931 / 0.1353	0.2313 / 0.0979	1.9311 / 7.3023	0.2139 / 0.1452	0.4149 / 0.2341
Nonstat.	0.1300 / 0.0296	0.1907 / 0.0659	0.0726 / 0.0102	0.0886 / 0.0176	0.2641 / 0.1261	1.4807 / 5.4365	0.2319 / 0.1635	0.0220 / 0.0009
ETSformer	0.1497 / 0.0365	0.2440 / 0.0966	0.1393 / 0.0335	0.2788 / 0.1346	0.2808 / 0.1400	1.5645 / 5.3189	0.4467 / 0.3718	0.0757 / 0.0094
LightTS	0.1307 / 0.0311	0.1900 / 0.0662	0.0744 / 0.0107	0.0928 / 0.0205	0.2124 / 0.0880	1.5199 / 4.6148	0.2862 / 0.2007	0.0252 / 0.0013
Informer	0.2312 / 0.0834	0.2750 / 0.1241	0.1058 / 0.0189	0.1201 / 0.0273	0.2564 / 0.1131	1.7666 / 8.0390	0.2406 / 0.1675	0.0335 / 0.0019
Reformer	0.2387 / 0.0916	0.2822 / 0.1298	0.1516 / 0.0408	0.2112 / 0.0812	0.3412 / 0.1985	1.7121 / 7.9650	0.3541 / 0.3019	0.0388 / 0.0028
Ablation configurations (shared pretraining and context)								
LLM-SFT	0.1593 / 0.0452	0.2925 / 0.1622	0.1134 / 0.0259	0.3141 / 0.2204	0.3204 / 0.2931	1.6674 / 5.9906	0.3222 / 0.4347	0.0189 / 0.0009
VQ-LLMTS	0.1180 / 0.0243	0.1805 / 0.0629	0.0941 / 0.0126	0.0986 / 0.0219	0.2298 / 0.1059	1.2653 / 4.3891	0.1342 / 0.1194	0.0133 / 0.0004
VQ-LLMTS-Reg	0.1087 / 0.0201	0.1608 / 0.0460	0.0616 / 0.0071	0.0848 / 0.0149	0.2046 / 0.0764	1.9110 / 6.7587	0.1663 / 0.1031	0.0103 / 0.0003

Table 3: Experiment B: M4 forecasting results. sMAPE / MASE / OWA (lower is better). Red and blue-underlined values indicate the lowest and second-lowest scores across all models for each frequency and metric.

Method Metric	Yearly sMAPE / MASE / OWA	Quarterly sMAPE / MASE / OWA	Monthly sMAPE / MASE / OWA	Weekly sMAPE / MASE / OWA	Daily sMAPE / MASE / OWA	Hourly sMAPE / MASE / OWA	Weighted avg. sMAPE / MASE / OWA
Language-model backbones							
Time-LLM	16.3526 / 3.3166 / 0.9176	13.0001 / 1.4962 / 1.1358	16.3899 / 1.2029 / 1.1337	17.3859 / 5.8352 / 1.9994	3.5874 / 3.8253 / 1.1724	26.4011 / 5.4604 / 1.8580	15.0716 / 1.9046 / 1.0922
GPT4TS	17.0661 / 3.6988 / 0.9875	12.7041 / 1.4309 / 1.0985	15.6439 / 1.0923 / 1.0558	13.9463 / 5.2707 / 1.7101	3.5336 / 3.6923 / 1.1433	39.3390 / 10.6838 / 3.3004	14.8456 / 1.9377 / 1.0657
S^2IP-LLM	13.7540 / 3.1489 / 0.8170	10.3669 / 1.1892 / 0.9043	13.0271 / 0.9603 / 0.9030	13.9334 / 5.3063 / 1.7158	3.1951 / 3.4621 / 1.0526	18.4118 / 2.9382 / 1.1142	12.1658 / 1.6481 / 0.8937
Pretrained time-series models							
Chronos	13.9326 / 3.2183 / 0.8312	10.0992 / 1.1657 / 0.8836	12.4893 / 0.9209 / 0.8659	11.3930 / 2.7463 / 1.1162	3.1173 / 3.3832 / 1.0278	11.0492 / 0.9371 / 0.4962	11.8416 / 1.6187 / 0.8683
Timer	29.7096 / 8.6761 / 2.0005	23.5891 / 3.0528 / 2.1842	28.1291 / 2.4388 / 2.1216	15.5956 / 10.0803 / 2.6659	6.0070 / 6.4373 / 1.9680	19.5097 / 3.2636 / 1.2120	26.3872 / 4.2206 / 2.1005
Forecasting-specific models							
DLinear	15.0528 / 3.2576 / 0.8704	11.1498 / 1.2876 / 0.9757	14.0447 / 0.9941 / 0.9542	11.9434 / 4.5392 / 1.4690	3.4003 / 3.6616 / 1.1167	19.5565 / 3.0386 / 1.1663	13.1471 / 1.7191 / 0.9497
PatchTST	14.7985 / 3.1354 / 0.8472	11.3312 / 1.2933 / 0.9861	14.6465 / 1.0210 / 0.9877	13.5075 / 4.2157 / 1.4962	3.9901 / 4.3148 / 1.3132	36.0987 / 10.2017 / 3.1116	13.5201 / 1.7614 / 0.9794
TimesNet	15.6463 / 3.3827 / 0.9043	11.8011 / 1.3148 / 1.0152	14.9097 / 1.0756 / 1.0225	14.8580 / 3.5761 / 1.4547	3.7222 / 3.9243 / 1.2097	32.7514 / 8.3201 / 2.6277	13.9338 / 1.8230 / 1.0097
FEDformer	15.2822 / 3.2198 / 0.8726	12.4508 / 1.3753 / 1.0668	16.0398 / 1.1115 / 1.0785	14.6180 / 5.2281 / 1.7390	4.8830 / 5.2656 / 1.6048	34.7839 / 11.4029 / 3.3266	14.6051 / 1.8927 / 1.0623
Autoformer	15.4419 / 3.2932 / 0.8868	12.4874 / 1.4718 / 1.1036	19.8692 / 1.5759 / 1.4296	14.5206 / 5.2868 / 1.7443	5.2356 / 5.7592 / 1.7380	31.8944 / 8.4705 / 2.6359	16.4913 / 2.1646 / 1.2457
Nonstat.	15.6765 / 3.3604 / 0.9024	12.2540 / 1.4210 / 1.0745	15.2982 / 1.1768 / 1.0835	16.3275 / 5.4793 / 1.8776	3.5819 / 3.7509 / 1.1602	44.3851 / 12.5744 / 3.8323	14.2835 / 1.9091 / 1.0572
ETSformer	19.1313 / 4.2796 / 1.1237	11.4992 / 1.3452 / 1.0126	15.0370 / 1.1164 / 1.0461	11.5263 / 3.0216 / 1.1731	3.6734 / 3.9173 / 1.2006	23.8485 / 4.1071 / 1.5061	14.6731 / 2.0365 / 1.0648
LightTS	15.0725 / 3.2680 / 0.8723	11.4831 / 1.2707 / 0.9847	15.1231 / 1.0577 / 1.0215	13.4825 / 4.5199 / 1.5496	3.4557 / 3.6452 / 1.1233	24.7434 / 3.7873 / 1.4637	13.7786 / 1.7503 / 0.9864
Informer	15.4954 / 3.2573 / 0.8839	12.1014 / 1.3605 / 1.0455	16.4672 / 1.2385 / 1.1531	12.6385 / 4.8070 / 1.5552	3.8056 / 4.0035 / 1.2354	46.4352 / 16.8923 / 4.7895	14.7710 / 1.9266 / 1.0853
Reformer	16.9699 / 3.5955 / 0.9715	13.1499 / 1.4995 / 1.1438	16.4852 / 1.2646 / 1.1660	16.1326 / 5.8498 / 1.9336	3.9842 / 4.2292 / 1.2992	39.9732 / 11.4359 / 3.4747	15.3638 / 2.0409 / 1.1339
Ablation configurations (shared pretraining and context)							
LLM-SFT	12.8409 / 7.2972 / 1.3109	10.4044 / 2.0915 / 1.2351	14.3071 / 3.4914 / 2.1373	10.0252 / 3.3594 / 1.1518	3.2588 / 3.2918 / 1.0371	27.1956 / 2.8414 / 1.3329	12.6042 / 4.0191 / 1.6773
VQ-LLMTS	13.1923 / 2.7331 / 0.7475	11.2276 / 1.1381 / 0.9247	16.9670 / 1.2380 / 1.1703	9.2290 / 2.3424 / 0.9254	3.7289 / 4.1916 / 1.2515	14.1839 / 1.0451 / 0.6040	14.1225 / 1.6859 / 1.0143
VQ-LLMTS-Reg	12.5648 / 2.7861 / 0.7349	8.9807 / 0.9701 / 0.7615	10.9303 / 0.7676 / 0.7397	6.6688 / 1.7700 / 0.6826	2.3370 / 2.3965 / 0.7492	16.6110 / 3.6346 / 1.2106	10.4833 / 1.3648 / 0.7460

has the lowest sMAPE, MASE, and OWA in the Quarterly, Monthly, Weekly, and Daily groups, as well as the lowest weighted-average scores for all three metrics. Compared with Time-LLM under the same target-derived qualitative context, VQ-LLMTS-Reg also achieves lower sMAPE, MASE, and OWA. VQ-LLMTS reduces the weighted-average MASE from 4.0191 for LLM-SFT to 1.6859, while VQ-LLMTS-Reg achieves a lower weighted-average MASE of 1.3648. Unlike in Experiment A, VQ-LLMTS shows mixed results on M4: its weighted-average MASE and OWA improve relative to LLM-SFT, whereas sMAPE does not.

4.4 Evaluation: Context-aware Forecasting on CiK Dataset

We evaluate VQ-LLMTS on the official test set of CiK for context-aware forecasting, which contains 355 instances from 71 tasks. Representative scenarios are presented in Appendix Table 13.

We report the benchmark's capped, task-normalized Reweighted CRPS (RCRPS) as the primary metric, together with overall CRPS, RoI CRPS (within the context-specified region of interest), Non-RoI CRPS (outside that region), and Constraint CRPS (penalizing probability mass assigned to values that violate textual constraints). Lower values indicate better performance for all metrics; formal definitions are provided in Appendix A.8. Table 4 compares VQ-LLMTS with Chronos (Ansari

et al., 2024), UniTime (Liu et al., 2024a), Time-LLM (Jin et al., 2024), and general-purpose language models.

On CiK, VQ-LLMTS achieves the lowest context-conditioned RCRPS (0.2730) among the compared methods, versus 0.3538 for the next-best context-enabled model in Table 4. Its RCRPS decreases from 0.2820 without context to 0.2730 with context. The RoI CRPS decreases by 5.53%, compared with a 4.07% reduction in Non-RoI CRPS, indicating a larger descriptive improvement within the context-specified region. VQ-LLMTS also achieves the lowest Constraint CRPS among the compared methods with context, with a score of 0.0259.

Table 4: CiK forecasting with VQ-LLMTS (lower is better). RCRPS is the primary score; red and blue-underlined entries mark the best and second-best distinct results per metric across all rows.

Model	Ctx.	RCRPS↓	CRPS↓	RoI CRPS↓	Non-RoI CRPS↓	Constraint CRPS↓
VQ-LLMTS	Yes	0.2730	0.2471	0.2751	0.2191	0.0259
	No	0.2820	0.2598	0.2912	0.2284	0.0222
DeepSeek-V4	Yes	0.3538	0.3087	0.3374	0.2800	0.0451
	No	0.3540	0.3088	0.3375	0.2801	0.0451
Qwen2.5-1.5B-Instruct	Yes	0.5508	0.4426	0.4790	0.4062	0.1158
	No	0.5991	0.3521	0.3809	0.3233	0.2470
Chronos	n/a	0.3906	0.2651	0.3195	0.2107	0.1255
UniTime	Yes	0.3645	0.2645	0.2953	0.2337	0.1000
	No	0.3992	0.2685	0.2992	0.2378	0.1307
Time-LLM	Yes	0.6577	0.3035	0.3326	0.2744	0.3542
	No	0.5839	0.3039	0.3327	0.2752	0.2800

5 Conclusion

VQ-LLMTS learns a discrete temporal vocabulary through residual vector quantization and integrates temporal tokens into an autoregressive language model for joint text–time-series modeling. On the evaluated benchmarks, VQ-LLMTS achieves the best performance on all five time-series understanding metrics and the lowest context-conditioned RCRPS on CiK. Reconstruction experiments further demonstrate that the learned codebooks preserve time-series information across signals with diverse scales. Under matched pretraining and contextual inputs, VQ-LLMTS outperforms direct fine-tuning of the underlying LLM, supporting the effectiveness of VQ-based temporal representations for cross-modal alignment. These results demonstrate the potential of discrete temporal tokenization as a bridge between numerical dynamics and language-based interpretation and forecasting.

Limitations. Fine-grained RVQ requires a number of temporal tokens that grows with both sequence length and codebook depth, increasing pressure on the LLM context window and constraining long-horizon modeling. Moreover, although the evaluations in Experiments A and B do not directly expose target values, they rely on privileged expert information about the forecast period. Our analysis on ETTh1 in Appendix A.7.2 examines how the granularity and precision of textual descriptions of the forecast window affect forecasting performance, but a more comprehensive evaluation of this dependence is still needed. Future work will explore more token-efficient quantization and adaptive token allocation, as well as forecasting with independently supplied and potentially noisy future context.

AI use statement

Generative AI tools were used solely to assist with language editing, including grammar correction, wording refinement, and improvement of readability. These tools were not used to generate scientific claims, experimental results, data, figures, or citations. All AI-assisted text was carefully reviewed and revised by the authors. The authors take full responsibility for the accuracy, originality, and integrity of the final manuscript.

Ethics statement

This work studies time-series forecasting and understanding using heterogeneous data sources. We use data in accordance with applicable licenses and privacy requirements and do not include personal data. We acknowledge that forecasting systems can affect operational decisions; therefore, their outputs should be monitored and used with appropriate human oversight.

Reproducibility statement

The main paper and appendix describe the model architecture, training strategy, data-generation procedure, evaluation protocols, and ablation configurations. These details are intended to support independent verification and reproduction of the reported results. Additional implementation and data-access details will be provided in the anonymous supplementary material where permitted.

References

Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Bernie Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=gerNCVqqtR. Expert Certification.

Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, and Neil Zeghidour. Audiolm: A language modeling approach to audio generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:2523–2533, 2023. doi: 10.1109/TASLP.2023.3288409.

Defu Cao, Furong Jia, Sercan O Arik, Tomas Pfister, Yixiang Zheng, Wen Ye, and Yan Liu. Tempo: Prompt-based generative pre-trained transformer for time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5132940b1bced8a7b28e9695d49d435a-Abstract-Conference.html.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10148–10167. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/das24c.html.

Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=ivCd8z8zR2. Featured Certification, Reproducibility Certification.

Weiwei Fang, Wenhao Zhuo, Jingwen Yan, Youyi Song, Dazhi Jiang, and Teng Zhou. Attention meets long short-term memory: A deep learning network for traffic flow forecasting. *Physica A: Statistical Mechanics and its Applications*, 587:126485, 2022. ISSN 0378-4371. doi: 10.1016/j.physa.2021.126485. URL https://www.sciencedirect.com/science/article/pii/S0378437121007585.

R. Gray. Vector quantization. *IEEE ASSP Magazine*, 1(2):4–29, 1984. doi: 10.1109/MASSP.1984.1162229.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G Wilson. Large language models are zero-shot time series forecasters. In *Advances in Neural Information Processing Systems*, volume 36, pp. 19622–19635, 2023. doi: 10.52202/075280-0861. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/3eb7ca52e8207697361b2c0fb3926511-Paper-Conference.pdf.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025. doi: 10.1038/s41586-024-08328-6. URL https://doi.org/10.1038/s41586-024-08328-6.

Yushan Jiang, Zijie Pan, Xikun Zhang, Sahil Garg, Anderson Schneider, Yuriy Nevmyvaka, and Dongjin Song. Empowering time series analysis with large language models: A survey. In Kate Larson (ed.), *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pp. 8095–8103. International Joint Conferences on Artificial Intelligence Organization, 8 2024. doi: 10.24963/ijcai.2024/895. URL https://doi.org/10.24963/ijcai.2024/895. Survey Track.

Yushan Jiang, Wenchao Yu, Geon Lee, Dongjin Song, Kijung Shin, Wei Cheng, Yanchi Liu, and Haifeng Chen. Timexl: Explainable multi-modal time series prediction with llm-in-the-loop. 2026. URL https://arxiv.org/abs/2503.01013.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-llm: Time series forecasting by reprogramming large language models. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.01728.

Siqi Kou, Jiachun Jin, Zhihong Liu, Chang Liu, Ye Ma, Jian Jia, Quan Chen, Peng Jiang, and Zhijie Deng. Orthus: Autoregressive interleaved image-text generation with modality-specific heads. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 31619–31633. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/kou25a.html.

Jindong Li, Yali Fu, Jiahong Liu, Linxiao Cao, Wei Ji, Menglin Yang, Irwin King, and Ming-Hsuan Yang. Discrete tokenization for multimodal llms: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(8):9093–9113, 2026. doi: 10.1109/TPAMI.2026.3676982.

Li Li, Xiaonan Su, Yi Zhang, Yuetong Lin, and Zhiheng Li. Trend modeling for traffic time series analysis: An integrated study. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):3430–3439, 2015. doi: 10.1109/TITS.2015.2457240.

Chenxi Liu, Shaowen Zhou, Qianxiong Xu, Hao Miao, Cheng Long, Ziyue Li, and Rui Zhao. Towards cross-modality modeling for time series analytics: A survey in the llm era. In James Kwok (ed.), *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pp. 10564–10572. International Joint Conferences on Artificial Intelligence Organization, 8 2025a. doi: 10.24963/ijcai.2025/1173. URL https://doi.org/10.24963/ijcai.2025/1173. Survey Track.

Haoxin Liu, Harshavardhan Kamarthi, Zhiyuan Zhao, Shangqing Xu, Shiyu Wang, Qingsong Wen, Tom Hartvigsen, Fei Wang, and B. Aditya Prakash. How can time series analysis benefit from multiple modalities? a survey and outlook. 2025b. URL https://arxiv.org/abs/2503.11835.

Xu Liu, Junfeng Hu, Yuan Li, Shizhe Diao, Yuxuan Liang, Bryan Hooi, and Roger Zimmermann. Unitime: A language-empowered unified model for cross-domain time series forecasting. In *Proceedings of the ACM Web Conference 2024*, pp. 4095–4106, 2024a. doi: 10.1145/3589334.3645434. URL https://doi.org/10.1145/3589334.3645434.

Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Autotimes: Autoregressive time series forecasters via large language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 122154–122184, 2024b. doi: 10.52202/079017-3882. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/dcf88cbc8d01ce7309b83d0ebaeb9d29-Paper-Conference.pdf.

Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Generative pre-trained transformers are large time series models. In Ruslan Salakhutdinov, Zico

Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32369–32399. PMLR, 21–27 Jul 2024c. URL https://proceedings.mlr.press/v235/liu24cb.html.

Mike A Merrill, Mingtian Tan, Vinayak Gupta, Thomas Hartvigsen, and Tim Althoff. Language models still struggle to zero-shot reason about time series. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 3512–3533, 2024. doi: 10.18653/v1/2024.findings-emnlp.201. URL https://aclanthology.org/2024.findings-emnlp.201/.

Mohammad Amin Morid, Olivia R. Liu Sheng, and Joseph Dunbar. Time series prediction using deep learning methods in healthcare. *ACM Trans. Manage. Inf. Syst.*, 14(1), January 2023. ISSN 2158-656X. doi: 10.1145/3531326. URL https://doi.org/10.1145/3531326.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. 2023. URL https://arxiv.org/abs/2211.14730.

Zijie Pan, Yushan Jiang, Sahil Garg, Anderson Schneider, Yuriy Nevmyvaka, and Dongjin Song. s^2IP-LLM: Semantic space informed prompt learning with LLM for time series forecasting. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 39135–39153. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/pan24c.html.

Meng Qi, Yuanyuan Shi, Yongzhi Qi, Chenxin Ma, Rong Yuan, Di Wu, and Zuo-Jun Shen. A practical end-to-end inventory management model with deep learning. *Management Science*, 69(2): 759–773, 2023. doi: 10.1287/mnsc.2022.4564. URL https://doi.org/10.1287/mnsc.2022.4564.

Xiangfei Qiu, Hanyin Cheng, Xingjian Wu, Junkai Lu, Jilin Hu, Chenjuan Guo, Christian S. Jensen, and Bin Yang. A comprehensive survey of deep learning for multivariate time series forecasting: A channel strategy perspective. 2026. URL https://arxiv.org/abs/2502.10721.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8821–8831. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/ramesh21a.html.

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

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/5f8e2fa1718d1bbcadf1cd9c7a54fb8c-Paper.pdf.

Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181, 2020. ISSN 1568-4946. doi: 10.1016/j.asoc.2020.106181. URL https://www.sciencedirect.com/science/article/pii/S1568494620301216.

Sabera J Talukder, Yisong Yue, and Georgia Gkioxari. TOTEM: TOkenized time series EMbeddings for general time series analysis. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=QlTLkH6xRC.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, 2017. URL https://arxiv.org/abs/1711.00937.

Andrew Robert Williams, Arjun Ashok, Étienne Marcotte, Valentina Zantedeschi, Jithendaraa Subramanian, Roland Riachi, James Requeima, Alexandre Lacoste, Irina Rish, Nicolas Chapados, and Alexandre Drouin. Context is key: A benchmark for forecasting with essential textual information. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 66887–66944. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/williams25a.html.

Yemane Wolde-Rufael. Electricity consumption and economic growth: a time series experience for 17 african countries. *Energy Policy*, 34(10):1106–1114, 2006. ISSN 0301-4215. doi: 10.1016/j.enpol.2004.10.008. URL https://www.sciencedirect.com/science/article/pii/S0301421504003155.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53140–53164. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/woo24a.html.

Zhe Xie, Zeyan Li, Xiao He, Longlong Xu, Xidao Wen, Tieying Zhang, Jianjun Chen, Rui Shi, and Dan Pei. Chatts: Aligning time series with llms via synthetic data for enhanced understanding and reasoning. *Proc. VLDB Endow.*, 18(8):2385–2398, April 2025. ISSN 2150-8097. doi: 10.14778/3742728.3742735. URL https://doi.org/10.14778/3742728.3742735.

Yumo Xu and Shay B Cohen. Stock movement prediction from tweets and historical prices. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1970–1979, 2018. doi: 10.18653/v1/P18-1183. URL https://aclanthology.org/P18-1183/.

Luoxiao Yang, Yun Wang, Xinqi Fan, Israel Cohen, jingdong chen, and Zijun Zhang. Vitime: Foundation model for time series forecasting powered by vision intelligence. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=XInsJDBIkp.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2022. doi: 10.1109/TASLP.2021.3129994.

Xiyuan Zhang, Ranak Roy Chowdhury, Rajesh K. Gupta, and Jingbo Shang. Large language models for time series: A survey. In Kate Larson (ed.), *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pp. 8335–8343. International Joint Conferences on Artificial Intelligence Organization, 8 2024. doi: 10.24963/ijcai.2024/921. URL https://doi.org/10.24963/ijcai.2024/921.

Yiyuan Zhang, Kaixiong Gong, Kaipeng Zhang, Hongsheng Li, Yu Qiao, Wanli Ouyang, and Xiangyu Yue. Meta-transformer: A unified framework for multimodal learning. 2023. URL https://arxiv.org/abs/2307.10802.

Siru Zhong, Weilin Ruan, Ming Jin, Huan Li, Qingsong Wen, and Yuxuan Liang. Time-VLM: Exploring multimodal vision-language models for augmented time series forecasting. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 78478–78497. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/zhong25a.html.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. 2021. URL https://arxiv.org/abs/2012.07436.

Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. One fits all: Power general time series analysis by pretrained lm. In *Advances in Neural Information Processing Systems*, volume 36, pp. 43322–43355, 2023. doi: 10.52202/075280-1877. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/86c17de05579cde52025f9984e6e2ebb-Paper-Conference.pdf.

A Technical Appendices and Supplementary Material

A.1 Dataset

A.1.1 VQ Hybrid Time-Series Data Generation

We train the RVQ codebooks on signals sampled from a generalized hybrid generator that varies temporal structure, numerical scale, and observation noise:

1. **Pattern Diversity:** Eight generation modes cover trends, oscillations, stochastic dynamics, and sparse events.
2. **Dynamic Amplitude Scaling:** The positive scale parameter S controls signal magnitudes (e.g., S between 100 and 10,000), exposing the codebooks to a broad numerical range.
3. **Noise Robustness:** Additive Gaussian noise and high-frequency perturbations model measurement variability.

Generative Modes Table 5 summarizes the eight generation modes and their parameterizations.

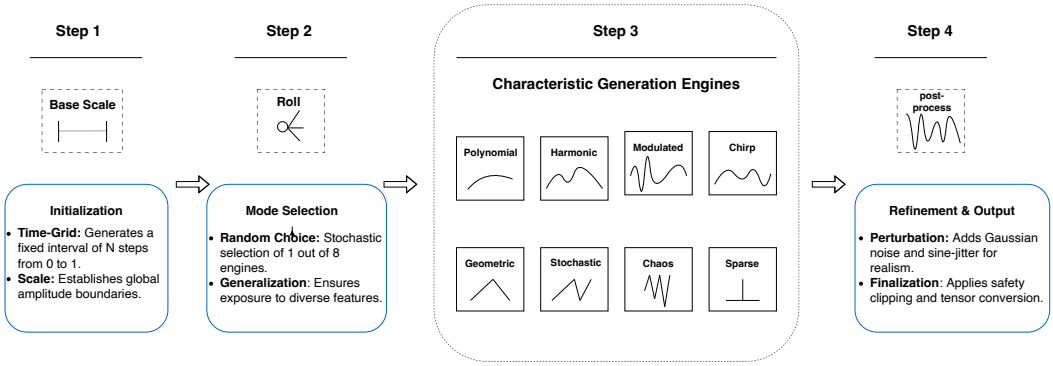

Figure 3: Overview of the synthetic data generation framework for VQ codebook training, where eight fundamental temporal pattern functions are combined to construct a controllable synthetic dataset with diverse signal patterns.

Post-processing Pipeline Each base sequence $x_{\text{base}}(t)$ undergoes the following post-processing steps to obtain $x_{\text{final}}(t)$:

1. **High-frequency Noise Injection:** With a 20% probability, high-frequency noise is injected: $x_1(t) = x_{\text{base}}(t) + 0.3S \cdot \sin(2\pi \cdot 40 \cdot t)$.
2. **Additive Gaussian Noise:** Additive noise is introduced via $x_2(t) = x_1(t) + \eta_t$, where $\eta_t \sim \mathcal{N}(0, \sigma^2)$, $\sigma = 0.02Su$, and $u \sim U(0.1, 2.0)$.
3. **Safety Clipping:** To maintain numerical stability, sequences are clipped: $x_{\text{final}}(t) = \text{clip}(x_2(t), -1.5S, 1.5S)$.

Table 5: Mathematical Formulations of Time-Series Generative Modes

Name	Mathematical Description	Key Parameter Descriptions
Polynomial Trend	$x(t) = a_0 \cdot 1.5St^3 + a_1 \cdot 1.0St^2 + a_2 \cdot 1.0St + b$	$a_i \sim U(-1, 1)$, $b \sim U(-0.5S, 0.5S)$.
Harmonic	$x(t) = 1.5 \cdot \frac{1}{\sqrt{N}} \sum_{k=1}^{N} A_k \sin(2\pi f_k t + \phi_k)$	$N \sim U\{1, \ldots, 9\}$, $A_k \sim U(0.05S, 0.5S)$, $f_k \sim U(1, 15)$, $\phi_k \sim U(0, 2\pi)$.
AM Oscillation	$x(t) = \sin(2\pi f_c t) \cdot S \cdot \sin(2\pi f_e t)$	$f_c \sim U(20, 45)$, $f_e \sim U(1, 5)$.
Chirp Signal	$x(t) = 0.8S \cdot \sin(2\pi(f_s t + \frac{1}{2}kt^2))$	$f_s \sim U(1, 5)$, $f_e \sim U(10, 30)$, $k = f_e - f_s$.
Geometric	$x(t) = 0.8S \cdot g(2\pi f t + \phi)$	$f \sim U(2, 10)$, $\phi \sim U(0, 2\pi)$.
Random Walk	$x(t) = \text{Stretch}[\sum \xi_i - P_{linear}(\sum \xi_i)]$	$\xi_i \sim N(0, (0.05S)^2)$
Logistic Chaos	$x_{t+1} = r x_t (1 - x_t)$ $x(t) = 1.6S \cdot (x_{t+10} - 0.5)$	$r \sim U(3.6, 4.0)$, $x_0 \sim U(0, 1)$.
Sparse Pulses	$x(t) = \epsilon_t + \sum_i v_i \mathbf{1}_{\{t=t_i\}}$	$\epsilon_t \sim N(0, (0.02S)^2)$, $v_i \in \{\pm S, \pm 0.8S\}$.

Amplitude Transform After signal generation, the encoder input is compressed as $x_{\text{in}} = \text{arsinh}(x_{\text{final}}) = \ln(x_{\text{final}} + \sqrt{x_{\text{final}}^2 + 1})$. The decoder output is mapped back to the original scale using the inverse transform $\hat{x}_{\text{out}} = \sinh(\hat{x}) = (e^{\hat{x}} - e^{-\hat{x}})/2$.

A.1.2 Synthetic Modality Alignment Dataset

The synthetic modality-alignment corpus comprises general-capability and time-series-understanding examples. The two subsets are detailed below.

Type I: General Capability Synthesis The general-capability subset comprises two task families:

- **Pattern Recognition (100,000 samples):** We utilize a library of 9 fundamental temporal primitives, including *exponential*, *logarithmic*, *polynomial*, *step*, *random_walk*, *poisson*, *linear_up*, *sinusoidal*, and *linear_down*. These primitives are combined to form multi-scale complexity:
  - **Single Mode:** Direct extrapolation of one base pattern.
  - **Composite Modes:** Stochastic combinations of two to four primitives (e.g., combining a linear trend with sinusoidal seasonality and Poisson noise).
  - **Stochastic Perturbations:** We inject dynamic noise (periodic and pulse-based), random mutations (up to 4 points with magnitudes within $[-20, +20]$), and an 8% probability of extreme outliers (ranging from 0.01 to 500).
- **Causal Inference (100,000 samples):** Multivariate sequences are generated using Directed Acyclic Graphs (DAGs) to simulate intricate $X \to Y$ dependencies. Each node in the graph represents a time-series variable, where child nodes are calculated as linear or nonlinear functions of their parent nodes' historical states, forcing the model to distinguish between simple correlation and directed causal influence.
- **Pattern Recognition example:** A generated history is paired with a continuation or transformation instruction; the target is sampled from the same synthetic process. These instructions define synthetic tasks rather than external contextual observations.
- **Example (Causal Inference):**

```
Prompt:  The 15-day historical sequence of Variable
1 is <TS_start>31.99,...,28.74,40.1<TS_end>,
the 15-day historical sequence of Variable
```

```
2 is <TS_start>33.7,...,33.15<TS_end>, the
future 14-day sequence of Variable 2 is
<TS_start>45.23,...,-84.54<TS_end>, predict the
future 14-day sequence of Variable 1.
Target: Variable 1 future 14-day sequence is
<TS_start>31.59,...,130.38,160.58,160.88<TS_end>
```

Type II: Time-Series Understanding Inspired by ChatTS (Xie et al., 2025), this subset maps time-series attributes to numerical sequences and corresponding natural-language descriptions:

- **Feature Mapping:** Numerical sequences are first mapped to a high-dimensional attribute pool defining four core dimensions: *periodicity*, *trend*, *local characteristics*, and *noise level*.
- **Semantic Translation:** These attributes are converted into natural language descriptions (e.g., quantifying the slope, identifying the standard deviation of noise, and pinpointing local phase changes). To improve numerical sensitivity, the targets include segment-wise statistical descriptors, such as mean values for 32 distinct sub-intervals of the sequence.
- **Example (Time-Series Understanding):**

```
Prompt: There is a time series of length 151:
[Offset: -3.3454|Scaled by: 9.2087]<TS_start>0.7128,
0.7013, ... <TS_end>. Please analyze the series in
terms of periodicity, trend, local characteristics,
and noise.
Target: The overall trend is decreasing. Local
phase changes include a decrease after an upward
spike. The value starts around 9.91 and ends at
-0.20. No periodic fluctuations are observed. The
curve is smooth with almost no noise (std ≈ 0.00).
In terms of local characteristics, an upward spike
(amplitude 5.04) occurred between points 6 and 9...
[Segment-wise means]: [9.75, 10.57, ... -0.2]. Max:
30.97, Min: -0.2.
```

A.1.3 UTSD-1G Dynamic Context Construction

The selected UTSD-1G dataset contains $68{,}679$ univariate series and approximately 121.79 million time points, covering diverse sampling frequencies, temporal spans, value scales, and dynamical regimes. Each univariate series is segmented into non-overlapping windows with a 96-point history and an output horizon sampled from 7 to 28 points, subject to the remaining series length. For each pretraining example, rule-based history and forecast-period descriptions are constructed from the observed input and target windows, respectively. For Experiments A and B, the pretrained LLM and VQ configurations are supplied with qualitative forecast-period descriptions constructed from the observed target window; numerical target values are not included in the input. The same target-derived description protocol is used for the ETTh1 forecast-period context-detail study (Appendix A.7.2); its previously reported VQ-LLMTS L2 result matches Experiment A, while the LLM-SFT and VQ-LLMTS-Reg L2 entries there use their corresponding Experiment A results. These evaluations quantify conditional forecasting with privileged future information rather than history-only forecasting.

For the history-derived description, let $H = (h_1, \ldots, h_{96})$ be the lookback window and define a scale $s_H = \max(\mathrm{std}(H), \max(H) - \min(H), 10^{-12})$. The overall trend is determined from $(h_{96} - h_1)/s_H$, with thresholds $+0.15$ and -0.15. Recent dynamics compare means of the last 24 points and the preceding 24 points, normalized by s_H; local variability is the standard deviation of first differences in the last 24 points, also normalized by s_H (thresholds 0.03 and 0.12). The prompt additionally describes the lag in $[4, 48]$ with maximum autocorrelation (strong at ≥ 0.75, moderate at ≥ 0.45) and compares the latest 24-point mean with the full-history mean using the same ± 0.15 trend thresholds.

For the future-derived description, let $F = (f_1, \ldots, f_n)$ be the target window and define $s_F = \max(|\,\mathrm{mean}(F)|, \max(F) - \min(F), 10^{-12})$. The overall direction is classified by $(f_n - f_1)/s_F$

using thresholds ± 0.12. The prompt describes monotonicity when at least 80% of first differences share a sign; otherwise, it describes fluctuations using $\mathrm{std}(\Delta F)/s_F$, with 0.03 as the mild-fluctuation threshold. When the range is nonzero and this volatility exceeds 0.03, the locations of the minimum and maximum are described as early, midway, or late in the horizon. At test time, the resulting qualitative descriptions accompany the numerical history and forecasting instruction for the pretrained configurations in Experiments A and B and the ETTh1 context-detail study. The target window is used to generate the text, but its numerical sequence is withheld from the input.

For the ETTh1 forecasting experiments, we use the OT channel with a 96-step history and a 28-step forecast horizon, yielding 8,517/2,853/2,853 train/validation/test windows. Experiment A uses the same fixed 28-step horizon across all eight datasets; these evaluation horizons are distinct from the 7–28-step sampling used only for UTSD-1G pretraining. M4 follows its official frequency-specific horizons (Table 10), including 48 steps for Hourly. Global standardization uses the mean and standard deviation fitted only on the training partition. Each training window is then instance-normalized using statistics computed from its standardized history; the same history-only normalization is applied to validation and test windows, without using target values to fit the normalization. Descriptions are constructed from the resulting normalized windows. Unlike these benchmark experiments, UTSD-1G pretraining retains values in their original scale.

A.1.4 Forecasting Training and Evaluation Protocols

Experiments were run on a server equipped with four NVIDIA A100-80GB GPUs, 64 CPU cores, and 512 GB of memory. In Experiment A, 12 training-based baselines use seed 2021, at most 10 epochs, and a validation patience of 3. Time-LLM and GPT4TS use per-device batch size 8 and learning rate 10^{-4}; the other ten training-based baselines use batch size 16 and learning rate 10^{-3} (weight decay zero). Chronos, Timer, and S^{2}IP-LLM have separate model-specific training/inference protocols and are not assigned these shared hyperparameters. The training, validation, and test window counts are listed in Table 9; validation selects checkpoints, and test data are held out for final reporting. For M4, training windows are sampled from the recent tail of the official historical series within frequency-specific limits; the last H historical points form the validation holdout, and the official test horizon is reserved for final evaluation. The twelve archived M4 training-based baseline configurations use seed 2021, up to 10 epochs, a validation patience of 3, batch size 256, evaluation batch size 512, learning rate 10^{-3}, and zero weight decay. Chronos, Timer, and S^{2}IP-LLM follow separate scripts rather than these shared batch-size and learning-rate settings. The main M4 comparisons use the archived aligned-tail protocol, not a sliding-window retraining protocol.

Independent ETTh1 context and horizon adaptations. For the additional comparisons in Tables 12 and 11, LLM-SFT and VQ-LLMTS-Reg are adapted for the context-detail and horizon comparisons using the configurations described below. The context-detail (L0–L4) and horizon (28, 56, 84, 112) studies use independently trained variants, including separate 28-step runs with different context inputs. To match the main-table comparison, the displayed L2 and 28-step entries use their respective Experiment A values; the other displayed entries use the corresponding additional runs. All variants use the chronological train/validation/test split; 28-step experiments have 2,853 test windows, and the 56-, 84-, and 112-step variants have 2,825, 2,797, and 2,769, respectively. The LLM-SFT runs initialize from the separately pretrained pure-LLM variant based on Qwen2.5-1.5B-Instruct (without VQ vocabulary expansion), use two-GPU distributed training with batch size 8 per GPU, gradient accumulation 1, one epoch, learning rate 2×10^{-4}, bf16, warmup ratio 0.03, cosine schedule, and maximum sequence length 8,192. Generative inference allocates a horizon-dependent maximum number of new tokens. The VQ-LLMTS-Reg runs initialize from the same hybrid pretrained adapter, reinitialize the horizon-specific regression head, and jointly update LoRA, temporal projector, and regression head (LoRA rank 16, alpha 32, dropout 0.05; two-GPU distributed training, batch size 8 per GPU, one epoch, learning rate 2×10^{-4}, bf16, warmup ratio 0.03, cosine schedule, MSE regression loss). The pretrained hybrid adapter supplies compatible LoRA and temporal-projector weights; its language-model head is not reused. All VQ-LLMTS-Reg test windows yield complete predictions. Each forecast is inverse-transformed from instance-normalized space using statistics of its history before MAE/MSE are computed in globally standardized space. These run-specific hyperparameters do not necessarily apply to the earlier VQ-LLMTS row or the other baselines.

A.2 RVQ Framework and Experiments

The commitment-loss weight in the Stage I RVQ-VAE objective is $\beta = 0.25$. To eliminate the "checkerboard artifacts" often caused by transposed convolutions, we employ a "Nearest Neighbor Upsampling + Smooth Convolution" strategy, ensuring high-fidelity signal reconstruction.

The residual block implements the following update:

$$z_{l+1} = z_l + \mathcal{F}(z_l, W_l) \tag{4}$$

where $\mathcal{F}$ denotes the nonlinear transformation (LayerNorm and GELU activation), and W_l represents the learnable parameters.

We replicate boundary values before encoding, as defined in equation (5). After decoding, center-cropping removes the padded positions and restores the original sequence length L:

$$x_{\text{pad}} = [\underbrace{x_1, \ldots, x_1}_{P}, x_1, \ldots, x_L, \underbrace{x_L, \ldots, x_L}_{P}] \tag{5}$$

$$\hat{x} = \hat{x}_{\text{pad}}[P : P + L] \tag{6}$$

The primary reconstruction objective is to minimize the distance between the original and reconstructed signals, defined as:

$$\underset{E,D,\mathcal{Q}}{\operatorname{argmin}} \|x - \hat{x}\|_2^2 \tag{7}$$

A.2.1 EMA Codebook Updates

We implement a quantizer dropout strategy during training, which compels the model to distribute information across all hierarchical levels and fosters the learning of robust, multi-resolution representations. Furthermore, to mitigate training instability and codebook collapse—common pitfalls of Straight-Through Estimators (STE)—we update each codebook $\mathcal{Q}_i$ via an Exponential Moving Average (EMA) mechanism, with K entries per codebook as defined in the main text.

For each level $i \in \{1, \ldots, I\}$ and codebook entry $n \in \{1, \ldots, K\}$, we maintain exponentially weighted assignment counts $N_{i,n}$ and sums of assigned residual vectors $M_{i,n}$. Given the hard assignments in a batch, these statistics are updated as follows:

$$\begin{aligned} N_{i,n}^{(t)} &= \gamma N_{i,n}^{(t-1)} + (1-\gamma) \sum_{b=1}^{B} \mathbb{1}[k_{i,b} = n], \\ M_{i,n}^{(t)} &= \gamma M_{i,n}^{(t-1)} + (1-\gamma) \sum_{b=1}^{B} r_{i,b} \mathbb{1}[k_{i,b} = n]. \end{aligned} \tag{8}$$

where $\gamma \in [0, 1]$ is the decay rate, B counts the residual vectors in the batch, $r_{i,b}$ is the residual at level i for vector b, and $k_{i,b}$ is its selected entry index. Subsequently, each codebook entry is updated by $q_{i,n} = M_{i,n}/N_{i,n}$ for nonzero $N_{i,n}$. The moving averages smooth changes in codebook assignments between training steps.

A.2.2 Detailed Experimental Results and Ablations

We evaluate the contribution of augmentation, architecture, training stabilization, and amplitude scaling to VQ encoder–decoder reconstruction. The forecasting comparisons of the no-VQ control and VQ prediction heads are reported in Tables 2 and 3.

- **Baseline:** A vanilla VQ-VAE model. It employs a standard encoder-decoder architecture with a single codebook. No specialized temporal augmentations or scale-handling mechanisms are applied.
- **Group A (+Augmentation):** Adds *Data Augmentation* strategies, including Multi-level Noise (Gaussian white noise and harmonic frequency overlays) and Dynamic Amplitude Scaling. These augmentations increase the diversity of temporal patterns and amplitudes in the training set.

- **Group B (+Architecture):** Built upon Group A, this configuration introduces *Architectural Improvements*:
  - **Residual Vector Quantization (RVQ):** Uses multiple codebooks to represent signals through hierarchical residuals, improving granularity.
  - **Pre-copy Padding:** Replicates edge values to reduce boundary artifacts during reconstruction.
- **Group C (+Training):** Adds exponential moving average (EMA) codebook updates and a learning-rate warmup schedule to Group B.
- **Full Model:** Adds the inverse hyperbolic sine amplitude transform, $f(x) = \operatorname{arsinh}(x) = \ln(x + \sqrt{x^2 + 1})$, to the Group C reconstruction model.

In Table 6, successive configurations reduce reconstruction MAE from 5.13 (baseline) to 0.21 (Group C) and 0.03 (full model); the final step yields an additional 85.7% reduction relative to Group C.

Table 6: VQ encoder–decoder reconstruction ablation. Improvement is the percentage reduction in MAE relative to the baseline; configurations are detailed in Appendix A.2.2.

Group	Included Strategies	MAE	Reduction
Baseline	No optimization	5.13	-
Group A	+ Data Augmentation	2.45	+52.2%
Group B	+ Arch. Improvement (RVQ + Boundary)	0.89	+82.7%
Group C	+ Training Opt. (EMA + Warmup)	0.21	+95.9%
Full Model	**+ Scale Optimization (Arsinh)**	**0.03**	**+99.4%**

Table 7 compares codebook and vocabulary sizes. The two-codebook, 512-token configuration attains an MAE of 0.03, matching the four-codebook, 1024-token configuration at the reported precision, with a smaller parameter footprint. Its nonzero-position MAE is 0.20, compared with 0.16 for the latter.

Table 7: Reconstruction errors and qualitative resource comparison across VQ codebook configurations. The two-codebook, 512-token configuration is highlighted as a compact choice; it does not achieve the lowest MAE-NZ.

Model Configuration	MAE	MAE-NZ	Params	Efficiency
4 Codebooks + 1024 Tokens	0.03	**0.16**	Large	Medium
2 Codebooks + 1024 Tokens	0.07	0.43	Medium	High
2 Codebooks + 2048 Tokens	0.07	0.41	Large	Medium
2 Codebooks + 512 Tokens	**0.03**	0.20	**Small**	**High**
2 Codebooks + 256 Tokens	0.09	0.57	Small	High

A.2.3 Configuration of Patch Size and Sequence Generalization

The reported encoder configuration uses an effective temporal downsampling factor $P = 1$ (unit stride across the temporal encoder). Each time step is therefore represented by one patch and I codebook tokens, where I is the number of codebooks.

We justify this design choice based on the following considerations:

- **Reconstruction Fidelity:** Larger patches reduce token count but may discard local variation; $P = 1$ retains point-level information at the expense of longer token sequences.
- **Suitability for Short-Sequence Prediction:** Point-wise tokenization preserves individual observations when only a short history is available, such as for intermittent demand.
- **Generalization to Arbitrary Lengths:** Although the VQ-VAE codebook is trained on fixed-length sequences, point-wise encoding can be applied to inputs of different lengths. Figure 4 illustrates reconstruction for two lengths:

– **Short sequence** ($L = 3$)**:** Reconstruction from a three-point input.
– **Long sequence** ($L = 10{,}000$)**:** Reconstruction of a 10,000-point input.

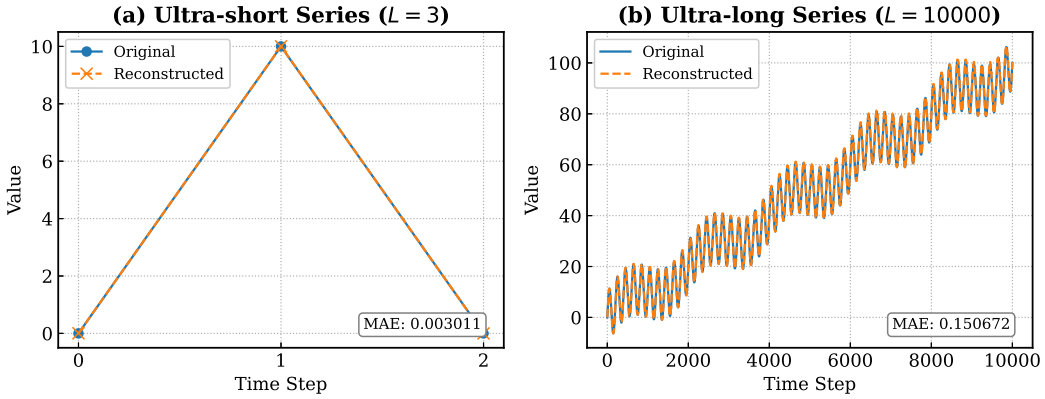

Figure 4: Reconstruction examples at sequence lengths $L = 3$ (a) and $L = 10{,}000$ (b). Mean absolute error (MAE) is reported within each panel.

A.3 VQ Encoder-Decoder Reconstruction Evaluation

We assess reconstruction fidelity of the synthetically trained RVQ-VAE on daily sales data from an anonymized online grocery retailer (10,000 sequences of length 35). We report MAE between the input x and reconstruction $\hat{x}$, as well as zero-masked MAE (MAE-NZ) over nonzero target positions; definitions appear in Appendix A.5.

Table 8 compares numerical ranges with and without inverse hyperbolic sine (arsinh) amplitude compression. The arsinh configuration yields an MAE of 0.03; the effects of codebook and vocabulary size are reported separately in Table 7.

Table 8: Reconstruction error under different input ranges and amplitude transforms. Improvement is the MAE reduction relative to the first untransformed configuration.

Method	Mapped Range	Representable Range	MAE	vs. first row
Base (No Opt.)	$-100 \sim 100$	$\pm 10^2$	5.13	-
Base (No Opt.)	$-1000 \sim 1,000$	$\pm 10^3$	1.41	72.5%
Arsinh Trans.	$\mathbf{-20 \sim 20}$	$\mathbf{\approx \pm 2 \times 10^8}$	**0.03**	**99.4%**

A.4 Evaluation Metrics for Time-Series Understanding

Time-series understanding is evaluated using lexical overlap, rule-based attribute matching, and an LLM-as-a-judge score.

A.4.1 Objective Textual Metrics

We use BLEU to measure n-gram overlap between generated descriptions and references, reporting two variants:

- **Original BLEU:** Evaluates the complete output including numerical values.
- **BLEU (w/o num):** Evaluates descriptions after removing numerical tokens from both the generated and reference texts.

For a candidate description c and reference r, BLEU is computed as

$$\text{BLEU} = \text{BP} \exp\left(\sum_{n=1}^{N} w_n \log p_n\right), \qquad \text{BP} = \begin{cases} 1, & |c| > |r|, \\ \exp(1 - |r|/|c|), & |c| \le |r|, \end{cases} \tag{9}$$

where p_n is the clipped n-gram precision, w_n are the n-gram weights with $\sum_n w_n = 1$, and $|c|$ and $|r|$ are candidate and effective reference lengths. BLEU (w/o num) applies the same computation after removing numerical tokens from both the candidate and the reference; the original BLEU retains them.

A.4.2 Domain-Specific Rule-Based Evaluation

Because BLEU does not directly measure temporal-attribute accuracy, we additionally use rule-based attribute evaluation inspired by ChatTS (Xie et al., 2025). For a set of K evaluated descriptions, let $q_i \in [0, 1]$ be the keyword-match score for the target temporal attributes in example i; then the table's Regex Keyword score is $\mathrm{RegexKeyword} = K^{-1} \sum_{i=1}^{K} q_i$. The keyword score depends on task-specific matching rules. The evaluation comprises:

1. **Block-wise Localization:** Identifying specific segments of the response describing *trend*, *periodicity*, and *residual noise* using predefined keywords.
2. **Qualitative Evaluation:** Assessing the correctness of categorical descriptions (e.g., "upward trend", "sinusoidal oscillation") via keyword matching.
3. **Quantitative Numerical Score:** Numerical features (e.g., amplitude, mean, standard deviation) are extracted using regular expressions. To keep the relative-error score defined when a target is zero, we use a denominator floor of 10^{-6}:

$$Score = \max\left(0, \min\left(1, 1 - \left|\frac{V_{output} - V_{target}}{\max(|V_{target}|, 10^{-6})}\right|\right)\right). \tag{10}$$

When $V_{target} = 0$, the denominator is set to 10^{-6}; deviations of at least 10^{-6} then receive a score of zero. If regular-expression extraction fails for an evaluated numerical attribute, that attribute receives a score of zero and remains in the average over evaluated numerical attributes. The denominator floor is specific to this evaluation rather than prescribed by ChatTS.

A.4.3 LLM-as-a-Judge

GPT-5.4 assesses each generated description against the reference and ground-truth metadata on a 100-point rubric comprising six dimensions.

- **Trend Accuracy (20 pts):** Correct identification of overall direction (e.g., rising, falling, stable).
- **Periodicity Accuracy (20 pts):** Precision in describing cycle length, amplitude, and waveform.
- **Local Characteristics (20 pts):** Accuracy of local features such as peaks, valleys, and change points.
- **Noise Level (10 pts):** Correct qualitative description of variance and stochasticity.
- **Statistical Features (20 pts):** Accuracy in describing mean, standard deviation, maximum, and minimum values.
- **Numerical Precision (10 pts):** Overall exactness of key figures provided in the text.

For each example i, the reported LLM-Judge score combines the six rubric subscores as $J_i = J_{i,\mathrm{trend}} + J_{i,\mathrm{period}} + J_{i,\mathrm{local}} + J_{i,\mathrm{noise}} + J_{i,\mathrm{stats}} + J_{i,\mathrm{numeric}}$ (maximum $20+20+20+10+20+10 = 100$); the result in the table is $K^{-1} \sum_{i=1}^{K} J_i$. The individual subscores are assigned by the judge using the rubric rather than computed by a deterministic analytic formula.

The evaluation prompt is:

```
 You are a time series analysis expert.  Please evaluate
the quality of the model's description of the time series by
comparing it with the ground truth description.

Original time series data (length {len(timeseries)}):
```

```
{timeseries}

Ground truth description:
{target_description}

Model's description:
{model_description}

Evaluation criteria (100 points total):
1. Trend accuracy (20 points): Does the description
correctly identify the overall trend (rising, falling,
stable, etc.)?
2. Periodicity accuracy (20 points): Does the description
correctly describe periodic patterns (cycle length,
amplitude, waveform)?
3. Local characteristics accuracy (20 points): Does
the description correctly identify local features (peaks,
valleys, change points)?
4. Noise description accuracy (10 points): Is the noise
level correctly described?
5. Statistics accuracy (20 points): Are the statistical
features correctly described (mean, std, max, min)?
6. Numerical accuracy (10 points): Are key numerical
values accurate?

Output format (JSON only, no additional text):
{
"trend_score": int, "trend_reasoning": str,
"periodicity_score": int, "periodicity_reasoning": str,
"local_features_score": int, "local_features_reasoning":
str,
"noise_score": int, "noise_reasoning": str,
"statistics_score": int, "statistics_reasoning": str,
"numerical_score": int, "numerical_reasoning": str,
"total_score": int
}

Requirements:
1. Each sub-score must be an integer between 0 and the
maximum stated for its dimension; the total score must be
between 0 and 100
2. Sum of all sub-scores must equal the total score
3. Provide specific reasoning for each aspect
4. Output JSON only, no other content
```

Together, the lexical, rule-based, and judge-based scores assess complementary aspects of descriptive accuracy.

A.5 Point-Forecasting Evaluation Metrics

For Experiment A, let $y_{i,t}$ and $\hat{y}_{i,t}$ denote the observed and point-forecast values at evaluated time step t in series i, and let N be the total number of evaluated points. The table reports the mean absolute error and mean squared error in standardized space:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i,t}|y_{i,t} - \hat{y}_{i,t}|, \qquad \mathrm{MSE} = \frac{1}{N}\sum_{i,t}(y_{i,t} - \hat{y}_{i,t})^2. \tag{11}$$

The same definitions apply to the ETTh1 horizon and forecast-period context-detail comparisons in Appendix A.7. Reconstruction MAE in Appendix A.3 uses original versus reconstructed values in place of forecast versus target values; MAE-NZ averages absolute error only over the nonzero target positions $\mathcal{N}$, namely $\mathrm{MAE\text{-}NZ} = |\mathcal{N}|^{-1}\sum_{(i,t)\in\mathcal{N}}|y_{i,t} - \hat{y}_{i,t}|$ when $\mathcal{N}$ is nonempty.

Experiment B uses the native M4 per-series seasonal frequency m_i and forecast horizon H_i. For series i with training observations $y_{i,1:T_i}$, forecasts $\hat{y}_{i,T_i+1:T_i+H_i}$, and a seasonal-naive scaling denominator,

the scores are

$$\begin{aligned} \mathrm{sMAPE}_i &= \frac{200}{H_i} \sum_{h=1}^{H_i} \frac{|y_{i,T_i+h} - \hat{y}_{i,T_i+h}|}{|y_{i,T_i+h}| + |\hat{y}_{i,T_i+h}|}, \\ \mathrm{MASE}_i &= \frac{H_i^{-1} \sum_{h=1}^{H_i} |y_{i,T_i+h} - \hat{y}_{i,T_i+h}|}{\max\Big((T_i - m_i)^{-1} \sum_{t=m_i+1}^{T_i} |y_{i,t} - y_{i,t-m_i}|,\, 10^{-8}\Big)}. \end{aligned} \tag{12}$$

The sMAPE summand is set to zero when both numerator and denominator vanish. In our M4 evaluation implementation, the seasonal-naive MASE scaling denominator is floored at 10^{-8} to avoid division by zero. Within a group g, reported sMAPE_g and MASE_g aggregate the corresponding per-series scores. If $\mathrm{sMAPE}_g^{\mathrm{N2}}$ and $\mathrm{MASE}_g^{\mathrm{N2}}$ denote the M4 Naive2 reference scores, the overall weighted average is

$$\mathrm{OWA}_g = \tfrac{1}{2} \left(\frac{\mathrm{sMAPE}_g}{\mathrm{sMAPE}_g^{\mathrm{N2}}} + \frac{\mathrm{MASE}_g}{\mathrm{MASE}_g^{\mathrm{N2}}} \right). \tag{13}$$

For the weighted-average column, sMAPE and MASE group means are weighted by the number of series in each M4 group. Weighted-average OWA is the corresponding series-count-weighted mean of the six group-level OWA scores. Lower values indicate better forecasts for all reported forecasting metrics.

A.6 Benchmark Dataset Statistics

Tables 9 and 10 summarize the Experiment A datasets (including ETT) and the M4 frequency groups, respectively. In both benchmarks, the series are globally standardized with statistics fitted on the respective training partitions. During training, each sample is additionally normalized using the mean and standard deviation of its observed input history; the same history-only transformation is used for validation and test samples, so target observations do not determine the scaling parameters. Full forecasting results, including LLM-SFT and VQ-LLMTS-Reg, are presented in Tables 2 and 3.

Table 9: Experiment A dataset statistics and configurations. Train, Validation, and Test give the numbers of forecasting windows; history length is 96 and the evaluation horizon is fixed at 28.

Dataset	Freq.	Rows	Channels	Target	Split	Train	Val.	Test
ETTh1	1h	17,420	7	OT	ETT official	8,517	2,853	2,853
ETTh2	1h	17,420	7	OT	ETT official	8,517	2,853	2,853
ETTm1	15m	69,680	7	OT	ETT official	34,437	11,493	11,493
ETTm2	15m	69,680	7	OT	ETT official	34,437	11,493	11,493
Weather	10m	52,696	21	OT	70/10/20	36,764	5,242	10,513
Electricity	1h	26,304	321	156	70/10/20	18,289	2,604	5,234
Traffic	1h	17,544	862	260	70/10/20	12,157	1,728	3,482
ILI	1w	966	7	ILITOTAL	70/10/20	553	69	167

Table 10: Statistics of the M4 native univariate benchmark (Experiment B). Each sample is an independent univariate series, following the official M4 protocol with sMAPE, MASE, and OWA as the evaluation metrics.

Group	Freq.	Series	History length (min/med/max)	Input	Test horizon
Yearly	Year	23,000	13 / 29 / 835	12	6
Quarterly	Quarter	24,000	16 / 88 / 866	16	8
Monthly	Month	48,000	42 / 202 / 2,794	36	18
Weekly	Week	359	80 / 934 / 2,597	26	13
Daily	Day	4,227	93 / 2,940 / 9,919	28	14
Hourly	Hour	414	700 / 960 / 960	96	48
Total	—	100,000	—	—	—

Table 11: ETTh1 horizon sensitivity (MAE / MSE; lower is better). The 28-step LLM-SFT and VQ-LLMTS-Reg entries match Experiment A; their longer-horizon entries are from independently trained variants with the original combined context. The VQ-LLMTS row and the 15 baseline rows come from the previously reported horizon study. Red and blue-underlined values denote the lowest and second-lowest distinct scores across all 18 methods for each horizon and metric; ties at the displayed precision share a rank.

Method	**Horizon 28** MAE / MSE	**Horizon 56** MAE / MSE	**Horizon 84** MAE / MSE	**Horizon 112** MAE / MSE
VQ-LLMTS	0.1180 / 0.0243	0.1689 / 0.0551	0.2082 / 0.0818	0.2297 / 0.0893
Time-LLM	0.1228 / 0.0258	0.1644 / 0.0465	0.1794 / 0.0556	0.1944 / 0.0641
GPT4TS	0.1158 / 0.0235	0.1593 / 0.0430	0.1779 / 0.0542	0.1927 / 0.0625
DLinear	0.1209 / 0.0254	0.1681 / 0.0506	0.1803 / 0.0591	0.1891 / 0.0627
PatchTST	0.1187 / 0.0247	0.1581 / 0.0430	0.1746 / 0.0535	0.1868 / 0.0611
TimesNet	0.1216 / 0.0257	0.1622 / 0.0445	0.1783 / 0.0548	0.1885 / 0.0616
FEDformer	0.1323 / 0.0287	0.1743 / 0.0486	0.1808 / 0.0556	0.1981 / 0.0643
Autoformer	0.1764 / 0.0504	0.2078 / 0.0695	0.2086 / 0.0712	0.2292 / 0.0824
Nonstat.	0.1300 / 0.0296	0.1728 / 0.0518	0.1929 / 0.0644	0.1884 / 0.0609
ETSformer	0.1497 / 0.0365	0.2929 / 0.1235	0.4986 / 0.3114	0.3866 / 0.2035
LightTS	0.1307 / 0.0311	0.1670 / 0.0482	0.2121 / 0.0781	0.2090 / 0.0775
Informer	0.2312 / 0.0834	0.3904 / 0.2054	0.3590 / 0.1806	0.4865 / 0.3092
Reformer	0.2387 / 0.0916	0.4105 / 0.2396	0.4527 / 0.2747	0.5176 / 0.3491
Chronos	0.1464 / 0.0375	0.3074 / 0.1477	0.2050 / 0.0745	0.3443 / 0.1792
Timer	0.1692 / 0.0465	0.1863 / 0.0577	0.1952 / 0.0645	0.2030 / 0.0694
S^2IP-LLM	0.1165 / 0.0237	0.1611 / 0.0447	0.1752 / 0.0531	0.1845 / 0.0581
LLM-SFT	0.1593 / 0.0452	0.1737 / 0.0529	0.1831 / 0.0571	0.2339 / 0.0858
VQ-LLMTS-Reg	0.1087 / 0.0201	0.1363 / 0.0323	0.1580 / 0.0443	0.1662 / 0.0475

A.7 Additional Forecasting Experiments

A.7.1 Effect of Forecasting Horizon

We evaluate VQ-LLMTS and 15 forecasting baselines on ETTh1 at output horizons of 28, 56, 84, and 112 steps, and additionally report independently trained LLM-SFT and VQ-LLMTS-Reg variants. Table 11 reports MAE / MSE for each method and horizon. At 28 steps, the LLM-SFT and VQ-LLMTS-Reg entries use the Experiment A values. The original VQ-LLMTS is competitive at 28 steps but is outperformed by several baselines at longer horizons. Red and blue-underlined entries identify the lowest and second-lowest distinct values across all methods at each horizon and metric.

A.7.2 Effect of Forecast-Period Context Detail

We examine five levels of forecast-period context on ETTh1: no description (L0), overall direction (L1), direction, shape, and coarse extrema positions (L2), coarse level transitions (L3), and six-segment relative levels with exact extrema time-step positions (L4). The L4 descriptions do not disclose numerical target values, but do reveal the indices of future extrema. For one 28-step ETTh1 test window, every level retains the same history-derived context, beginning "History-derived context: the full lookback shows a broadly stable level." The following excerpts reproduce its generated future-derived text; each level adds to the previous one:

L0. No future-derived context (history-derived context only).

L1. "Future-derived context: the forecast horizon fell overall."

L2. "Future-derived context: the forecast horizon fell overall, with noticeable fluctuations. The minimum occurred late in the horizon; the maximum occurred midway in the horizon."

L3. L2 plus "The average level from the early to middle third rose; from the middle to late third it fell."

L4. L3 plus "Across six consecutive equal-length segments, their mean levels relative to the horizon average are below, below, above, above, near, well below. Their within-segment directions are stayed near the same level, stayed near the same level, rose, stayed near the same level, fell, stayed near the same level. The minimum is at forecast position 28; the maximum is at forecast position 15."

These excerpts are part of the qualitative prompt, not a numerical target sequence. At evaluation time, the L1–L4 descriptions are generated from the observed forecast window and supplied as

Table 12: ETTh1 forecast-period context detail (MAE / MSE; lower is better). L0 has no future-derived description; L1–L4 use descriptions derived from the target window but omit its numerical values. The previously reported VQ-LLMTS series is shown alongside independently trained LLM-SFT and VQ-LLMTS-Reg variants. The L2 entries use the Experiment A results for direct comparison. Red and blue-underlined values denote the lowest and second-lowest MAE and MSE across the three models at each context level.

Level	Future-derived context	LLM-SFT MAE / MSE	VQ-LLMTS MAE / MSE	VQ-LLMTS-Reg MAE / MSE
L0	None	0.1642 / 0.0480	0.1356 / 0.0354	0.1306 / 0.0306
L1	Overall direction only	0.1609 / 0.0455	0.1269 / 0.0322	0.1199 / 0.0253
L2	Direction, shape, and extrema positions	0.1593 / 0.0452	0.1180 / 0.0243	0.1087 / 0.0201
L3	L2 plus early/middle/late level shifts	0.1557 / 0.0433	0.1129 / 0.0232	0.1023 / 0.0181
L4	Six-segment qualitative levels/directions; exact extremum indices	0.1535 / 0.0427	0.1065 / 0.0198	0.0886 / 0.0137

qualitative inputs without exposing the target-value sequence. Table 12 compares VQ-LLMTS with independently trained LLM-SFT and VQ-LLMTS-Reg variants at each level. The original VQ-LLMTS result decreases in MSE from 0.0354 to 0.0198 (44.1%) and in MAE from 0.1356 to 0.1065 (21.5%) between L0 and L4. The additional LLM-SFT and VQ-LLMTS-Reg results also improve with more detailed descriptions, with VQ-LLMTS-Reg showing the largest change from L0 to L4 (MAE 0.1306 to 0.0886). The L2 entries use the Experiment A values for direct comparison. These comparisons illustrate sensitivity to target-derived text within a privileged-information setting, not history-only forecasting performance or a fully input-matched causal estimate of text utilization.

A.8 CiK Evaluation Metrics

Table 13 illustrates CiK context types and representative scenarios; it does not enumerate individual tasks.

Table 13: Qualitative overview of context types and task scenarios in the Context is Key (CiK) benchmark (Williams et al., 2025). The categories are non-exclusive and are illustrative, not an exhaustive task-by-task inventory. The official benchmark contains 71 task definitions and 355 test instances.

Context type	How context is used	Illustrative scenarios	Task / test count
Historical	Retrieve or reason over information from before the forecast origin	Historical events and patterns; causal effects and confounders in narrative history; target localization by analogy.	–
Future	Apply information or constraints relevant to the forecast period	Electricity and traffic changes; news-informed forecasts; ATM cash/closure scenarios; sensor maintenance and value constraints; regional unemployment forecasts.	–
Intemporal	Use background knowledge not anchored solely to a past/future time point	Solar irradiance from location, zenith, or cloud conditions; traffic with holiday context; speed/pressure relationships; causal and analogy reasoning.	–
Whole benchmark	Forecast using essential textual information	The benchmark spans instruction following, context/memory retrieval, and analogy, deduction, mathematical, and causal reasoning.	**71 / 355**

Table 4 reports capped Reweighted CRPS (RCRPS), overall CRPS, RoI CRPS, Non-RoI CRPS, and Constraint CRPS. We define these metrics and the finite-sample CRPS estimator below. Let $\widetilde{X}$ be a predictive distribution and x the realized value. The Continuous Ranked Probability Score (CRPS) is

$$\mathrm{CRPS}(\widetilde{X}, x) = \mathbb{E}_{X\sim\widetilde{X}}|X - x| - \tfrac{1}{2}\mathbb{E}_{X,X'\sim\widetilde{X}}|X - X'|. \tag{14}$$

For a forecast horizon F, let $\mathcal{I} \subseteq F$ be the time steps designated as the region of interest (RoI) by the task context, and let $F \setminus \mathcal{I}$ be the Non-RoI steps. RoI CRPS and Non-RoI CRPS are the mean per-step CRPS over the corresponding subsets:

$$\text{CRPS}_{\text{RoI}} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \text{CRPS}(\widetilde{X}_i, x_i), \qquad \text{CRPS}_{\text{Non-RoI}} = \frac{1}{|F \setminus \mathcal{I}|} \sum_{i \in F \setminus \mathcal{I}} \text{CRPS}(\widetilde{X}_i, x_i). \tag{15}$$

The RCRPS balances performance inside and outside the context-specified RoI and penalizes violations of value constraints. Let $\mathcal{C}$ be the constraint set and $v_{\mathcal{C}}(\widetilde{\mathbf{X}}_F)$ the predictive distribution of the violation amount (zero denotes no violation). For instances with nonempty RoI and non-RoI regions, the score is

$$\begin{aligned} \text{RCRPS}(\widetilde{\mathbf{X}}_F, \mathbf{x}_F) = \alpha \Big[& \frac{1}{2|\mathcal{I}|} \sum_{i \in \mathcal{I}} \text{CRPS}(\widetilde{X}_i, x_i) + \frac{1}{2|F \setminus \mathcal{I}|} \sum_{i \in F \setminus \mathcal{I}} \text{CRPS}(\widetilde{X}_i, x_i) \\ & + \beta \, \text{CRPS}\big(v_{\mathcal{C}}(\widetilde{\mathbf{X}}_F), 0\big) \Big], \end{aligned} \tag{16}$$

where $\beta = 10$ weights constraint violations and α normalizes for the scale of each task (Williams et al., 2025). The latter is estimated from $M_{\text{task}} = 25$ additional instances per task that are not among the five evaluated instances per task (355 in total). It is the inverse of their mean observed forecast-window range,

$$\alpha = \left[\frac{1}{M_{\text{task}}} \sum_{m=1}^{M_{\text{task}}} \left(\max_{i \in F} x_i^m - \min_{i \in F} x_i^m \right) \right]^{-1}. \tag{17}$$

The table's constraint CRPS is the constraint penalty score $\text{CRPS}_{\text{constraint}} = \text{CRPS}(v_{\mathcal{C}}(\widetilde{\mathbf{X}}_F), 0)$; RCRPS includes this term weighted by β. Overall CRPS is the mean per-step CRPS over the full forecast horizon,

$$\text{CRPS}_{\text{overall}} = \frac{1}{|F|} \sum_{i \in F} \text{CRPS}(\widetilde{X}_i, x_i). \tag{18}$$

The reported primary metric is capped RCRPS: for K evaluated instances with instance-level scores $r_1, \ldots, r_K$, the reported score is $K^{-1} \sum_{k=1}^{K} \min(r_k, 5)$. For VQ-LLMTS, the CiK evaluation pairs context-on and context-off predictions by task type, official seed, and window index across 355 instances (five per task for 71 tasks). Each instance has five sampled trajectories; the archived inference configuration uses temperature 0.8, top-p 0.95, and base seed 2025. These generation settings are specific to VQ-LLMTS, not all compared methods. A paired task-cluster bootstrap of the capped, officially weighted instance scores resamples 71 tasks with replacement, retaining their five paired instances per draw (10,000 replicates; NumPy generator seed 20260925). The context-off minus context-on improvement is 0.008939 with a 95% percentile bootstrap interval $[-0.010408,\ 0.025045]$; the interval includes zero, so the context-on improvement is not established as significant at the 95% level. For a predictive distribution represented by $M > 1$ sorted samples $x_{(1)} \leq \cdots \leq x_{(M)}$, the unbiased pairwise-difference estimator of CRPS is

$$\widehat{\text{CRPS}} = \frac{1}{M} \sum_{n=1}^{M} |x_{(n)} - x| + \frac{1}{M} \sum_{n=1}^{M} x_{(n)} - \frac{2}{M(M-1)} \sum_{n=1}^{M} (n-1) x_{(n)}. \tag{19}$$

A.9 Single-Example Inference Profiling

We profile the Qwen2.5-1.5B-Instruct hybrid-embedding model with its trained LoRA adapter and RVQ-VAE decoder on the server described in Appendix A.1.4, with all four A100-80GB GPUs participating in inference. The script processes the first 100 CiK test prompts with batch size 1 and at most 128 generated tokens per prompt. It uses bf16 model weights, automatic device placement, the chat template, greedy next-token selection, and early stopping at EOS. The average time to first token (TTFT), including the full-prompt forward pass and first-token selection, is $0.1578\,\text{s}$. The average time per output token (TPOT), computed as CUDA-synchronized decode time divided by the number of subsequently generated tokens (or zero if none are generated), is $0.0642\,\text{s/token}$.

After loading both the RVQ-VAE and language model, the total allocated memory across visible GPUs is $3.82\,\text{GB}$. The average estimated prefill overhead is $1.47\,\text{GB}$, calculated as

$\max(\text{summed prefill peak allocation} - \text{loaded-model allocation}, 0)$; it can include intermediate activations as well as KV-cache tensors and is not a direct measurement of KV-cache memory. The reported peak GPU memory is $5.50\,\text{GB}$, defined as the maximum over prompts of the sum of per-device peak allocations, which need not occur simultaneously. The RVQ-VAE decoder contributes to the loaded-model memory, but the timed forward and decode loops call only the language model. Prompt-length statistics are not recorded, and these figures characterize this inference setup rather than training memory or hardware-normalized performance of other configurations.