Continuity and Ordinality Matter: Constraining Time Series Tokens for Effective Time Series Analysis with Large Language Models

Anonymous ACL submission

Abstract

Token-based time series large language models (TS-LLMs) have emerged as a promising direction for time series analysis and reasoning. However, prior studies largely overlook the inherent continuity and ordinality of time series tokens, which substantially limits model performance. In this paper, we argue that preserving these properties in time series token embeddings is crucial for the effectiveness of token-based TS-LLMs. To this end, we propose **COM** (*Continuity and Ordinality Matter*), a continuity- and ordinality-aware strategy that integrates geometric constraints into both the initialization and training stages. Empirical results on multiple time series analysis benchmarks demonstrate that COM consistently improves the performance of token-based TS-LLMs, achieving competitive results and strong generalizability. Code is available at `https://anonymous.4open.science/r/COM`.

1 Introduction

Time series analysis plays a crucial role in a wide range of real-world applications, including finance, healthcare, traffic, energy, and industrial monitoring (Sezer et al., 2020; Pinto et al., 2021; Li et al., 2015; Zhou et al., 2021; Hewage et al., 2020). Despite its importance, conventional time series modeling is often highly task-specific: different tasks, datasets, and domains typically require separately designed architectures or independently trained models (Liu et al., 2024a; Nie et al., 2023; Zeng et al., 2022; Ramaswamy et al., 2000), which substantially limits their scalability and general applicability. Inspired by the recent success of large language models (LLMs) in unified sequence modeling and general-purpose reasoning, time series large language models (TS-LLMs) have emerged as a promising direction for building more generalizable models for time series analysis (Xie et al., 2025; Wang et al., 2025; Jin et al., 2023; Chow et al., 2024).

Existing TS-LLMs generally fall into four paradigms: text-, vision-, alignment-, and token-based approaches. Text-based methods (Kong et al., 2025) regard time series as plain text. Vision-based methods (Zhuang et al., 2024) visualize time series and leverage vision-language models (VLMs) (Bai et al., 2025). Alignment-based methods (Xie et al., 2025) introduce a time series encoder and align it with the LLM backbone. Token-based methods (Wang et al., 2025; Quinlan et al., 2026; Ansari et al., 2024) quantize numerical values into a finite set of time series tokens and extend the LLM vocabulary accordingly. Among these paradigms, token-based methods are particularly promising for building general-purpose TS-LLMs, as they provide a unified token space for time series and text, scale naturally with long-context LLMs, and largely retain the pretrained capabilities of LLM backbones.

However, existing token-based TS-LLMs overlook the intrinsic continuity and ordinality of time series in both token embedding initialization and subsequent training (Wang et al., 2025). Specifically, newly introduced time series token embeddings are typically initialized with random Gaussian vectors, while the training process lacks explicit constraints to preserve the continuous and ordinal structure among time series tokens. This may result in a fragmented embedding space and distorted time series representations, ultimately limiting the model performance.

To bridge this gap, we propose **COM** *(Continuity and Ordinality Matter)*, a continuity- and ordinality-aware strategy for token-based TS-LLMs. COM explicitly preserves the structural properties of time series tokens at two stages: during initialization, it injects structured priors through controllable TS token embedding initialization; during training, it regularizes the embed-

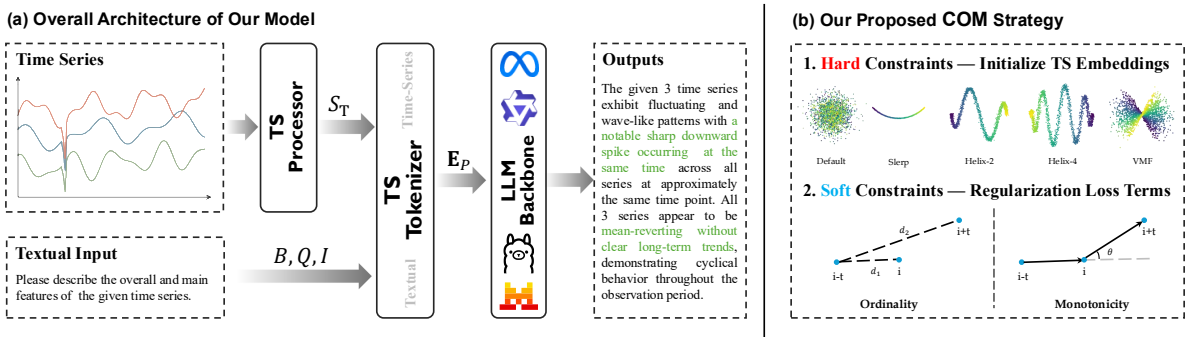


Figure 1: Overall illustration of our work. **(a)** Our approach follows the token-based TS-LLM paradigm, consisting of a TS-Processor, a TS-Tokenizer, and an LLM backbone. **(b)** To preserve the continuity and ordinality of TS tokens, we introduce the **COM** (*Continuity and Ordinality Matter*) strategy, which combines *hard constraints* through embedding initialization with *soft constraints* through dedicated regularization loss terms.

ding geometry with dedicated loss terms. We further design quantitative metrics to evaluate continuity and ordinality in the embedding space. Extensive ablation studies on time series question-answering and reasoning benchmarks verify the effectiveness of these constraints, and additional evaluations on other tasks demonstrate the broader potential of COM.

Empirical results demonstrate that COM consistently improves the performance ceiling and convergence efficiency of TS-LLMs. The performance gains are strongly correlated with the geometric properties of TS token embeddings, highlighting the importance of continuity and ordinality preservation in the embedding space. Across diverse model families and parameter scales, COM maintains robust effectiveness, indicating strong generalizability. Moreover, evaluations on three TSA tasks, namely question-answering, classification, and description, show that COM-enhanced TS-LLMs achieve competitive performance against task-specific baselines, demonstrating their potential for general-purpose TSA.

Our contributions are summarized as follows:

1. We identify the lack of continuity and ordinality modeling as a critical limitation of existing token-based TS-LLMs, and propose COM, a strategy that explicitly preserves these structural properties at both the initialization and training stages.

2. Through comprehensive comparative and ablation studies, we show that COM consistently improves the performance and convergence efficiency of TS-LLMs. We further reveal that such improvements are closely correlated with the continuity and ordinality of TS embeddings, offering insights into the mechanism and design of effective constraints.

3. We achieve competitive performance against specialized models across diverse TSA tasks, demonstrating the generalizability, robustness, and universal potential of COM-enhanced TS-LLMs.

2 Preliminary

In the field of time series large language models (TS-LLMs), diverse time series analysis (TSA) tasks can be unified under a time series question-answering (TSQA) framework. Formally, denote a dataset as $\mathcal{D} = \{(T_i, Q_i, G_i), \mathcal{M}\}_{i=1}^{N}$, where $\mathcal{M}$ represents the task-specific evaluation metric. For the i-th sample, $T_i = \{\mathbf{t}_{i,1}, \ldots, \mathbf{t}_{i,C}\}$ comprises C channels of time series data, where $\mathbf{t}_{i,j} \in \mathbb{R}^{L_j}$ denotes the time series in the j-th channel with an arbitrary length L_j.

The question Q_i is formulated based on the specific TSA task requirements. For instance, in multiple-choice QA, Q_i directs the model to select the most plausible option from a given candidate set; in classification, Q_i instructs the model to categorize the sequence into a class $\mathcal{S}_c$. Accordingly, G_i serves as the ground truth aligned with the Q_i.

Given a TS-LLM denoted by Φ and its generating function $f_\Phi(\cdot)$, the inference process is defined as:

$$A_i = f_\Phi\left(T_i, Q_i, P_i\right) \tag{1}$$

where A_i is the generated response by TS-LLM Φ, and P_i represents the prompt template encompassing task descriptions, context, and operational con-

straints. The overall performance of Φ on dataset $\mathcal{D}$ is evaluated as:

$$\text{Perf.}(\Phi, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{M}(G_i, A_i) \tag{2}$$

In practice, the evaluation metric $\mathcal{M}$ is task-dependent and can be instantiated as accuracy, F1-score, BLEU, ROUGE-L, semantic similarity, MAE, MSE, or other appropriate metrics.

3 Methodology

3.1 Token-Based TS-LLM Architecture

The architecture of our model, as illustrated in Figure 1 (a), follows a token-based TS-LLM paradigm. It consists of three core components: a TS-Processor, a TS-Tokenizer, and a pre-trained LLM backbone.

TS Token Vocabulary and Processing. To bridge the gap between continuous numerical values and discrete linguistic tokens, we expand the LLM's vocabulary with N time series tokens, denoted as $\mathcal{V}_{ts}$. Given a precision ϵ, the number of TS tokens is $N = \lceil 2/\epsilon \rceil + 1$. The TS-Processor transforms the raw time series T into a token sequence S_T through three stages: (1) **Normalization**: The entire time series T is scaled by its global maximum absolute value: $\hat{\mathbf{t}}_j = \frac{\mathbf{t}_j}{\max_{\mathbf{t} \in T}(|\mathbf{t}|)}$, mapping all values into the range $[-1, +1]$. (2) **Quantization**: The normalized values are mapped to the nearest TS tokens in $\mathcal{V}_{ts}$ via nearest-neighbor quantization. (3) **Prompt Construction**: We extract statistical metadata (e.g., min, max, mean, std) and integrate them with the quantized time series using a pre-defined template $\mathcal{T}$. This process is formalized as:

$$S_T = \text{TS-Processor}(T; \epsilon, \mathcal{T}) \tag{3}$$

As shown in Figure 8, our method inherently supports multivariate inputs with different sequence lengths.

Prompt and Tokenization. The generated S_T is then wrapped into a comprehensive prompt P using a task-specific wrapper $\mathcal{W}(\cdot)$:

$$P = \mathcal{W}(S_T, B, Q, I) \tag{4}$$

where B denotes the background information of the task, Q is the specific question, and I represents operational instruction constraints (e.g., output format or option restrictions), or reasoning requirements (e.g., "think step-by-step").

The TS-Tokenizer processes P into a sequence of token IDs, and then maps them into embeddings:

$$\mathbf{E}_P = \text{TS-Tokenizer}(P) \tag{5}$$

where $\mathbf{E}_P \in \mathbb{R}^{L \times D}$ represents the embedding tensor of the sequence P, which consists of textual and TS tokens.

LLM Backbone and Inference. The embeddings $\mathbf{E}_P$ are then fed into the LLM backbone Φ. The model generates the answer A in an autoregressive manner:

$$A = f_\Phi(\mathbf{E}_P) \tag{6}$$

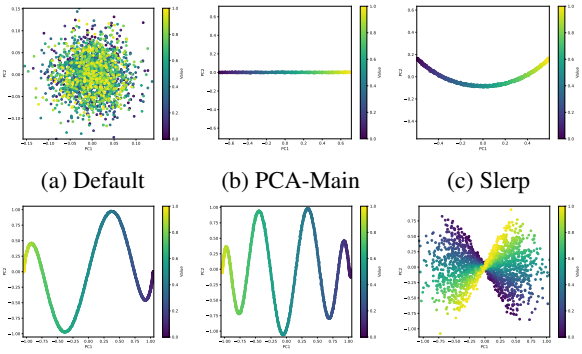
By incorporating TS tokens directly into the vocabulary, Φ can generate sequences that interleave natural language with time series segments. This unified manner enables a single model to handle diverse downstream tasks.

3.2 COM: Continuity- and Ordinality-Aware TS Token Modeling

Existing token-based TS-LLMs typically treat TS tokens as ordinary discrete vocabulary entries (Wang et al., 2025). However, they overlook a fundamental modality discrepancy: unlike linguistic tokens (e.g., "apple", "play"), which lack an inherent notion of order, time series tokens possess intrinsic continuity and ordinality. For instance, "*-0.1, 0.0, 0.1, 0.2*" follows a rigorous increasing order. We argue that enforcing these properties within the embedding space is a prerequisite for effective TSA and superior performance. Interestingly, this argument is further supported by the visualization of TS embeddings in ChatTime (Wang et al., 2025): although no explicit geometric constraints are imposed, the learned embeddings already exhibit emerging continuous and ordered structures, as shown in Figure 9. Consequently, we propose **COM** *(Continuity and Ordinality Matter)*, a continuity- and ordinality-aware strategy for token-based TS-LLMs.

Initialization Stage: Manifold Initialization. To instantiate the continuity and ordinality of TS embeddings $\mathbf{E}_{ts}$, we initialize them as a continuous, ordered manifold rather than via random initialization. This process is governed by two priors: (1) **Distribution Consistency:** newly added TS embeddings should match the mean and variance of existing embeddings, which ensures the LLM backbone identifies TS inputs within a familiar numerical range (Dobler et al., 2025; Chung

et al., 2024). (2) **Low-dimensional Manifold Hypothesis:** since TS tokens represent scalar values, which contain significantly lower information than semantic text, they should reside in a low-dimensional subspace. This is also consistent with recent studies showing that features of image and text modalities tend to concentrate on low-dimensional manifolds (Song and Ermon, 2019; Niu et al., 2021; Wikipedia contributors, 2026). Based on these priors, we design a series of initialization schemes (see Appendix C for details). Figure 2 illustrates the 2D PCA visualization of TS embeddings related to different initialization schemes.



(a) Default (b) PCA-Main (c) Slerp

(d) Helix-2 (e) Helix-4 (f) VMF

Figure 2: 2D PCA visualizations of TS embeddings under different initialization schemes (hard constraints). The color denotes TS values ranging from -1 to $+1$. Details are provided in Appendix C.

Training Stage: Geometry-Preserving Regularization. To dynamically constrain the manifold's geometric properties during training, we introduce two regularization terms:

1) Ordinality Loss ($\mathcal{L}_{ord}$). To preserve ordinal consistency, we encourage the embedding-space distances among TS tokens to reflect their numerical ordering. For a step size k, $\mathbf{e}'_{i-k}$ should be closer to $\mathbf{e}'_i$ than $\mathbf{e}'_{i-2k}$:

$$\mathcal{L}_{\text{ord}} = \frac{1}{N-2k} \sum_{i=2k}^{N} \max(0, \|\mathbf{e}'_i - \mathbf{e}'_{i-k}\|_2 - \|\mathbf{e}'_i - \mathbf{e}'_{i-2k}\|_2 - m_{\text{ord}}) \tag{7}$$

To mitigate the "curse of dimensionality" and preserve meaningful distance semantics, we project the embeddings into a 3D PCA subspace using $\mathbf{P} \in \mathbb{R}^{D\times 3}$, yielding $\mathbf{e}'_i = \mathbf{P}^\top \mathbf{e}_i$.

2) Monotonicity Loss ($\mathcal{L}_{mono}$). To promote smooth embedding trajectories and suppress erratic fluctuations such as zig-zag patterns, we encourage consecutive displacement vectors in the embedding space to be directionally aligned:

$$\mathcal{L}_{\text{mono}} = \frac{1}{N-2k} \sum_{i=k}^{N-k} \max(0, -\text{cos_sim}(\mathbf{e}'_i - \mathbf{e}'_{i-k}, \mathbf{e}'_{i+k} - \mathbf{e}'_i) - m_{\text{mono}}) \tag{8}$$

where m_{ord} and m_{mono} denote controllable margin thresholds in the ordinality and monotonicity losses, respectively. By adjusting the step k, we achieve multi-granularity constraints: a small k constrains local structure, while a large k constrains global structure. We also derive metrics $\mathcal{R}_{\text{ord}}$ and $\mathcal{R}_{\text{mono}}$ by setting margins to 0 to evaluate geometric properties. The overall process is summarized in Algorithm 1.

3.3 Training Objective and Pipeline

Optimization Objective. Following the standard supervised fine-tuning (SFT) paradigm, we optimize the TS-LLM backbone Φ (together with hard-constrained TS embedding weights φ_h) by minimizing the negative log-likelihood of the ground truth answer G, conditioned on the composite prompt P (Equation 4). Let G be a sequence of tokens $\{g_1, g_2, \ldots, g_M\}$. The cross-entropy loss is formulated as:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{M} \sum_{t=1}^{M} \log P_{\Phi,\varphi_h}(g_t \mid P, g_{<t}) \tag{9}$$

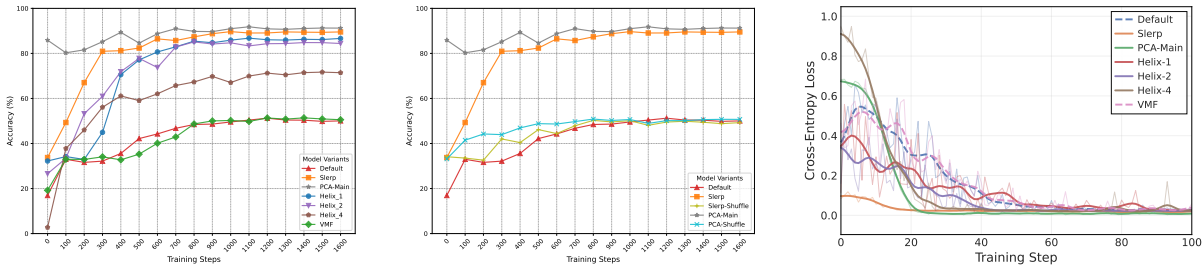
where $g_{<t}$ denotes the preceding tokens in the answer. To enforce the geometric constraints (Sec. 3.2), we incorporate the soft regularization terms into the final objective:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}} + \lambda_{\text{ord}}\mathcal{L}_{\text{ord}} + \lambda_{\text{mono}}\mathcal{L}_{\text{mono}} \tag{10}$$

where λ_{ord} and λ_{mono} are hyperparameters controlling the strength of the ordinality and monotonicity losses.

Training Pipeline. The pipeline consists of two phases:

1) Self-Distillation Alignment. General-purpose LLMs already possess zero-shot capabilities for TSA. A common practice is to distill this knowledge by training smaller models on synthetic data generated by teacher models. However, significant discrepancies in linguistic styles between different model families can introduce distributional shifts, causing the model to overfit to linguistic artifacts rather than learning key TSA patterns. To mitigate this, we propose a self-distillation alignment, utilizing larger models from the same family to synthesize instruction-tuning data. Experiments show



(a) Acc. vs. steps on different variants (b) Ablation on shuffled variants (c) Training loss of different variants

Figure 3: The figure illustrates the accuracy trajectories of different initialized model variants (descriptions provided in Appendix C) over training steps, together with the evolution of cross-entropy loss during training.

this phase effectively activates the model's preliminary TSA capabilities (Appendix I.3).

2) Downstream Fine-Tuning. The aligned model serves as a task-agnostic foundation for subsequent adaptation. In this phase, we perform SFT on varying datasets to enable the model for downstream tasks.

4 Experiments

We conduct a comprehensive evaluation of COM-enhanced token-based TS-LLMs. Our experiments are designed to systematically address the following four research questions (RQs):

RQ1 (Efficacy of Hard Constraints): Does the manifold initialization (hard constraints) improve performance consistently, and what is the underlying explanation?

RQ2 (Mechanism of Soft Constraints): How do the regularization losses (soft constraints) affect different models, what is the relationship between gains and regularization? What are the best practices for controlling margin and granularity?

RQ3 (Generalizability): As a lightweight strategy, can COM be seamlessly integrated into various LLMs to provide consistent performance gains?

RQ4 (Universal Potential): Can our model demonstrate its potential as a universal TSA model by achieving competitive performance across diverse fundamental tasks?

See Appendix E for experimental settings.

4.1 RQ1: Efficacy of Hard Constraints

We systematically investigate the efficacy of manifold initialization (hard constraints) in terms of TS-LLMs' time series modeling capabilities (RQ1) on TSQA (Wang et al., 2025) dataset.

Impact of Geometric Priors. We benchmark five initialization schemes representing distinct geometric priors: *Slerp*, *PCA-Main*, *Helix* (with varying turns k), and *VMF*, utilizing Gaussian random initialization (*Default*) as the baseline.

As illustrated in Figure 3a, the training trajectories reveal a distinct stratification correlated with geometric attributes: **Top Tier** (PCA-Main, Slerp, Helix-1/2) benefits from high continuity and strict ordinality, achieving superior accuracy. **Middle Tier** (Helix-4) exhibits degradation due to increased structural complexity, where excessive spiral turns induce aliasing between tokens of adjacent cycles. Conversely, **Bottom Tier** (Default, VMF), characterized by discontinuity or numerical overlap, fails to effectively capture TS dependencies, resulting in the worst performance. Notably, PCA-Main achieves competitive performance even without training, and ordered initialization variants consistently outperform disordered counterparts in the untrained setting, indicating that the hard constraints could provide a favorable starting point for adaptation. This finding remains consistent on the MMTS-InWild dataset (Table 6).

The Critical Role of Ordinality. To investigate the contribution of spatial ordinality, we conduct an ablation experiment by introducing a *Shuffle* operation to the Slerp and PCA-Main schemes. This operation shuffles the TS token indices while preserving the overall geometric shape of the manifold. As shown in Figure 3b, the shuffled variants suffer a catastrophic performance drop, reverting to the level of the Default baseline. This confirms that geometric shape alone is insufficient; the ordinality of TS tokens along the manifold is decisive for modeling time series. Furthermore, comparing the convergence rates in Figure 3c,

ordered models (solid lines) demonstrate significantly faster convergence compared to disordered counterparts (dashed lines), validating proper hard constraints effectively ease the optimization.

Table 1: Accuracy (%) on TSQA dataset across different initialization variants. * denotes shuffled models. T, S, V, and O represent Trend, Seasonality, Volatility, and Outliers, respectively. Background colors indicate Top, Middle, and Bottom tiers.

VARIANTS	AVG.	T.	S.	V.	O.
PCA-MAIN	**91.23**	98.77	68.39	98.46	99.22
SLERP	89.54	98.69	66.64	95.72	97.04
HELIX1	86.65	97.21	67.14	84.48	98.09
HELIX2	84.40	98.11	67.06	85.77	86.43
HELIX4	71.44	95.15	62.04	56.35	72.35
PCA-MAIN*	50.71	76.50	39.38	39.94	46.78
VMF	50.56	73.21	31.27	44.46	53.22
DEFAULT	50.04	76.83	36.45	41.79	44.70
SLERP*	49.27	76.25	34.87	38.72	47.04

Moreover, Table 1 reveals a performance difference driven by task sensitivity. **Shape-dominant tasks** (Trend, Outliers), which focus on global patterns or obvious anomalies, prove relatively robust to embedding granularity, yielding nontrivial performance even under misordered initialization. Conversely, **Value-sensitive tasks** (Seasonality, Volatility) demand precise modeling of numerical magnitudes and periodic shifts; in this regime, models with continuous and ordered embeddings significantly outperform disordered counterparts, underscoring the necessity of manifold initialization for capturing subtle time series features.

Finding 1: Hard constraints are decisive for TSA: *ordinality* preserves relative ordinal relationships, while *continuity* ensures smooth transitions across adjacent tokens. These constraints optimize the performance, accelerate convergence, and ensure precise numerical modeling.

4.2 RQ2: Mechanism of Soft Constraints

To investigate the underlying mechanisms and optimal practices of soft constraints, we conduct a systematic ablation study of the regularization losses on TSQA dataset. As formulated previously, the regularization consists of *Ordinality Loss* ($\mathcal{L}_{\text{ord}}$) and *Monotonicity Loss* ($\mathcal{L}_{\text{mono}}$). By configuring the steps $s \in \{1, 100\}$, we derive four loss variants: $\mathcal{L}_{\{\text{ord, mono}\}\times\{\text{local, global}\}}$, targeting local and global geometric characteristics, respectively. We evaluate these constraints on two representative model variants: Default and Slerp. Experimental results with varying margins are illustrated in Figure 4.

The results yield several key insights. For the **Default** model, which inherently exhibits high structural disorder, the introduction of global and local ordinality losses significantly boosts performance (up to $\sim 12\%$). Conversely, the monotonicity loss leads to a performance degradation below the baseline. This phenomenon can be attributed to the substantial loss gap stemming from random initialization. The high initial monotonicity penalty likely weakens the effective gradient contribution from the cross-entropy objective, thereby hurting optimization. This observation highlights the sensitivity of the monotonicity loss and suggests that realizing its full benefit may require more carefully calibrated training recipes. In contrast, for the **Slerp** model with favorable geometric properties, the four regularization terms have a marginal impact, with performance fluctuating slightly around the baseline.

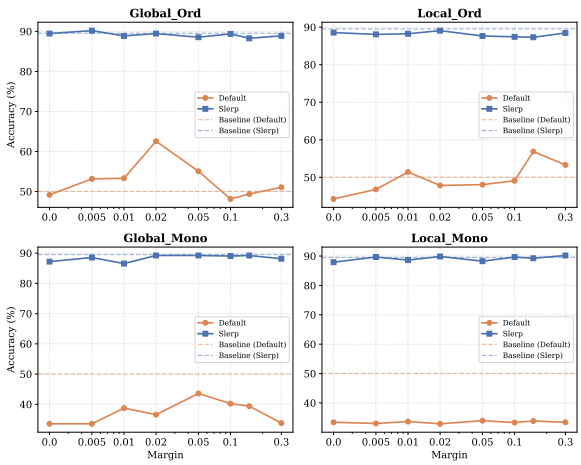


Figure 4: Ablation study of four regularization loss variants on the TSQA dataset. `Ord` and `Mono` denote ordinality and monotonicity losses, while `Local` and `Global` refer to 1 and 100 steps, respectively. The dashed lines represent the baseline performance without soft constraints.

To further quantify the relationship between geomteric attributes and model performance, we design four metrics ($\mathcal{R}_{\{\text{ord, mono}\}\times\{\text{local, global}\}}$) by setting the margin thresholds to 0. These metrics reflect the extent of ordinality and monotonicity in TS embedding space, where higher values indicate poorer geometric consistency. We perform a linear regression between the model performance and the negative logarithm of these metrics. As shown in Figure 5, the performance exhibits a

strong positive correlation with the global metrics (Pearson's $|r| > 0.9$) and a positive trend with the local metrics. This correlation validates the regularization losses as effective proxies for geometric properties and demonstrates a strong alignment with model performance. While other factors (e.g., training epochs, geometric shapes) also affect the performance, this correlation remains consistent and aligns with theoretical expectations.

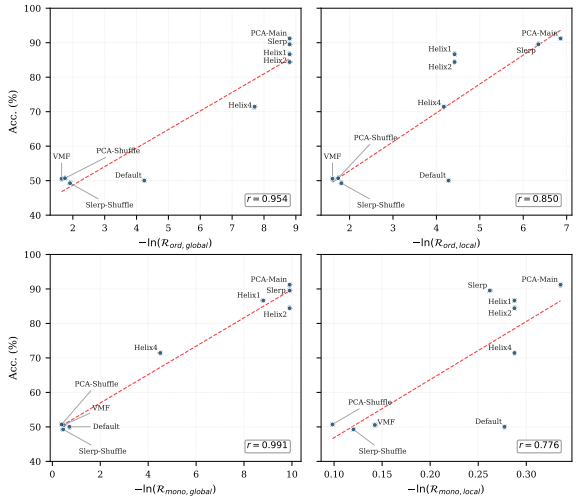


Figure 5: Linear regression analysis between model performance and the negative logarithm of metrics $\mathcal{R}_{\{\text{ord, mono}\}\times\{\text{local, global}\}}$. The consistent positive correlation suggests lower $\mathcal{R}$ (stronger geometric regularity) is predictive of better downstream performance.

Finding 2: Appropriately selected soft constraints $\mathcal{L}$ significantly enhance model performance. The metrics $\mathcal{R}$ serve as reliable proxies for geometric regularity, exhibiting a robust correlation with task performance.

4.3 RQ3: Generalizability of COM

While the efficacy of the COM strategy has been established on the Qwen2.5-3B-Instruct (Qwen et al., 2025) backbone in previous sections, it is imperative to verify its generalizability across different backbones. To this end, we extend our evaluation to LLM backbones of different families and parameter scales, including Llama-3.2-3B-Instruct (Grattafiori et al., 2024), Llama-3.2-1B-Instruct, Gemma-2-2b-it (Team et al., 2024), and Qwen2.5-7B-Instruct (Qwen et al., 2025), while maintaining previous experimental configuration. To streamline the analysis, we focus on comparing the model performance under **Default** and **Slerp** initialization schemes.

Table 2: Accuracy (%) on the TSQA dataset across different LLM backbone families and parameter scales.

Models	Llama-1B		Gemma-2B		Qwen-3B		Llama-3B		Qwen-7B	
	Default	Slerp	Default	Slerp	Default	Slerp	Default	Slerp	Default	Slerp
Avg.	31.46	**89.12**	34.29	**90.38**	50.04	**89.54**	50.65	**90.25**	54.83	**88.60**
T.	31.88	**99.67**	32.87	**99.67**	76.83	**98.69**	76.17	**99.67**	58.92	**99.18**
S.	30.69	**69.90**	35.28	**66.22**	36.45	**66.64**	36.12	**69.57**	51.51	**67.56**
V.	32.66	**88.36**	35.17	**99.51**	41.79	**95.72**	40.42	**94.91**	50.36	**89.81**
O.	30.52	**98.43**	33.83	**95.83**	44.70	**97.04**	49.74	**96.78**	58.78	**98.00**

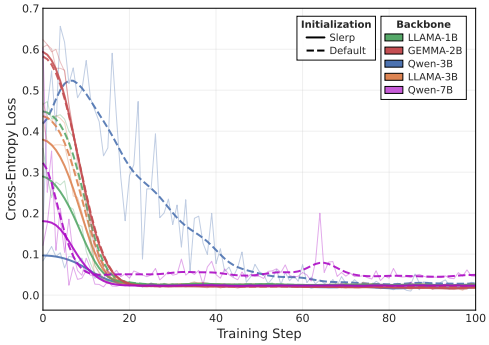


Figure 6: Training loss curves under Default and Slerp initialization schemes across different LLM backbones on TSQA dataset.

As presented in Table 2, the Slerp method yields significant performance improvements over the Default baseline across all evaluated backbones on the TSQA dataset. Furthermore, the performance disparity between shape-dominant tasks and value-sensitive tasks remains consistent across different models, reinforcing the interpretability and efficacy of the COM strategy. In terms of training, the loss curves illustrated in Figure 6 demonstrate that the ordered initialization consistently achieves a higher convergence rate compared to the disordered counterparts, regardless of the backbone (with Gemma as a minor exception). This consistency aligns with the observations reported in Section 4.1. Collectively, these results underscore the robust cross-architecture and cross-scale generalizability of COM, indicating that its benefits are generalizable across model families and parameter scales.

Finding 3: The COM strategy exhibits robust generalizability, consistently achieving performance gains and maintaining interpretability across LLM backbones of different families and parameter scales.

4.4 RQ4: Universal Potential of COM

To investigate the potential of our COM-enhanced TS-LLMs as a universal model for diverse TSA tasks, we evaluate its performance across three

fundamental tasks: question-answering, classification, and description. Following the best practices identified in previous analysis, we benchmark against established domain-specific baselines.

Question-answering (QA). We assess question-answering capabilities on the TSQA dataset. Table 3 presents the performance of various models. The empirical results demonstrate that our model significantly outperforms all baselines (TS-LLMs and general LLMs). Notably, at a 3B parameter scale, our model achieves a superior performance level, surpassing the state-of-the-art ChatTime by over 20%. This underscores the critical role of the COM strategy in augmenting the question-answering capacities of TS-LLMs.

Table 3: Accuracy (%) comparison on the TSQA dataset, where Llama3 (8B), Qwen2.5 (3B), and DeepSeek-V4-Pro represent general LLMs with text-based TS inputs, while ChatTime and ChatTS represent TS-LLMs.

	Ours	ChatTime	ChatTS	DeepSeek-V4	Llama3	Qwen2.5
T.	**99.75**	87.89	82.91	79.59	28.57	35.66
S.	**95.82**	64.68	57.19	42.31	40.38	33.19
V.	**99.43**	45.89	33.06	46.30	38.89	37.75
O.	**99.48**	84.69	74.26	48.84	32.56	31.30
Avg.	**98.62**	76.05	61.58	54.04	35.35	34.54

Classification. The classification capability of our model is evaluated on the RWC dataset (see Appendix G for details). As shown in Table 4, our COM-enhanced token-based TS-LLM achieves the highest accuracy and F1 score compared to task-specific classification baselines. These results indicate that the COM strategy helps effectively capture discriminative features.

Table 4: Classification performance on the RWC dataset. Our model is compared against various task-specific baselines.

Model	Acc.	F1
FormerTime (Cheng et al., 2023a)	0.7803	0.7796
TCN (Bai, 2018)	0.7113	0.7109
MiniROCKET (Dempster et al., 2021)	0.7569	0.7556
TimeMAE (Cheng et al., 2023b)	0.7690	0.7664
TS-TCC (Eldele et al., 2021)	0.6979	0.6931
InstructTime (Cheng et al., 2025)	0.7599	0.7578
Ours (Slerp)	**0.8135**	**0.8153**

Description. We evaluate our model on four subsets of BEDTime, covering both synthetic and real-world scenarios with short and long textual descriptions (see Appendix G for details). As shown in Table 5, our model consistently outperforms larger LLMs, VLMs, and TS-LLMs across a range of description-oriented metrics, including lexical overlap, sequence-level similarity, and semantic consistency. These results demonstrate that the COM strategy effectively enhances the model's ability to generate faithful time series descriptions and generalizes well across different data sources and description granularities.

Table 5: Performance comparison on the BEDTime dataset. The best results are highlighted in bold, and the second-best results are underlined.

Model	BLEU-2	BLEU-4	ROUGE-L	METEOR	DeBERTa F1	SimCSE Cos
Qwen2.5-32B	0.0623	0.0218	0.1624	0.1493	0.5255	0.5623
Qwen3-32B	0.1156	0.0599	0.2316	0.2075	0.5509	0.6066
Llama-3-11B-V	0.0871	0.0524	0.1920	0.1788	0.5269	0.5356
GLM-5.1	0.1222	0.0649	0.2291	0.1909	0.5388	0.5798
DeepSeek-V4-Pro	0.1183	0.0671	0.2425	0.2087	0.5506	0.5870
ChatTS-14B	0.0832	0.0339	0.1794	0.1696	0.5173	0.5190
COM (ours)	**0.2171**	**0.1357**	**0.3662**	**0.3082**	**0.6179**	**0.6388**

Overall, these results across question-answering, classification, and description tasks suggest COM-enhanced TS-LLMs exhibit strong potential as a universal model for diverse TSA scenarios. Rather than being tailored to a single task type, the COM strategy consistently improves understanding, discrimination, and generation within the same token-based modeling paradigm.

Finding 4: Our COM-enhanced TS-LLM exhibits significant universal potential, demonstrating competitive performance across TSA tasks compared to various baselines.

5 Conclusion

In this paper, we address a critical limitation in current token-based TS-LLM paradigms: the neglect of intrinsic continuity and ordinality of time series tokens in the embedding space. To bridge this gap, we propose **COM** *(Continuity and Ordinality Matter),* a continuity- and ordinality-aware strategy designed to enforce these geometric properties through a combination of hard and soft constraints. Experimental results demonstrate that COM substantially enhances time series modeling capabilities, accelerates model convergence, and exhibits robust generalizability across model architectures. Moreover, its competitive performance across diverse TSA tasks highlights its potential as a foundation for large-scale universal TS-LLMs. Beyond time series analysis, our findings may provide useful insights for modeling other forms of intrinsically ordered data.

Limitations

This work has several limitations. First, our study focuses on token-based TS-LLMs, where continuous time series values are quantized into discrete TS tokens and incorporated into the LLM vocabulary. Therefore, our conclusions may not directly transfer to other TS-LLM paradigms, such as alignment-based or vision-based approaches.

Second, although COM introduces geometric priors for TS token embeddings, we do not systematically characterize how different geometric properties affect downstream performance. Our experiments compare several initialization strategies with varying degrees of continuity and ordinality, leaving a more principled geometry–performance analysis for future work.

Third, the effectiveness of COM may depend on tokenization precision, normalization strategy, and regularization hyperparameters. In particular, finer quantization improves numerical resolution but increases vocabulary size, leading to a trade-off between modeling precision and computational cost.

Finally, while our experiments cover representative TSA tasks, the evaluation is still limited to selected benchmarks, model backbones, and data distributions. Real-world time series may involve stronger noise, missing values, irregular sampling, distribution shifts, or domain-specific constraints. Broader evaluations on larger models, longer sequences, and more diverse application domains are needed to further validate the generalizability of COM.

Ethical Considerations

This work focuses on improving token-based TS-LLMs and is evaluated only on research benchmarks. It does not involve collecting personal data, human-subject annotations, or direct deployment in real-world decision-making systems. Therefore, we do not identify immediate ethical concerns in the current study. Nevertheless, since time series data may appear in sensitive domains such as healthcare, finance, and industrial monitoring, future deployment should ensure proper domain validation, privacy protection, and human oversight.

References

Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, and 1 others. 2025. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*.

Anonymous. 2026. MMTS-bench: A comprehensive benchmark for multimodal time series understanding and reasoning.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, and 1 others. 2024. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*.

Shaojie Bai. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Steven Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 interactive presentation sessions*, pages 69–72.

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level bleu. In *Proceedings of the ninth workshop on statistical machine translation*, pages 362–367.

Lifan Chen, Xiaoqin Tan, Dingyan Wang, Feisheng Zhong, Xiaohong Liu, Tianbiao Yang, Xiaomin Luo, Kaixian Chen, Hualiang Jiang, and Mingyue Zheng. 2020. Transformercpi: improving compound–protein interaction prediction by sequence-based deep learning with self-attention mechanism and label reversal experiments. *Bioinformatics*, 36(16):4406–4414.

Mingyue Cheng, Yiheng Chen, Qi Liu, Zhiding Liu, Yucong Luo, and Enhong Chen. 2025. Instructime: Advancing time series classification with multimodal language modeling. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pages 792–800.

Mingyue Cheng, Qi Liu, Zhiding Liu, Zhi Li, Yucong Luo, and Enhong Chen. 2023a. Formertime: Hierarchical multi-scale representations for multivariate time series classification. In *Proceedings of the ACM web conference 2023*, pages 1437–1445.

Mingyue Cheng, Qi Liu, Zhiding Liu, Hao Zhang, Rujiao Zhang, and Enhong Chen. 2023b. Timemae: Self-supervised representations of time series with decoupled masked autoencoders. *arXiv preprint arXiv:2303.00320*.

Winnie Chow, Lauren Gardiner, Haraldur T Hallgrímsson, Maxwell A Xu, and Shirley You Ren. 2024. Towards time series reasoning with llms. *arXiv preprint arXiv:2409.11376*.

Woojin Chung, Jiwoo Hong, Na Min An, James Thorne, and Se-Young Yun. 2024. Stable language model pre-training by reducing embedding variability. *arXiv preprint arXiv:2409.07787*.

Alex O Davies, Roussel Nzoyem, Nirav Ajmeri, and 1 others. 2025. Language models do not embed numbers continuously. *arXiv preprint arXiv:2510.08009*.

DeepSeek-AI. 2026. Deepseek-v4: Towards highly efficient million-token context intelligence.

Angus Dempster, Daniel F Schmidt, and Geoffrey I Webb. 2021. Minirocket: A very fast (almost) deterministic transform for time series classification. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 248–257.

Konstantin Dobler, Desmond Elliott, and Gerard de Melo. 2025. Awedist: Attention-aware embedding distillation for new input token embeddings. *arXiv preprint arXiv:2505.20133*.

Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. 2021. Time-series representation learning via temporal and contextual contrasting. *arXiv preprint arXiv:2106.14112*.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 6894–6910.

Yury Gorishniy, Ivan Rubachev, and Artem Babenko. 2022. On embeddings for numerical features in tabular deep learning. *Advances in Neural Information Processing Systems*, 35:24991–25004.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Huifeng Guo, Bo Chen, Ruiming Tang, Weinan Zhang, Zhenguo Li, and Xiuqiang He. 2021. An embedding learning framework for numerical features in ctr prediction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2910–2918.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Pradeep Hewage, Ardhendu Behera, Marcello Trovati, Ella Pereira, Morteza Ghahremani, Francesco Palmieri, and Yonghuai Liu. 2020. Temporal convolutional neural (tcn) network for an effective weather forecasting using time-series data from the local weather station. *Soft Computing*, 24(21):16453–16482.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and 1 others. 2023. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*.

Yaxuan Kong, Yiyuan Yang, Yoontae Hwang, Wenjie Du, Stefan Zohren, Zhangyang Wang, Ming Jin, and Qingsong Wen. 2025. Time-mqa: Time series multi-task question answering with context enhancement. *arXiv preprint arXiv:2503.01875*.

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

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Huaiyuan Liu, Donghua Yang, Xianzhang Liu, Xinglei Chen, Zhiyu Liang, Hongzhi Wang, Yong Cui, and Jun Gu. 2024a. Todynet: temporal dynamic graph neural network for multivariate time series classification. *Information Sciences*, 677:120914.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. 2024b. itransformer: Inverted transformers are effective for time series forecasting. In *International conference on learning representations*, volume 2024, pages 11116–11140.

KV Mardia. 1975. Distribution theory for the von mises-fisher distribution and its application. In *A Modern Course on Statistical Distributions in Scientific Work: Volume 1Models and Structures Proceedings of the NATO Advanced Study Institute held at the University of Calgagry, Calgary, Alberta, Canada July 29–August 10, 1974*, pages 113–130. Springer.

Marinexplore and Cornell University. 2013. Whale detection challenge dataset. Kaggle. Accessed: 2025-12-01.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Shikai Qiu Nate Gruver, Marc Finzi and Andrew Gordon Wilson. 2023. Large Language Models Are Zero Shot Time Series Forecasters. In *Advances in Neural Information Processing Systems*.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*.

Chuang Niu, Wenxiang Cong, Feng-Lei Fan, Hongming Shan, Mengzhou Li, Jimin Liang, and Ge Wang. 2021. Low-dimensional manifold-constrained disentanglement network for metal artifact reduction. *IEEE transactions on radiation and plasma medical sciences*, 6(6):656–666.

Open-Meteo. 2021. Open-meteo: Free weather api. https://open-meteo.com. Accessed: 2024-05-01.

Feiyang Pan, Shuokai Li, Xiang Ao, Pingzhong Tang, and Qing He. 2019. Warm up cold-start advertisements: Improving ctr predictions via learning to learn id embeddings. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 695–704.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Tiago Pinto, Isabel Praça, Zita Vale, and Jose Silva. 2021. Ensemble learning for electricity consumption forecasting in office buildings. *Neurocomputing*, 423:747–755.

Martin F Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Paul Quinlan, Qingguo Li, and Xiaodan Zhu. 2026. Chat-ts: Enhancing multi-modal reasoning over time-series and natural language data. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5621–5647.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 427–438.

Medhasweta Sen, Zachary Gottesman, Jiaxing Qiu, C Bayan Bruss, Nam Nguyen, and Tom Hartvigsen. 2025. Bedtime: A unified benchmark for automatically describing time series. *arXiv preprint arXiv:2509.05215*.

Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. 2020. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90:106181.

Yang Song and Stefano Ermon. 2019. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Chengsen Wang, Qi Qi, Jingyu Wang, Haifeng Sun, Zirui Zhuang, Jinming Wu, Lei Zhang, and Jianxin Liao. 2025. Chattime: A unified multimodal time series foundation model bridging numerical and textual data. In *AAAI Conference on Artificial Intelligence*.

Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Zhang, and Jun Zhou. 2024a. Timemixer: Decomposable multiscale mixing for time series forecasting. In *International conference on learning representations*, volume 2024, pages 38626–38652.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Yong Liu, Chen Wang, Mingsheng Long, and Jianmin Wang. 2026. Deep time series models: A comprehensive survey and benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qiu, Jianmin Wang, and Mingsheng Long. 2024b. Timexer: Empowering transformers for time series forecasting with exogenous variables. *Advances in Neural Information Processing Systems*, 37:469–498.

Wikipedia contributors. 2026. Manifold hypothesis. https://en.wikipedia.org/wiki/Manifold_hypothesis. Accessed: 2026-01-27.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2022. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430.

Zhe Xie, Zeyan Li, Xiao He, Longlong Xu, Xidao Wen, Tieying Zhang, Jianjun Chen, Rui Shi, and Dan Pei. 2025. Chatts: Aligning time series with llms via synthetic data for enhanced understanding and reasoning. In *Proceedings of the VLDB Endowment, 2025*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2022. Are transformers effective for time series forecasting? *arXiv preprint arXiv:2205.13504*.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Yunkai Zhang, Yawen Zhang, Ming Zheng, Kezhen Chen, Chongyang Gao, Ruian Ge, Siyuan Teng, Amine Jelloul, Jinmeng Rao, Xiaoyuan Guo, and 1 others. 2023. Insight miner: A large-scale multimodal model for insight mining from time series. In *NeurIPS 2023 AI for Science Workshop*.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, volume 35, pages 11106–11115. AAAI Press.

Jiaxin Zhuang, Leon Yan, Zhenwei Zhang, Ruiqi Wang, Jiawei Zhang, and Yuantao Gu. 2024. See it, think it, sorted: Large multimodal models are few-shot time series anomaly analyzers. *arXiv preprint arXiv:2411.02465*.

A Related Work

A.1 Time Series Large Language Models

TS-LLMs leverage the question-answering and reasoning capabilities of LLMs for time series analysis (TSA) through four primary paradigms. **Text-based** methods (e.g., Time-MQA (Kong et al., 2025), LLM-Time (Nate Gruver and Wilson, 2023)) convert time series data into plain-text sequences, but often suffer from high amout of tokens and sensitivity to numerical precision. **Vision-based** methods (e.g., TAMA (Zhuang et al., 2024), Insight-Miner (Zhang et al., 2023)) plot time series as images to utilize vision-language models, though they are frequently constrained by image resolution and the high cost of visual tokens. **Alignment-based** methods (e.g., ChatTS (Xie et al., 2025), (Chow et al., 2024)) employ dedicated encoders to map temporal features into the LLM embedding space; however, their reliance on patch-based representations can obscure fine-grained local structures. **Token-based** methods (e.g., ChatTime (Wang et al., 2025; Quinlan et al., 2026; Ansari et al., 2024)) discretize values into an extended vocabulary, enabling efficient, unified modeling of text and time series. While promising, existing token-based methods typically treat TS tokens as normal vocabulary entries, overlooking their intrinsic continuity and ordinality. To address this, our COM strategy explicitly enforces these properties within the embedding space to better align with time series characteristics.

A.2 Numerical and Continuous Embeddings

Embedding techniques were originally introduced to map sparse, high-dimensional textual tokens into low-dimensional dense representations (Mikolov et al., 2013), and have therefore been particularly effective at modeling discrete concepts such as word pieces (Gage, 1994) and molecules composed of atoms (Chen et al., 2020). However, in other domains (including time series analysis), many concepts exhibit partial orders or continuous measurements. Naively discretizing such concepts often destroys their inherent ordering and continuity. As a result, it is necessary to leverage continuous embedding techniques that preserve these structures. For example, in recommender systems, AutoDis (Guo et al., 2021) models continuous features (e.g., user age and item price) using continuous embeddings based on meta-embeddings and soft bucketization, ensuring that numerically similar values remain close in the embedding space. Similarly, prior work (Gorishniy et al., 2022) extends continuous embedding methods to tabular data and demonstrates performance gains on tabular tasks such as click-through rate (CTR) prediction (Pan et al., 2019). Despite these advances, the application of continuous embeddings to time series data, which also exhibit partial ordering and continuous measurements, remains relatively underexplored in the context of TS-LLMs (Davies et al., 2025).

B Detailed Introduction on TS-LLMs

Time series large language models (TS-LLMs) are built upon large language models (LLMs) and leverage their natural language understanding and reasoning capabilities to address time series analysis tasks. Existing TS-LLMs can be broadly categorized into four paradigms based on how time series are integrated with LLMs: text-based, vision-based, alignment-based, and token-based approaches.

Text-based methods (e.g., Time-MQA (Kong et al., 2025), LLM-Time (Nate Gruver and Wilson, 2023)) convert time series into plain-text sequences and feed them into LLMs, with task-specific prompts used to solve different downstream tasks. While conceptually simple, these methods typically require substantial token overhead and are highly sensitive to the tokenizer design and numerical precision of different LLMs, resulting in limited numerical expressiveness and only preliminary performance.

Vision-based methods (e.g., TAMA (Zhuang et al., 2024), Insight-Miner (Zhang et al., 2023)) represent time series as images and rely on the visual understanding capabilities of vision-language models (VLMs) to perform analysis and reasoning. However, these methods are constrained by image resolution and the high cost of visual tokens, and its effectiveness heavily depends on the inherent ability of VLMs to capture temporal features and patterns.

Alignment-based methods (e.g., ChatTS(Xie et al., 2025) and (Chow et al., 2024)) introduce dedicated time series encoders and align the encoded time series features with the embedding space of LLMs. These methods often employ patch-based representations to reduce the number of input tokens, which limits the modeling of fine-

grained local time series structures and poses challenges for recovering original numerical values required in precise reasoning tasks.

Token-based methods (e.g., ChatTime (Wang et al., 2025; Quinlan et al., 2026; Ansari et al., 2024)c) discretize continuous and unbounded time series values into a finite set of tokens and incorporate them as an extension of the LLM vocabulary, enabling unified sequence modeling of textual and TS tokens. With the rapid expansion of LLM context lengths (e.g., up to 128k tokens) (Abouelenin et al., 2025), treating each numerical value as a token has become increasingly feasible, allowing this paradigm to strike a favorable balance between token efficiency and numerical quantization accuracy. Moreover, the natural interleaving of time series and text tokens offers the flexibility to support diverse time series analysis and reasoning tasks, making token-based approaches a promising direction toward general-purpose TS-LLMs.

C Implementation Details of Different initialization schemes

This appendix provides implementation details for the different initialization schemes used as hard constraints in our work, along with further conceptual explanations and illustrative pseudo-code examples.

Default strategy uses the Gaussian random initialization while expanding vocabulary size with the `model.resize_token_embeddings()` function.

Slerp strategy uses spherical linear interpolation to initialize TS token embeddings along a great-circle path between two randomly sampled points on the hypersphere. Small tangential and radial perturbations are added to increase diversity while preserving the global spherical geometry.

Listing 1: Pseudo-code for the Slerp initialization.

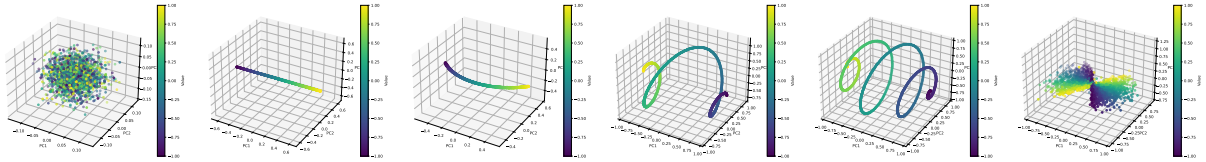
```
# Compute global embedding statistics
mean_embed  <- mean(embed_txt)
avg_radius  <- mean(||embed_txt - mean_embed
    ||)

# Sample start and end points on the
    hypersphere
start_point <- random_unit_vector() *
    avg_radius
end_point   <- random_unit_vector() *
    avg_radius

for token index i from start_idx to end_idx:
    t <- (i - start_idx) / (num_tokens - 1)  #
     normalized position

    # Compute spherical linear interpolation (
    Slerp)
    omega <- arccos(dot(start_point, end_point
    ) / avg_radius^2)
    point_on_sphere <- sin((1 - t) * omega) /
    sin(omega) * start_point +
                          sin(t * omega) /
    sin(omega) * end_point

    # Add tangential and radial perturbations
    radial_dir <- normalize(point_on_sphere)
    tangent_noise <- tangential_noise(
    radial_dir, scale=noise_scale *
                    avg_radius)
    radial_noise  <- radial_noise_magnitude()
    * radial_dir
    final_point   <- point_on_sphere +
    tangent_noise/10 + radial_noise

    # Shift back to global mean
    embed[i] <- mean_embed + final_point
```

PCA-Main strategy uses the first principal component of the existing embeddings to place TS tokens along a line in the principal direction. Small tangential perturbations are added to increase diversity while preserving the global linear order.

Listing 2: Pseudo-code for the PCA-Main initialization.

```
# Compute global embedding statistics
mean_embed <- mean(embed_txt)
centered   <- embed_txt - mean_embed

# Compute principal component via PCA
principal_direction <- top_PCA_component(
    centered)

# Project embeddings onto the principal
    component
projections <- project(embed_txt,
    principal_direction)
(min_proj, max_proj) <- min_max(projections)
margin <- (max_proj - min_proj) *
    predefined_margin
new_min <- min_proj - margin
new_max <- max_proj + margin

for token index i from start_idx to end_idx:
    # Linear placement along principal
    component
    t <- new_min + (i - start_idx) / (
    num_tokens - 1) * (new_max - new_min)
    main_component <- t * principal_direction

    # Add small tangential noise
    noise <- tangential_noise(
    principal_direction, scale=noise_scale)

    # Shift back to global mean
    embed[i] <- mean_embed + main_component +
    noise
```

Helix strategy uses a spherical helical trajectory along the top three principal components of the existing embeddings. TS tokens are placed along the helix with latitude θ and longitude ϕ, where n of



(a) Default (b) PCA-Main (c) Slerp (d) Helix-2 (e) Helix-4 (f) VMF

Figure 7: PCA-based 3D visualizations of TS embeddings under different embedding initialization schemes. The color from blue to yellow denotes time-series value tokens ranging from -1 to $+1$. (a) corresponds to the default Gaussian random initialization, while (b)(f) represent our proposed prior-based initialization schemes.

different variants Helix-n specifies the number of helical turns. Small tangential and radial perturbations are applied to increase diversity while maintaining the overall helical structure.

Listing 3: Pseudo-code for the Helix initialization.

```
# Compute global embedding statistics
mean_embed <- mean(embed_txt)
avg_radius <- mean(||embed_txt - mean_embed||)

# Hyperparameters
num_turns  <- predefined number of helical
     turns
noise_scale <- small perturbation magnitude

# Obtain principal axes via PCA
(axis1, axis2, axis3) <- top 3 PCA components
axis2 <- orthogonalize(axis2, axis1)

for token index i from start_idx to end_idx:
    t <- (i - start_idx) / (num_tokens - 1)  #
      normalized position

    # Spherical angles for helical placement
    theta <- (t - 0.5) * pi
    phi   <- t * num_turns * 2 * pi

    # Compute embedding on the hypersphere
    embedding <- avg_radius * [
        cos(theta) * cos(phi) * axis1 +
        cos(theta) * sin(phi) * axis2 +
        sin(theta) * axis3 ]

    # Add tangential and radial perturbations
    radial_dir <- normalize(embedding)
    tangent_noise <- tangential_noise(
     radial_dir, scale=noise_scale *
                    avg_radius)
    radial_noise  <- radial_noise_magnitude()
     * radial_dir
    final_embed   <- embedding + tangent_noise
      + radial_noise

    # Shift back to global mean
    embed[i] <- mean_embed + final_embed
```

VMF strategy places TS tokens along a spherical trajectory between two principal directions, sampling each token from a von Mises-Fisher (vMF) (Mardia, 1975) distribution around an interpolated mean direction. The sampled vectors are scaled to match the global radius distribution, preserving directional coherence.

Listing 4: Pseudo-code for the VMF initialization.

```
# Compute global embedding statistics
mean_embed <- mean(embed_txt)
centered   <- embed_txt - mean_embed
radius_mean <- mean(||centered||)
radius_std  <- std(||centered||)

# Obtain start and end directions using PCA
start_direction <- top_PCA_component(centered)
end_direction   <- mix(top_PCA_components(
     centered, 2))
end_direction   <- normalize(end_direction)

for token index i from start_idx to end_idx:
    t <- (i - start_idx) / (num_tokens - 1)  #
      normalized position

    # Interpolate mean direction along
     spherical path
    omega <- arccos(dot(start_direction,
     end_direction))
    mu_direction <- sin((1-t)*omega)/sin(omega
     )*start_direction +
                        sin(t*omega)/sin(omega
     )*end_direction
    mu_direction <- normalize(mu_direction)

    # Sample from vMF distribution around
     mu_direction
    sample <- sample_vmf(mu_direction,
     concentration, embed_dim)

    # Adjust to appropriate radius
    target_radius <-
     sample_from_radius_distribution(
     radius_mean,radius_std)
    sample <- sample * target_radius

    # Shift back to global mean
    embed[i] <- mean_embed + sample
```

D Prompts and Templates

This appendix presents implementation details of prompts and templates in our work. Figure 8 illustrates how the TS-Processor converts time series data with heterogeneous lengths and numbers of channels into TS tokens and integrates them into the prompt template (as illustrated in Equation 3), including both the template format and an exam-

ple. The following gray text boxes present the prompt templates used in our experiments on different datasets across tasks.

E Experimental Setup

Training Configuration. As described in Sec. 3.3, our training procedure follows a two-phase pipeline: *self-distillation alignment* followed by *downstream fine-tuning*. For mechanistic and ablation studies of the *COM* strategy (RQ1–RQ3), we primarily use only the second-phase fine-tuning to isolate the effect of the investigated components. The full two-phase pipeline is used for evaluating peak performance and generalizability (RQ4).

By default, we adopt `Qwen2.5-3B-Instruct` as the LLM backbone (Qwen et al., 2025). The precision ϵ of the TS-Processor is set to 0.001, yielding a TS-token vocabulary of $|\mathcal{V}_{\text{ts}}| = 2001$. This vocabulary size is chosen to be compatible with the original vocabulary size of the backbone, i.e., $|\mathcal{V}_{\text{llm}}| = 151{,}936$. For token formatting, we use the wrapper `"<<<{num}>>>"` and adopt `"|"` as the separator, which improves readability and preserves clear token boundaries.

We perform full-parameter fine-tuning with the AdamW optimizer using a learning rate of 1×10^{-5}, a cosine learning-rate scheduler with a warmup ratio of 0.03, and a weight decay of 0.01. All experiments are conducted on a single NVIDIA A800 80GB GPU with a total batch size of 64, implemented as a per-device batch size of 16 with 4 gradient accumulation steps, for one training epoch.

Unless otherwise specified, all main results are reported from a single run with random seed 42. We choose this reporting protocol due to the large computational cost of full-parameter fine-tuning on TS-LLMs across multiple datasets and task settings. To ensure reproducibility, we fix all random seeds, training configurations, data splits, and evaluation protocols throughout the experiments.

To assess potential sensitivity to random initialization and data-ordering effects, we additionally conduct experiments with alternative random seeds on representative settings. The observed trends are generally consistent across seeds. Unless otherwise specified, we report the seed-42 results in the main tables.

Datasets and Tasks. We evaluate our model across a broad spectrum of time series analysis (TSA) tasks. For fundamental time series question answering, we use TSQA (Wang et al., 2025), which covers four basic properties, including Trend, Seasonality, Volatility, and Outliers, over univariate sequences of lengths $\{64, 128, 256, 512\}$. We train on the ChatTime fine-tuning data (Wang et al., 2025) for TSQA-related experiments. For complex time series reasoning, we adopt the MMTS-InWild subset from MMTS-Bench (Anonymous, 2026), which evaluates diverse analysis and reasoning abilities under a hierarchical taxonomy; following the MMTS synthesis protocol, we generate approximately 2.5k fine-tuning samples on heterogeneous time-series data. To further assess general-purpose TSA capabilities, we include the RWC (Marinexplore and University, 2013) dataset for classification, BEDTime (Sen et al., 2025) for time-series description generation, and CGTSF-PTF, ETT (h1, h2, m1, m2), Traffic, and Weather for forecasting. Detailed dataset information is provided in Appendix G, and task modeling and prompt designs are provided in Appendix D.

F Implementation of Soft Constraints

Algorithm 1 presents the pseudo-code details for implementing our regularization losses, which serve as soft constraints in COM.

Algorithm 1 Implementation of regularization losses (soft constraints).

Input: TS embeddings $\mathbf{E}_{ts} \in \mathbb{R}^{N \times D}$, step size k, margins $m_{\text{ord}}, m_{\text{mono}}$
Output: Ordinality loss $\mathcal{L}_{\text{ord}}$, monotonicity loss $\mathcal{L}_{\text{mono}}$
$\mathbf{P} \leftarrow \text{PCA}(\mathbf{E}_{ts}, \text{dim} = 3)$
$\mathbf{E}' \leftarrow \mathbf{E}_{ts}\mathbf{P}$ $\{\mathbf{E}' \in \mathbb{R}^{N \times 3}\}$
Compute Ordinality Loss:
$\mathbf{E}_i \leftarrow \mathbf{E}'[2k:]$ $\{\mathbf{e}'_i\}$
$\mathbf{E}_{i-k} \leftarrow \mathbf{E}'[k:-k]$ $\{\mathbf{e}'_{i-k}\}$
$\mathbf{E}_{i-2k} \leftarrow \mathbf{E}'[:-2k]$ $\{\mathbf{e}'_{i-2k}\}$
$d_{\text{near}} \leftarrow \|\mathbf{E}_i - \mathbf{E}_{i-k}\|_2$
$d_{\text{far}} \leftarrow \|\mathbf{E}_i - \mathbf{E}_{i-2k}\|_2$
$\mathcal{L}_{\text{ord}} \leftarrow \text{mean}\left(\text{ReLU}(d_{\text{near}} - d_{\text{far}} - m_{\text{ord}})\right)$
Compute Monotonicity Loss:
$\mathbf{E}_{i-k} \leftarrow \mathbf{E}'[:-2k]$ $\{\mathbf{e}'_{i-k}\}$
$\mathbf{E}_i \leftarrow \mathbf{E}'[k:-k]$ $\{\mathbf{e}'_i\}$
$\mathbf{E}_{i+k} \leftarrow \mathbf{E}'[2k:]$ $\{\mathbf{e}'_{i+k}\}$
$\Delta_{\text{prev}} \leftarrow \mathbf{E}_i - \mathbf{E}_{i-k}$
$\Delta_{\text{next}} \leftarrow \mathbf{E}_{i+k} - \mathbf{E}_i$
$sim \leftarrow \text{CosineSimilarity}(\Delta_{\text{prev}}, \Delta_{\text{next}})$
$\mathcal{L}_{\text{mono}} \leftarrow \text{mean}\left(\text{ReLU}(-sim - m_{\text{mono}})\right)$
Compute Evaluation Metrics:
$\mathcal{R}_{\text{ord}} \leftarrow \text{mean}\left(\text{ReLU}(d_{\text{near}} - d_{\text{far}})\right)$
$\mathcal{R}_{\text{mono}} \leftarrow \text{mean}\left(\text{ReLU}(-sim)\right)$
Return: $\mathcal{L}_{\text{ord}}, \mathcal{L}_{\text{mono}}, \mathcal{R}_{\text{ord}}, \mathcal{R}_{\text{mono}}$

TS-Processor Template and Example

```
[Template]
Given n time series:
Time series {i} is of length {len}, with statistical information:{max, min, mean, std, left, right, mid}.
...

We normalize all time series values into range [−1.0, +1.0], with the same scale factor {factor}. After scaling, the {n}
time series are below ("{sep}" is the separator between values):
Time series {i} is: {seq_ts_i}.
...

[Example]
Given 3 time series:
Time series 1 is of length 4, with statistical information: {max:0.038, min:-0.964, mean:-0.229, std:0.424, left:0.001,
right:-0.964, mid:0.008}.
Time series 2 is of length 8, with statistical information: {max:1.100, min:-0.964, mean:-0.092, std:0.613, left:0.001,
right:-0.964, mid:1.100}.
Time series 3 is of length 5, with statistical information: {max:0.038, min:-0.964, mean:-0.260, std:0.385, left:0.001,
right:-0.383, mid:0.008}.

We normalize all time series values into range [-1.0, +1.0], with the same scale factor 1.100. After scaling, the 3 time
series are below ("|" is the separator between values):
Time series 1 is: |<0.001>|<0.035>|<0.007>|<−0.876>|.
Time series 2 is: |<0.001>|<0.035>|<0.007>|<−0.876>|<1.000>|<0.035>|<0.007>|<−0.876>|.
Time series 3 is: |<0.001>|<0.035>|<0.007>|<−0.876>|<−0.348>|.
```

Figure 8: Template and example of TS-Processor for time-series processing. The template consists of statistical metadata and quantized TS sequences. In the example, time series values are normalized to $[-1, +1]$ and enclosed by "<" and ">", with "|" used as the separator between TS tokens. The example demonstrates that the proposed method inherently supports multivariate inputs with heterogeneous sequence lengths. The resulting S_T serves as the time series information in the full prompt for TS-LLMs.

Prompt Template Used on TSQA Dataset

```
## Time Series Information:
<Insert the time series context here, e.g., statistical metadata, quantized sequences, etc.>

## Question:
<Insert the question here>

## Options:
<Insert all candidate options here, exactly as listed>

## Instruction:
Output the option letter followed by the full option text exactly as listed in Options. Do not
add explanations or generate any additional content.
```

G Detailed Information on Datasets

This appendix documents the dataset used in our experiments, including their task formats, data modalities, application domains, and language coverage. Overall, the benchmarks cover diverse time series analysis scenarios, including question answering, reasoning, classification, forecasting, description generation, and multimodal reasoning, with textual components primarily in English.

TSQA. The TSQA (Time Series Question Answering) (Wang et al., 2025) dataset is a multimodal multi-choice question dataset introduced to assess the translation of time series data into natural language. Containing four distinct subtasks (Trend, Seasonality, Volatility, and Outliers), TSQA specifically targets the TS-to-text task by requiring models to interpret and identify specific patterns within time series data, such as distinguishing between constant, upward, and downward trends. The dataset presents time series segments alongside multiple-choice textual descriptions, challenging the model to comprehend the time series and select the correct linguistic explanation, thereby serving as a robust benchmark for validating the modality alignment and reasoning proficiency of unified time series foundation models like ChatTime. In our experiments, TSQA contains 48,000 samples in total, split into 38,400 training samples, 4,800 validation samples, and

Prompt Template Used on MMTS-InWild Subset

```
## Time Series Information:
<Insert the time series context here, e.g., statistical metadata, quantized sequences, etc.>

<Insert any auxiliary information here, if applicable>

## Question:
<Insert the question here>

## Options:
<Insert all candidate options here, exactly as listed>

## Instruction:
Output the option letter followed by the full option text exactly as listed in Options. Do not
add explanations or generate any additional content.
```

Prompt Template Used on TimeSeriesExam Dataset

```
## Time Series Information:
<Insert the time series context here, e.g., statistical metadata, quantized sequences, etc.>

## Question:
<Insert the question here>

## Options:
<Insert all candidate options here, exactly as listed>

## Instruction:
For a given question, you should exactly choose one answer from the options, and output the full
answer. Don't generate anything else.
```

4,800 test samples, following an 8:1:1 ratio.

MMTS-Bench. MMTS-Bench (Anonymous, 2026) is a comprehensive multimodal benchmark designed to systematically evaluate the capabilities of Large Language Models (LLMs) in time series understanding and reasoning. Built upon a hierarchical taxonomy that spans feature analysis, temporal reasoning, and cross-modal alignment, the dataset comprises a total of 2,424 Time Series Question Answering (TSQA) pairs distributed across four distinct subsets: Base, InWild, Match, and Align. Specifically, the InWild subset serves as a critical component for assessing model performance in realistic, open-ended scenarios, consisting of 1,084 samples sourced from diverse real-world domains including finance, healthcare, climate, and cloud operations. This subset is constructed using a progressive real-world QA framework to ensure high data quality and relevance, providing a rigorous testbed for measuring the robustness and generalization ability of models when handling complex, authentic time series data.

RWC. The Right Whale Calls (RWC) (Marinexplore and University, 2013) dataset is derived from the Marinexplore and Cornell University Whale Detection Challenge on Kaggle and is designed for automated detection of North Atlantic right whale (Eubalaena glacialis) vocalizations from underwater acoustic recordings. It consists of 2-second audio clips sampled at 2 kHz, collected by autonomous hydrophone buoys in the North Atlantic. The task is formulated as a binary classification problem that identifies the presence of right whale up-calls, a characteristic low-frequency (60250 Hz), approximately one-second rising tonal call. The dataset includes 30,000 labeled training samples and 54,503 unlabeled test samples, and is challenging due to strong interference from ambient ocean noise and anthropogenic acoustic sources.

CGTSF-PTF. The CGTSF-PTF dataset (Wang et al., 2025) serves as a benchmark for context-guided time series forecasting, featuring 12 months of traffic flow data collected from 32 traffic detectors in Paris throughout 2012. Sampled at a one-hour granularity, this dataset aligns numerical time series records with auxiliary

Prompt Template Used on BEDTime Dataset

```
## Time Series Information:
<Insert the time series context here, e.g., statistical metadata, quantized sequences, etc.>

## Question:
<Insert the question here>

## Instruction:
Output a brief natural-language description of the given time series. Do not generate analysis
or any additional content.
```

Prompt Template Used on Forecasting Tasks

```
## Time Series Information:
<Insert the historical time series context here, e.g., statistical metadata and quantized
historical sequence.>

## Context Information:
<Insert additional context information here, e.g., dataset-specific background or
characteristics.>

## Instruction:
Given a univariate historical time series of length <hist_len>, your task is to predict the next
<pred_len> consecutive values.
Additional context information is provided to help you better understand the characteristics
or background of the time series. You may use this context as auxiliary information, but your
prediction must remain consistent with the historical data.
Requirements:
- Predict exactly <pred_len> numerical values.
- The prediction should be based only on the provided historical time series.
- Do NOT include explanations, analysis, or any additional text.
- Output must strictly follow the format below.
Output format:
The predicted next <pred_len> time series values are (scale <scale>):
[Your Answer]
```

textual informationincluding dataset background, Open-Meteo (Open-Meteo, 2021) weather forecasts (such as temperature and weather codes), and calendar details like the day of the week and holidaysconcatenated into coherent descriptions to enable multimodal modeling without future data leakage. Specifically, the forecasting task involves using historical sequence lengths of $\{48, 72, 96, 120\}$ to predict the traffic flow for the subsequent 24 time steps. In our experiments, CGTSF-PTF contains 46,080 samples in total, split into 36,864 training samples, 4,608 validation samples, and 4,608 test samples, following an 8:1:1 ratio.

BEDTime. BEDTime (Sen et al., 2025) is a unified benchmark for evaluating models' ability to describe the structural properties of time series in natural language. It reformulates multiple public time series description datasets into a consistent evaluation framework covering description recognition, description differentiation, and open-ended description generation. In this work, we mainly use four representative subsets from BEDTime: SUSHI, TaxoSynth, TRUCE-Synthetic, and TRUCE-Stock, containing 1,400, 1,400, 1,677, and 5,687 samples, respectively. Each subset follows an approximately 9:1 train/test split. These subsets provide broad coverage across both synthetic and real-world scenarios, as well as short and long textual descriptions. Specifically, TRUCE-Stock contains short, noisy real-world stock price sequences paired with crowd-sourced descriptions, while TRUCE-Synthetic provides short synthetic sequences with human or template-based descriptions. In contrast, SUSHI contains long synthetic time series with prominent trends, seasonality, and noise, and TaxoSynth includes variable-length synthetic time series annotated

with taxonomic descriptions of common time series patterns. Together, these subsets offer a diverse testbed for evaluating whether models can align time series patterns with natural language descriptions across different data sources, time series lengths, and linguistic complexities.

H Evaluation Metrics

This appendix provides the concrete instantiations of the task-specific evaluation metric $\mathcal{M}$ introduced in Sec. 2. As defined in the preliminary formulation, each sample in a dataset $\mathcal{D}$ consists of a time series input T_i, a question Q_i, and the corresponding ground-truth answer G_i. Given the model prediction A_i, the overall performance is computed by averaging $\mathcal{M}(G_i, A_i)$ over all N test samples, as shown in Equation 2. Since $\mathcal{M}$ depends on the tasks, we specify below the exact metrics used for multiple-choice question-answering and resaoning, classification, forecasting, and description tasks in our experiments.

H.1 Question Answering and Reasoning

For multiple-choice QA and reasoning, $\mathcal{M}$ reduces to label agreement and we report *accuracy*,

$$\text{Acc}(\mathcal{D}) \;=\; \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}[A_i = G_i]\,. \qquad (11)$$

H.2 Classification

For time series classification with C classes, we evaluate model predictions using accuracy and macro-averaged F_1 score. Accuracy measures the overall proportion of correctly predicted samples, while Macro-F_1 assigns equal weight to each class and is therefore more robust to class imbalance. For class c, let TP_c denote the number of samples correctly predicted as class c, FP_c the number of samples incorrectly predicted as class c, and FN_c the number of samples from class c incorrectly predicted as other classes. The class-wise precision and recall are then defined as

$$P_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c}, \quad R_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FN}_c},$$
$$\text{Macro-}F_1 = \frac{1}{C}\sum_{c=1}^{C} \frac{2\,P_c\,R_c}{P_c + R_c}. \qquad (12)$$

H.3 Time Series Description

For free-form time series description, we adopt a panel of complementary metrics to evaluate generation quality from multiple perspectives. We report metrics covering three aspects: *lexical overlap* (BLEU-n), *sequence-level surface similarity* (ROUGE-L and METEOR), and *semantic consistency* (DeBERTa F_1 and SimCSE cosine similarity). Let $G = (g_1, \ldots, g_{|G|})$ denote the reference description and $A = (a_1, \ldots, a_{|A|})$ denote the generated answer. All metrics are computed for each sample and then uniformly averaged over the test set $\mathcal{D}$.

BLEU-n. We report sentence-level BLEU-2 and BLEU-4 (Papineni et al., 2002) with uniform weights $w_k = 1/n$, and apply Chen and Cherry's smoothing method 1 (Chen and Cherry, 2014) to avoid zero precision for higher-order n-grams. Let $\mathcal{N}_k(\cdot)$ denote the multiset of k-grams and $c_\bullet(\omega)$ denote the count of an n-gram ω. The BLEU-n score is defined as

$$\text{BLEU-}n \;=\; \text{BP}\cdot\exp\Big(\frac{1}{n}\sum_{k=1}^{n}\log p_k\Big),$$
$$p_k = \frac{\sum_{\omega\in\mathcal{N}_k(A)} \min\big(c_A(\omega), c_G(\omega)\big)}{\sum_{\omega\in\mathcal{N}_k(A)} c_A(\omega)}. \qquad (13)$$

with brevity penalty $\text{BP} = \min\big(1, e^{1-|G|/|A|}\big)$.

ROUGE-L. ROUGE-L evaluates sequence-level overlap using the longest common subsequence between the reference G and the generated answer A (Lin, 2004). We report its F-measure form with $\beta = 1$:

$$\text{ROUGE-}L \;=\; \frac{2\,\text{LCS}(G, A)}{|G| + |A|}. \qquad (14)$$

where $\text{LCS}(G, A)$ denotes the length of the longest common subsequence between the reference G and the generated answer A. We compute ROUGE-L using the `rouge_score` package with Porter stemming (Porter, 1980), which reduces the effect of simple morphological variation, such as *rises* and *rising*.

METEOR. METEOR aligns hypothesis tokens to reference tokens through a cascade of exact, stem, and synonym matches (Banerjee and Lavie, 2005). Let $\mathfrak{A}$ be the resulting alignment, ch denote the number of contiguous matched chunks, and define $P = |\mathfrak{A}|/|A|$ and $R = |\mathfrak{A}|/|G|$. The METEOR score is computed as

$$\text{METEOR} \;=\; \Big(1 - \gamma\big(\text{ch}/|\mathfrak{A}|\big)^{\theta}\Big)\cdot\frac{P\,R}{\alpha P + (1-\alpha)R} \qquad (15)$$

with NLTK defaults $(\alpha, \gamma, \theta) = (0.9, 0.5, 3)$. The fragmentation penalty favors more contiguous alignments, while the stem- and synonym-aware matching makes METEOR less sensitive to surface-form variation than exact n-gram matching.

DeBERTa F_1. We instantiate BERTScore with `deberta-large-mnli` as the backbone (Zhang et al., 2019), taking contextual token embeddings from layer 9 following the library default. This backbone is based on DeBERTa (He et al., 2020) and is fine-tuned on MultiNLI (Williams et al., 2018). For every reference token g_i and answer token a_j, let $s_{ij} = \cos\big(\mathbf{e}(g_i), \mathbf{e}(a_j)\big)$ denote their cosine similarity. The BERTScore F_1 is obtained by greedy maximum matching:

$$F_{\mathrm{BS}} = \frac{2\, P_{\mathrm{BS}}\, R_{\mathrm{BS}}}{P_{\mathrm{BS}} + R_{\mathrm{BS}}},$$
$$P_{\mathrm{BS}} = \tfrac{1}{|A|} \textstyle\sum_j \max_i s_{ij}, \quad R_{\mathrm{BS}} = \tfrac{1}{|G|} \textstyle\sum_i \max_j s_{ij}. \tag{16}$$

We report F_{BS} without IDF weighting or baseline rescaling. That is, all tokens contribute equally, and no model-specific baseline calibration is applied. Since these configuration choices affect the absolute scale of BERTScore, we use this metric only for relative comparison among systems evaluated under the same setting.

SimCSE cosine. For sentence-level semantic similarity, we encode each text using `sup-simcse-roberta-large` (Gao et al., 2021). We take the `pooler_output` and apply ℓ_2 normalization; let $\hat{\mathbf{e}}(\cdot)$ denote the resulting unit vector. The SimCSE score is the cosine similarity between reference and hypothesis embeddings:

$$\mathrm{SimCSE}(G, A) = \hat{\mathbf{e}}(G)^{\top}\, \hat{\mathbf{e}}(A), \tag{17}$$

which is then averaged over $\mathcal{D}$. SimCSE complements token-level matching metrics by providing a sentence-level semantic similarity measure.

Implementation details. All automatic evaluation metrics are computed at the sample level and then uniformly averaged over the test set. For lexical and surface-form metrics, both references and generated answers are lowercased and tokenized by whitespace splitting. We do not remove punctuation. To avoid degenerate overly long generations causing memory issues for embedding-based metrics, both references and generated answers are truncated to at most 200 whitespace-separated words before evaluation.

We compute BLEU scores using `nltk.translate.bleu_score.sentence_bleu` with `SmoothingFunction().method1`. BLEU-2 and BLEU-4 use uniform n-gram weights, i.e., $(1/2, 1/2)$ and $(1/4, 1/4, 1/4, 1/4)$, respectively (Bird, 2006). ROUGE-L is computed using the `rouge_score` package with `rouge_scorer.RougeScorer(["rougeL"], use_stemmer=True)`, and we report the F-measure. METEOR is computed using `nltk.translate.meteor_score.meteor_score` with the NLTK default parameters.

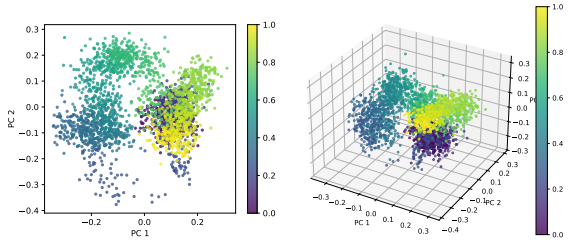
For DeBERTa F_1, we instantiate BERTScore using the `bert_score` package with `deberta-large-mnli` as the backbone, `num_layers=9`, `lang="en"`, `idf=False`, and `rescale_with_baseline=False`. We use batch size 16 for this metric. For SimCSE cosine similarity, we use `sup-simcse-roberta-large` with the Hugging Face `AutoTokenizer` and `AutoModel`; texts are tokenized with padding and truncation to `max_length=128`. We take the `pooler_output`, apply ℓ_2 normalization, and compute the cosine similarity between the reference and generated-answer embeddings. The SimCSE batch size is 64.

I Extended Experiments

I.1 PCA Visualization of ChatTime (Wang et al., 2025)'s TS Embeddings

To preliminarily validate our hypothesis that preserving the continuity and ordinality of TS Tokens in the embedding space is essential for effective time series modeling, we examine ChatTime, a representative token-based TS-LLM, by visualizing the geometry of its TS embeddings, which incorporates a vocabulary of 20,001 additional TS tokens.

As illustrated in the PCA-based 2D and 3D visualizations (Figure 9), interestingly, although ChatTime treats these TS tokens merely as discrete words from a general vocabulary without explicit constraints, the learned embeddings exhibit emerging continuous and ordered structures. This demonstrates that TS tokens spontaneously evolve towards meaningful continuity and ordinality through large-scale training.



(a) PCA-based 2D Visualization of TS Embeddings (b) PCA-based 3D Visualization of TS Embeddings

Figure 9: The PCA visualization of TS-Token embeddings in ChatTime. The color from blue to yellow represents time series values ranging from -1 to $+1$. Notably, the TS tokens exhibit emerging continuous and ordered structures.

I.2 Performance on MTMS-InWild Dataset

To examine whether the proposed COM strategy generalizes to more sophisticated time series reasoning scenarios, we further evaluate model variants with different initialization schemes on the **MMTS-InWild** dataset. MMTS-InWild consists of two major task categories, *analysis* and *reasoning*, covering a broad range of subtasks that reflect the challenges of complex real-world time series understanding.

As shown in Table 6, initialization schemes that better preserve continuity and ordinality consistently outperform variants lacking such geometric structures. These results demonstrate the effectiveness and robustness of hard constraints, suggesting that continuity- and ordinality-aware initialization can benefit not only basic time series analysis but also complex time series reasoning.

I.3 Efficacy of Self-Distillation Alignment

As introduced in Section 3.3, our full training pipeline consists of two phases: *self-distillation alignment* and *downstream fine-tuning*. To examine the effectiveness of the self-distillation alignment phase, we adopt Slerp initialization and evaluate the model on the TSQA (Wang et al., 2025) dataset, comparing the *Aligned* and *Unaligned* configurations.

We consider two evaluation settings: **ID** (in-distribution) and **SID** (semi-in-distribution). In the ID setting, the training data are drawn from the same distribution as the TSQA test set. In the SID setting, we instead use fine-tuning data from ChatTime (Wang et al., 2025); although the query format is consistent with that of TSQA, the underlying time series are from different distributions.

As shown in Table 7, incorporating the self-distillation alignment phase consistently improves performance over the unaligned counterparts under both ID and SID settings. The gains appear not only in overall accuracy but also across all individual subtasks, demonstrating the robustness and effectiveness of self-distillation alignment.

I.4 Performance on CGTSF-PTF Dataset

To assess the broader applicability of our model to time series forecasting, we evaluate it on the CGTSF-PTF dataset (Wang et al., 2025). The task requires predicting the next 24 time steps given historical observation windows of lengths $\{48, 72, 96, 120\}$; detailed dataset information is provided in Appendix G. To ensure a fair comparison, all baselines follow the same normalization procedure as our model, as detailed in Section 3.1.

The quantitative results are reported in Table 8. We implement all forecasting baselines based on the Time-Series-Library codebase (Wang et al., 2026), using its official implementations and recommended hyperparameter settings. For fair comparison, all baselines follow the same data preprocessing, normalization, and evaluation protocol as our method. Although COM is not specifically designed as a dedicated forecasting architecture, it achieves highly competitive performance on CGTSF-PTF, obtaining the best or second-best results across most settings. In particular, COM outperforms several established forecasting baselines under shorter historical windows and remains comparable to strong specialized models under longer histories. These results suggest that, when equipped with appropriate time series token embeddings, token-based TS-LLMs can serve as an effective and competitive formulation for time series forecasting. We further visualize representative results in Figure 10, showing that our model can capture major temporal variations across different history lengths and time series patterns.

J Artifact Licenses and Terms of Use

All datasets, pretrained models, and software packages used in this work are publicly available research artifacts. We use them in accordance with their original licenses and terms of use. Specifically, the forecasting baselines are implemented based on the Time-Series-Library codebase, while the language models, embedding models, and evaluation models used in our experiments retain their

Table 6: Performance on the MMTS-InWild different across different initialization schemes. Analysis tasks include basic, noise, seasonality, trend, and volatility analysis, whereas reasoning tasks include analogical, causal, counterfactual, deductive, and inductive reasoning. The baseline corresponds to `Qwen2.5-3B-Instruct` with plain-text time series input.

Variant	Default	Slerp	PCA-Main	Helix 1	Helix 2	Helix 4	Helix 8	VMF	Baseline
Analysis	0.3619	0.4176	**0.4354**	0.4298	0.4031	0.4198	0.4053	0.3909	0.3742
Reasoning	0.3670	0.4063	**0.4312**	0.4102	0.3997	0.4194	0.4076	0.3853	0.3617
Total	0.3635	0.4068	**0.4271**	0.4253	0.3994	0.4170	0.3994	0.3828	0.3708

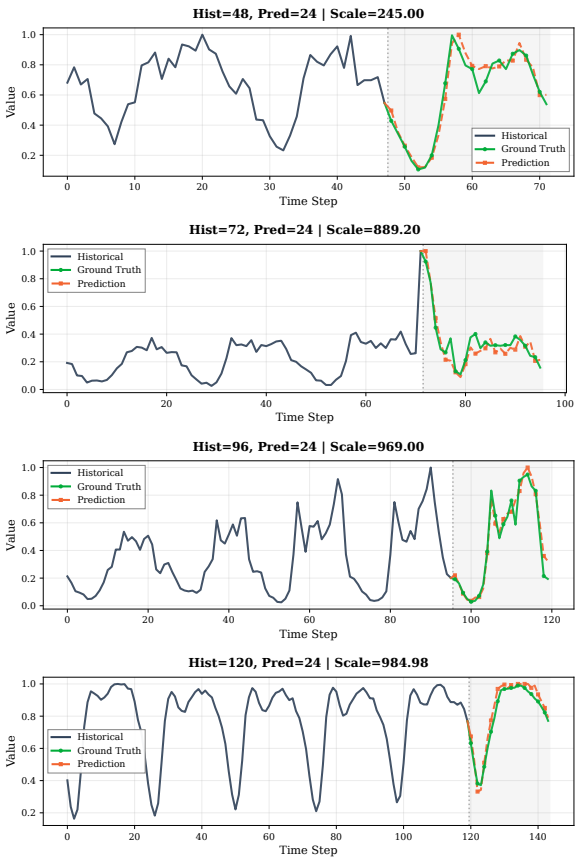


Figure 10: Visualization of our model's forecasting results on the CGTSF-PTF dataset. The four subplots correspond to input history lengths of $H \in \{48, 72, 96, 120\}$, respectively. These results demonstrate that our model achieves good forecasting performance and maintains robustness across varying history lengths and time series patterns.

Table 7: Accuracy (%) comparison of models with and without the *self-distillation alignment* phase.

SETTING	UNALIGNED		ALIGNED	
	SID	ID	SID	ID
TREND	96.71	99.67	98.27	99.67
SEASONALITY	68.98	71.24	73.49	74.75
VOLATILITY	94.50	99.03	95.07	99.43
OUTLIERS	95.48	99.74	98.78	99.83
AVG.	88.94	92.44	91.40	93.52

original licenses and usage restrictions. We do not redistribute third-party datasets or pretrained model weights. For the code and artifacts released with this paper, we will specify the corresponding license in the public repository.

K Use of LLM Assistants

We used LLM-based assistants during the preparation of this work for limited auxiliary purposes. These uses included grammar and style polishing, improving the clarity of English expressions, checking typographical or formatting issues, and performing preliminary manuscript consistency checks. We also used AI assistants to improve code readability and to assist with debugging during implementation.

All AI-assisted outputs were subject to human supervision and post-hoc verification. The authors manually reviewed, revised, and validated any suggested text or code before use. AI assistants were not used to autonomously design the research method, conduct experiments, produce experimental results, derive conclusions, or determine the scientific claims of the paper. The authors remain fully responsible for the correctness, originality, and integrity of the submitted work.

Table 8: Forecasting performance on the PTF dataset under different history lengths. MSE and MAE are reported as evaluation metrics, where lower values indicate better performance. The best and second-best results are highlighted in bold and underlined, respectively.

Model	**History Length**							
	48		**72**		**96**		**120**	
	MSE	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
TimesNet (Wu et al., 2022)	<u>0.0213</u>	<u>0.0928</u>	<u>0.0152</u>	**0.0749**	**0.0113**	**0.0649**	**0.0107**	**0.0636**
iTransformer (Liu et al., 2024b)	0.0321	0.1118	0.0232	0.0947	0.0186	0.0858	0.0171	0.0811
TimeXer (Wang et al., 2024b)	0.0403	0.1246	0.0351	0.1178	0.0239	0.0972	0.0242	0.0987
PatchTST (Nie et al., 2023)	0.0438	0.1323	0.0337	0.1139	0.0247	0.0985	0.0218	0.0910
TimeMixer (Wang et al., 2024a)	0.0462	0.1337	0.0359	0.1158	0.0253	0.0986	0.0226	0.0926
DLinear (Zeng et al., 2023)	0.0722	0.1701	0.0552	0.1487	0.0418	0.1314	0.0371	0.1236
Autoformer (Wu et al., 2021)	0.0871	0.2041	0.0629	0.1659	0.0462	0.1484	0.0434	0.1461
DeepSeek-V4-Pro (DeepSeek-AI, 2026)	0.0443	0.1389	0.0396	0.1293	0.0356	0.1207	0.0348	0.1224
Gemma3-12B-it (Team et al., 2025)	0.0955	0.2177	0.0800	0.1990	0.0661	0.1792	0.0637	0.1752
Qwen3-14B (Yang et al., 2025)	0.0951	0.2248	0.1058	0.2399	0.0863	0.2193	0.0803	0.2109
Qwen2.5-3B (Qwen et al., 2025)	0.1063	0.2286	0.1096	0.2386	0.0894	0.2097	0.0990	0.2265
Qwen2.5-14B (Qwen et al., 2025)	0.1833	0.3408	0.1520	0.3173	0.1386	0.3002	0.1421	0.3040
COM (ours)-3B	**0.0179**	**0.0820**	**0.0143**	<u>0.0755</u>	<u>0.0120</u>	<u>0.0703</u>	<u>0.0121</u>	<u>0.0721</u>