

Bridging Time Series and Language via Query-Adaptive Multi-Granularity Routing for Temporal Reasoning

Hyuna Jeon^{†,1} Jaeyoung Lee^{†,2} Misuk Kim^{*,1,2}

¹Department of Artificial Intelligence, Hanyang University, Seoul, Republic of Korea

²Department of Data Science, Hanyang University, Seoul, Republic of Korea

{hyunajeon, jaeylee, misukkim}@hanyang.ac.kr

Abstract

Time-Series Question Answering (TSQA) aims to interpret time-series information and generate answers according to the intent of natural language questions. However, in TSQA, the required temporal granularity and reasoning scope vary across queries, making it crucial to effectively organize the time-series information needed for each query. In this study, we propose TiMER (Time-series and language alignment with Multi-granularity Expert Routing), a time-series-language alignment framework that selectively constructs multi-granularity time-series information according to query intent and connects it to the LLM input space. Experimental results on a large-scale TSQA benchmark show that TiMER achieves an overall accuracy of 81.02% using only a small number of external alignment modules while keeping the LLM frozen, outperforming various LLM-based baselines. In addition, extensive ablation studies verify the effectiveness of each component. This study presents an effective direction for natural language-based time-series understanding and temporal reasoning by dynamically organizing and utilizing time-series information according to query intent. The code is publicly available at https://github.com/hyuna0926/TiMER_paper.

1 Introduction

Time-series data are observed across various domains, such as industrial equipment, healthcare, and finance, and contain complex temporal structures, including short-term fluctuations, long-term trends, periodicity, and anomalous patterns. To analyze such data, various tasks, including forecasting, classification, anomaly detection, and imputation, have been actively explored along with statistical methods, deep learning models, and recent

time-series foundation models (Nie et al., 2024; Yang et al., 2023; Jiang et al., 2024; Yuan et al., 2023). However, existing approaches are mainly optimized for predefined domains, tasks, and fixed output formats (Zhou et al., 2022, 2023), which limits their ability to perform complex temporal reasoning required in real-world settings, such as interpreting changes, diagnosing states, and identifying the causes and evidence of anomalous symptoms. In practical applications, natural language queries beyond simple prediction are increasingly required, such as “When did the anomaly begin?” and “Which segment serves as the evidence for the decision?” Accordingly, time-series analysis is expanding into an interactive analytical problem that interprets and reasons over data based on natural language queries (Chen et al., 2026; Kong et al., 2025). Motivated by these demands, recent studies have paid increasing attention to leveraging large language models (LLMs) (Zhao et al., 2026), which possess complex reasoning abilities and instruction-following capabilities, for time-series analysis. LLMs can flexibly incorporate the analytical purpose and context through natural language queries, but there remains a fundamental gap between natural language token-based representations and the time-dependent structure of time-series data. Therefore, it is necessary to develop methods that effectively preserve temporal information in time series and connect it to LLMs for utilization (Merrill et al., 2024; Liang et al., 2024).

Recent studies on combining LLMs with time-series data have expanded in various directions, including LLM adaptation, time-series-language transformation, representation alignment, and the construction of time-series question answering (TSQA) benchmarks (Xue and Salim, 2024; Zhou et al., 2023; Jin et al., 2024; Zhang et al., 2025). Existing approaches mainly adapt LLMs to time-series tasks or connect time-series representations to frozen LLMs by transforming them

[†]Equal contribution. *Corresponding author.

into textual descriptions or latent tokens. However, these methods often incur substantial additional training costs or compress time-series information into limited representations, making it difficult to sufficiently capture the temporal structure between global trends and local variations (Li et al., 2024; Wang et al., 2025; wang et al., 2025). In particular, in question-answering-based analysis such as TSQA, the required temporal granularity and reasoning scope vary across queries, and thus fixed time-series representation schemes are insufficient to address diverse analytical demands.

Therefore, in this study, we propose **TiMER** (**T**ime-series and language alignment with **M**ulti-granularity **E**xpert **R**outing), a time-series-language alignment framework that selectively utilizes temporal granularity information required by the intent of a query to effectively integrate time-series information into LLMs. TiMER first encodes the input time series from the perspectives of global trends and local variations, and then dynamically estimates the importance of each granularity based on the natural language query to integrate local, global, and cross-granularity information. The integrated time-series representation is then aligned with the input representation space of the LLM, enabling the model to effectively utilize time-series information that is relevant to the query intent. To validate the performance of the proposed model, we conduct experiments in a large-scale TSQA setting consisting of 13 domains, 6 time-series analysis tasks, and 3 question types. The results show that TiMER consistently outperforms LLM-based baselines of various scales and types using only a small number of additional trainable parameters, corresponding to 0.08% of the total parameters. In addition, extensive ablation studies and case studies verify the effectiveness of multi-granularity routing and representation alignment. The main contributions of this study are summarized as follows:

- We propose TiMER, a time-series-language alignment framework that selectively constructs temporal granularity information required by the query intent and effectively aligns it with LLMs.
- We evaluate TiMER in a large-scale TSQA setting composed of diverse time-series analysis tasks, and quantitatively and qualitatively

show that it consistently outperforms LLM-based methods of various scales and types using only a small number of additional trainable parameters, corresponding to 0.08% of the total parameters.

- Through extensive ablation studies, we analyze that the required temporal granularity varies depending on the question type and verify that query-aware granularity routing and representation alignment are key contributors to performance improvement.
- We empirically demonstrate that aligning time-series input representations with the LLM input space can achieve performance comparable to instruction tuning, while offering a more efficient alternative that avoids directly adapting the response generation behavior of the LLM.

2 Related Work

Recent studies have actively explored the use of LLMs for time-series data. GPT4TS and LLTime showed that pretrained language models can be utilized for time-series analysis (Zhou et al., 2023; Gruver et al., 2023), while Time-LLM, TEMPO, and UniTime extended LLM-based forecasting by transforming time-series representations into LLM-compatible inputs (Jin et al., 2024; Cao et al., 2024; Liu et al., 2024). In addition, time-series foundation models such as Chronos, TimesFM, and TimeGPT have demonstrated forecasting generalization across diverse domains through large-scale pretraining (Ansari et al., 2024; Das et al., 2024; Garza et al., 2024). However, most existing studies focus on forecasting-oriented objectives or rely on fixed latent representations or limited temporal abstractions. Such approaches struggle to handle cases where the required temporal granularity and reasoning scope vary with the query intent.

Meanwhile, recent studies have extended time-series analysis to language-based QA and instruction-following settings, such as Time-MQA, ChatTime, ChatTS, and ITFormer (Kong et al., 2025; Wang et al., 2025; Xie et al., 2025; wang et al., 2025). MTBench, Time-MQA, and TSAQA have also introduced benchmarks for evaluating temporal reasoning and TSQA capabilities (Chen et al., 2026; Kong et al., 2025; Jing et al., 2026). In TSQA, the scope and form of temporal evidence required from the same time series can vary de-

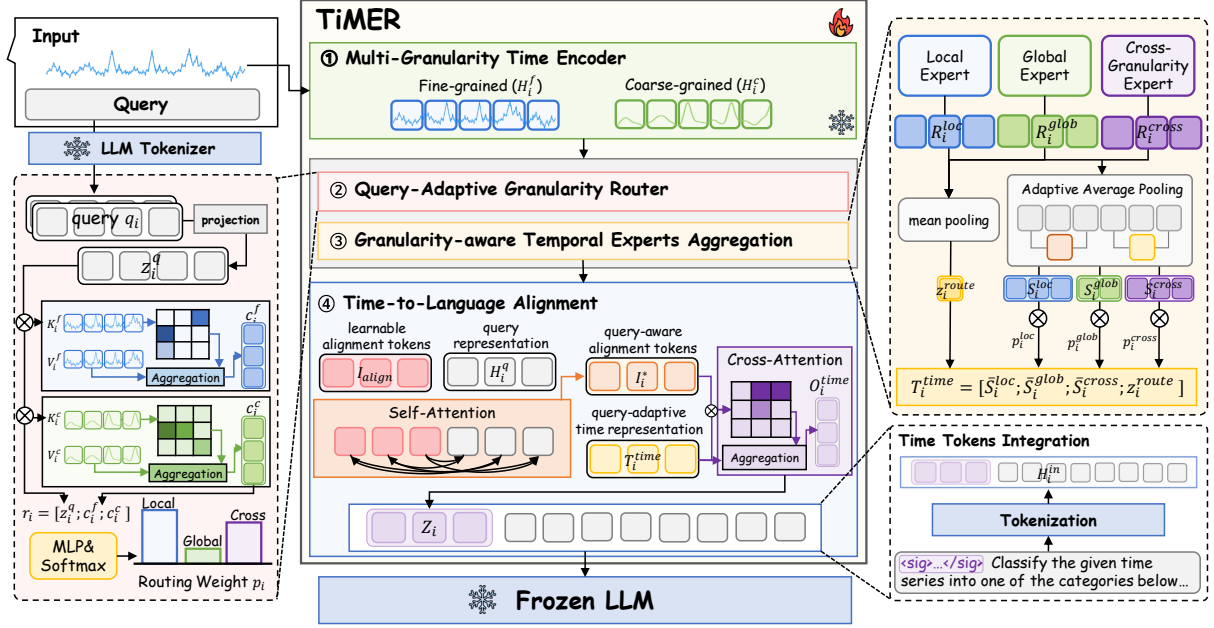


Figure 1: Overview of TiMER. TiMER consists of a Multi-Granularity Time Encoder that extracts fine- and coarse-grained time-series representations, an Query-Adaptive Granularity Router that estimates query-dependent granularity weights, Granularity-aware Temporal Experts Aggregation that construct instruction-adaptive time representations, and Time-to-Language Alignment that converts them into language-aligned time tokens for frozen LLM-based answer generation.

pending on the semantics of the query. From this perspective, our approach incorporates the query not only at the final QA stage but also directly during temporal representation construction. Rather than using a preconstructed shared representation uniformly across queries, TiMER performs query-conditioned time-series information construction, dynamically determining the scope and form of temporal information required by each question. The proposed Query-Adaptive Granularity Router dynamically adjusts the selection and aggregation of temporal information according to the query, enabling TiMER to flexibly accommodate the diverse ranges of temporal reasoning required in TSQA.

3 Problem Formulation

This study formulates TSQA as a problem to handle various time-series analysis tasks in a unified QA format. The TSQA dataset is defined as follows:

$$\mathcal{D} = \{(X_i, q_i, a_i)\}_{i=1}^N, \quad (1)$$

where N denotes the total number of samples, X_i denotes the time-series input, q_i denotes a query, and a_i denotes the answer to the corresponding query. The goal of TSQA is to generate the answer a_i given X_i and q_i . This study uses a pretrained frozen LLM and employs a trainable alignment

module f_ϕ to transform the time-series input and query into time-series representations:

$$Z_i = f_\phi(X_i, q_i), \quad (2)$$

where ϕ denotes the trainable parameters of the alignment module, and Z_i denotes the time-series token representation aligned with the LLM input space. The transformed Z_i is integrated into the LLM input sequence together with the query q_i and used to generate the answer.

4 TiMER

TiMER is a time-series-language alignment framework that selectively constructs temporal granularity information required by the semantics of a query and aligns it with the LLM input space (Figure 1). First, the raw time series is encoded into fine-grained and coarse-grained representations (Section 4.1), and routing weights for each granularity are computed by jointly considering the query and time-series information (Section 4.2). Then, local, global, and cross-granularity information is reconstructed, and query-adaptive time representations are generated by incorporating the routing weights (Section 4.3). Finally, these representations are aligned with the LLM input space, enabling the

LLM to generate answers based on time-series information (Section 4.4).

4.1 Multi-Granularity Time Encoder

TiMER encodes the input time series into fine-grained and coarse-grained representations to effectively utilize information at different temporal granularities (Park et al., 2025). Specifically, the fine-grained representation captures short-term changes, spikes, and local temporal patterns, while the coarse-grained representation represents long-term trends and overall temporal dynamics. These representations are generated as follows:

$$H_i^f, H_i^c = E_{\text{MTE}}(X_i), \quad (3)$$

where $H_i^f \in \mathbb{R}^{L_i^f \times d_{\text{MTE}}}$ and $H_i^c \in \mathbb{R}^{L_i^c \times d_{\text{MTE}}}$ denote the fine-grained and coarse-grained representations, respectively, and L_i^f and L_i^c denote the numbers of temporal tokens. These two representations are subsequently used as base representations for selectively constructing temporal granularity information required by the query intent. The Multi-Granularity Time Encoder (MTE) E_{MTE} is pretrained in advance, and the encoder is kept frozen during TSQA training and inference. The architecture and pretraining procedure of MTE are provided in Appendix A.

4.2 Query-Adaptive Granularity Router

Query-Adaptive Granularity Router (QGR) dynamically estimates the temporal granularity required for a query based on the interaction between the natural language query and multi-granularity time-series representations. Given a query q_i , we feed it into the LLM and extract the last-token representation from the final-layer hidden state. This representation is projected into the MTE hidden dimension to obtain the query representation $z_i^q \in \mathbb{R}^{d_{\text{MTE}}}$. The query representation z_i^q is then used as an attention query to perform cross-attention over H_i^f and H_i^c , producing attended summary representations for each temporal granularity.

$$c_i^{(\cdot)} = \text{Softmax} \left(\frac{Q_i^{(\cdot)} (K_i^{(\cdot)})^\top}{\sqrt{d_k}} \right) V_i^{(\cdot)}, \quad (4)$$

where $Q_i^{(\cdot)}$, $K_i^{(\cdot)}$, and $V_i^{(\cdot)}$ denote the query, key, and value for each temporal granularity, respectively, and d_k is the query/key projection dimension used for computing attention scores. Finally, QGR computes the routing weights p_i based on

$r_i = [z_i^q; c_i^f; c_i^c]$, which combines the query representation and the granularity-specific summaries:

$$p_i = \text{Softmax}(\text{MLP}(r_i)). \quad (5)$$

According to the granularity expert configuration of TiMER, p_i is defined as $p_i = (p_i^{\text{loc}}, p_i^{\text{glob}}, p_i^{\text{cross}})$, where each element is used in the subsequent stage to control the contribution of each expert output.

4.3 Granularity-aware Temporal Expert Aggregation

Granularity-aware Temporal Experts (GTE) take H_i^f and H_i^c extracted from MTE as inputs, and model and integrate information at different temporal granularities. GTE consists of local, global, and cross-granularity experts, and adaptively reflects the granularity information required by the query through the routing weights p_i computed by QGR.

Temporal Granularity Experts. The local, global, and cross-granularity experts model fine-grained patterns, coarse-grained dynamics, and interactions between granularities, respectively:

$$\begin{aligned} R_i^{\text{loc}} &= \mathcal{T}_{\text{loc}}(H_i^f), \\ R_i^{\text{glob}} &= \mathcal{T}_{\text{glob}}(H_i^c), \\ R_i^{\text{cross}} &= \mathcal{C}_{\text{cross}}(H_i^f, H_i^c), \end{aligned} \quad (6)$$

where $R_i^{\text{loc}}, R_i^{\text{cross}} \in \mathbb{R}^{L_i^f \times d_{\text{MTE}}}$ and $R_i^{\text{glob}} \in \mathbb{R}^{L_i^c \times d_{\text{MTE}}}$. $\mathcal{T}_{\text{loc}}$ and $\mathcal{T}_{\text{glob}}$ consist of self-attention and feed-forward networks (FFNs), while $\mathcal{C}_{\text{cross}}$ is implemented with cross-attention and an FFN, using H_i^f as the query and H_i^c as the key-value representation.

Query-adaptive Time Representation. GTE applies mean pooling to each expert output to generate expert-level summaries $z_i^{\text{loc}}, z_i^{\text{glob}}, z_i^{\text{cross}} \in \mathbb{R}^{d_{\text{MTE}}}$. Then, it computes the expert-aggregated representation $\tilde{z}_i$ by incorporating the routing weights p_i :

$$\tilde{z}_i = p_i^{\text{loc}} z_i^{\text{loc}} + p_i^{\text{glob}} z_i^{\text{glob}} + p_i^{\text{cross}} z_i^{\text{cross}}. \quad (7)$$

The aggregated representation $\tilde{z}_i$ is then transformed into the routed representation $z_i^{\text{route}} \in \mathbb{R}^{d_{\text{MTE}}}$ through output projection and normalization. In addition, to preserve temporal information for each granularity, adaptive average pooling is applied to each expert output to obtain $S_i^{\text{loc}} \in \mathbb{R}^{n_{\text{loc}} \times d_{\text{MTE}}}$, $S_i^{\text{glob}} \in \mathbb{R}^{n_{\text{glob}} \times d_{\text{MTE}}}$, and

$S_i^{\text{cross}} \in \mathbb{R}^{n_{\text{cross}} \times d_{\text{MTE}}}$. Each summary representation is then modulated by its corresponding routing weight (e.g., $\bar{S}_i^{\text{loc}} = p_i^{\text{loc}} S_i^{\text{loc}}$), and the final query-adaptive time representation is constructed as follows:

$$T_i^{\text{time}} = [\bar{S}_i^{\text{loc}}; \bar{S}_i^{\text{glob}}; \bar{S}_i^{\text{cross}}; z_i^{\text{route}}]. \quad (8)$$

$T_i^{\text{time}} \in \mathbb{R}^{N_{\text{time}} \times d_{\text{MTE}}}$ is a representation that reflects local, global, and cross-granularity information according to the query, where $N_{\text{time}} = n_{\text{loc}} + n_{\text{glob}} + n_{\text{cross}} + 1$, and n_{loc} , n_{glob} , and n_{cross} are hyperparameters denoting the numbers of local, global, and cross-granularity summary representations, respectively.

4.4 Time-to-Language Alignment

Time-to-Language Alignment (TLA) aligns T_i^{time} with the LLM input space and transforms it into language-aligned time representations that can be utilized by the LLM.

Language-aligned Time Tokens. Given a query q_i with token length L_i^q , we obtain the query representation $H_i^q \in \mathbb{R}^{L_i^q \times d_{\text{LLM}}}$ and introduce learnable alignment tokens $I_{\text{align}} \in \mathbb{R}^{L_{\text{align}} \times d_{\text{LLM}}}$ of length L_{align} . The alignment tokens are concatenated with the query representation, and self-attention is applied as $H_i^{\text{align}} = \text{SelfAttn}([I_{\text{align}}; H_i^q])$. We then extract the hidden states corresponding to I_{align} as the query-aware alignment tokens I_i^* , which encode the semantics of the query. Using I_i^* as the query and T_i^{time} as the keys and values, cross-attention produces the query-conditioned time representation O_i^{time} :

$$O_i^{\text{time}} = \text{CrossAttn}(I_i^*, T_i^{\text{time}}). \quad (9)$$

where $O_i^{\text{time}} \in \mathbb{R}^{L_{\text{align}} \times d_{\text{TLA}}}$. Within the attention operation, the query and key/value representations are projected into the shared internal TLA attention dimension d_{TLA} . Finally, O_i^{time} is projected into the LLM hidden dimension to generate the language-aligned time representations $Z_i \in \mathbb{R}^{L_{\text{align}} \times d_{\text{LLM}}}$.

Time Tokens Integration. Given the initial LLM input H_i^{in} , which includes the query and signal boundary tokens, and the language-aligned time representations Z_i , we apply the operation $\text{Integrate}(\cdot, \cdot)$ to place Z_i between the special tokens indicating the start and end of the time-series information:

$$\tilde{H}_i^{\text{in}} = \text{Integrate}(H_i^{\text{in}}, Z_i), \quad (10)$$

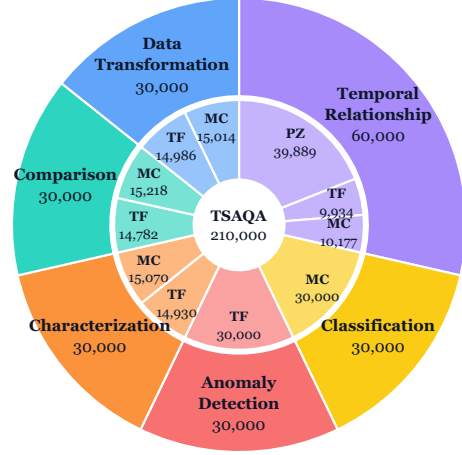


Figure 2: Distribution of task types and question types in the TSAQA dataset.

Through this process, the time-series information is integrated into the same LLM input sequence as the query, and the LLM generates the answer a_i given $\tilde{H}_i^{\text{in}}$ as input.

Alignment Training. TiMER is trained with a causal language modeling objective, where the loss is computed only on the answer tokens. During training, both the LLM and MTE are kept frozen, and only the parameters of QGR, GTE, and TLA are updated. This enables efficient alignment of time-series representations with the LLM input space while preserving the language understanding and generation capabilities of the LLM and the time-series representation capability of MTE. In addition, to mitigate expert collapse, we apply balance regularization to the routing distribution, and the final objective is defined as $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{align}} + \lambda_{\text{bal}} \mathcal{L}_{\text{bal}}$. The detailed loss formulation is provided in Appendix B.

5 Experimental Setup

Datasets. To evaluate the performance of TiMER, we use the Time Series Analysis Question Answering Benchmark (TSAQA) (Jing et al., 2026), a large-scale TSQA benchmark. TSAQA consists of 13 domains, 6 time-series analysis tasks, and 3 question types, comprising a total of 210K QA pairs. The time-series analysis tasks include anomaly detection (A.D), classification (CLS), characterization, comparison, data transformation, and temporal relationship, while the question types include true/false (TF), multiple-choice (MC), and puzzling (PZ). Task and question-type distributions are shown in Figure 2, with further dataset details

Type	Model	Size	A.D	CLS	Characterization		Comparison		Data Transform		Temporal Relation			Overall
			MC	MC	TF	MC	TF	MC	TF	MC	TF	MC	PZ	
Proprietary LLMs	GPT-4.1	-	55.85	50.38	92.97	89.36	<u>83.57</u>	76.99	54.36	51.13	65.90	79.09	45.77	62.82
	GPT-4o	-	54.32	47.20	88.15	<u>84.15</u>	78.61	69.07	60.66	53.24	62.25	75.58	45.61	60.73
	Claude-3.5-Sonnet	-	51.27	41.23	74.39	78.45	66.59	74.14	65.79	57.07	82.05	82.15	54.56	61.19
	Gemini-2.5-Flash	-	52.08	49.07	85.48	81.08	77.79	72.21	63.62	60.17	75.05	84.49	60.84	65.08
Open-weight LLMs	LLaMA3.3	70B	54.23	55.78	76.76	74.17	58.57	55.35	52.40	39.16	51.56	59.97	36.04	53.35
	Qwen2.5	72B	53.73	49.15	86.21	79.61	74.77	60.69	60.86	34.06	57.05	39.11	36.98	54.53
	Qwen3	32B	53.82	48.03	81.92	77.34	74.06	58.85	51.77	29.79	49.27	37.03	35.21	52.06
	Gemma-4	31B	51.45	47.00	77.59	81.92	72.70	68.15	55.26	40.36	72.03	42.93	46.17	56.65
	Qwen3	8B	50.60	50.52	77.35	66.87	71.04	63.21	52.43	34.46	65.22	67.14	21.93	51.04
	LLaMA3.1	8B	54.92	50.20	68.10	62.26	67.84	49.98	51.90	36.56	54.82	40.95	6.80	44.93
	Ministral	8B	53.35	34.08	71.06	63.93	47.54	52.90	50.70	25.28	50.58	33.88	30.77	44.65
	Qwen3	0.6B	50.40	35.83	62.00	48.78	58.03	37.51	49.03	23.62	51.99	37.33	13.38	39.06
	LLaMA3.2	1B	49.47	39.48	63.74	52.55	61.02	36.82	48.87	4.20	48.97	5.44	6.76	35.70
	Gemma3	1B	49.15	49.83	63.74	47.71	61.19	43.37	49.37	24.88	49.42	25.84	23.97	43.03
Time-Series LLMs	ChatTime	7B	54.68	35.10	47.49	32.40	45.53	40.56	49.50	40.56	47.81	26.30	27.70	39.94
	ITFormer	7B	49.12	52.95	65.20	54.99	62.17	47.30	50.57	25.32	48.74	24.72	22.01	44.11
	ITFormer [†]	7B	50.22	56.57	85.78	71.14	79.32	59.21	51.10	24.92	51.12	25.64	28.74	50.93
Proposed Method	TiMER	0.6B	<u>85.90</u>	93.82	89.88	74.21	79.93	62.03	<u>85.64</u>	<u>79.42</u>	<u>86.72</u>	80.01	58.75	<u>78.45</u>
	TiMER	8B	86.00	<u>92.73</u>	<u>90.28</u>	81.65	85.30	<u>75.71</u>	87.24	81.99	88.67	<u>83.72</u>	<u>59.85</u>	81.02

Table 1: Performance comparison on the TSAQA benchmark. All values are reported in percentage (%), and *Overall* denotes the mean accuracy across all task/question types. Best results are shown in bold, and second-best results are underlined. [†] denotes ITFormer additionally fine-tuned on the TSAQA training set; all other Time-Series LLM results are obtained using their publicly released checkpoints.

in Appendix C.1.

Baselines & Evaluation. We use Qwen3 as the frozen LLM and compare its performance with baselines of various types and scales, including proprietary LLMs, open-weight LLMs, and time-series LLMs. Performance is evaluated based on accuracy following the TSAQA evaluation protocol. TF and MC questions are evaluated using exact match, while PZ questions are measured by the position-wise matching ratio between the predicted patch order and the ground-truth order.

Implementation. We set the GTE summary lengths to $(n_{\text{loc}}, n_{\text{glob}}, n_{\text{cross}}) = (8, 4, 8)$ and the TLA alignment token length to $L_{\text{align}} = 25$. The balance loss weight is set to $\lambda_{\text{bal}} = 10^{-3}$. All experiments are conducted on a single NVIDIA H200 GPU with mixed-precision BF16 training. Additional experimental settings, including implementation details, are provided in Appendix C.

6 Experimental Results

6.1 Main TSQA Results

Table 1 presents the performance comparison results on the TSAQA benchmark. Overall, TiMER achieves the highest overall accuracy among all comparison models and obtains the best or second-best performance on most tasks and question types. In particular, TiMER-8B improves the overall accu-

racy by 15.94% compared with Gemini-2.5-Flash, the strongest proprietary LLM on average. These results demonstrate that TiMER can generalize robustly across diverse analytical objectives and question formats, rather than being limited to a specific time-series task.

Proprietary LLMs show competitive performance in some task/question types. For example, GPT-4.1 achieves strong performance on TF and MC questions in characterization, while Gemini-2.5-Flash shows performance comparable to TiMER on some question types in temporal relationship. This suggests that semantic reasoning and pattern recognition capabilities acquired through large-scale pretraining can be effective for certain temporal reasoning tasks. However, even these large-scale multimodal-capable LLMs underperform TiMER in most TSQA settings, indicating that query-based time-series understanding cannot be sufficiently addressed by model scale or general reasoning capability alone.

Open-weight LLMs tend to improve as model size increases, but their overall generalization performance on TSQA remains limited. In contrast, TiMER-0.6B outperforms 70B- and 72B-scale open-weight LLMs by 23.92%-25.10%. Time-series LLMs such as ChatTime and ITFormer also exhibit relatively low overall accuracy. Notably, even ITFormer[†], which is directly fine-tuned on the TSAQA dataset used in this study, achieves

[Question]

Considering the first time series, which of the following is a correct observation regarding the components of changing seasonality, trend strength, and variance shift? Respond ONLY with the letter of the correct choice (A, B, C, or D).

Choices:

A: The seasonality is constant, the overall trend is absent, and there's no variance shift.

B: The series exhibits changing seasonality, weak trend strength, and noticeable variance shifts.

C: There is strong seasonality, increasing trend, and constant variance.

D: The series has irregular seasonality, strong trend direction, and stable variance.

✓ Prediction: B

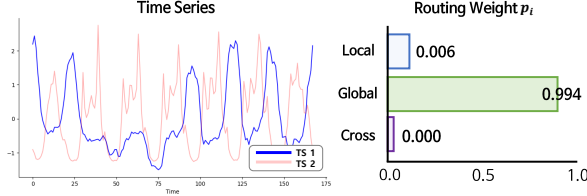


Figure 3: Case study of TiMER on a characterization task. QGR assigns the dominant weight to the global expert, illustrating query-dependent selection of temporal representations.

an overall accuracy of only 50.93, while TiMER-0.6B surpasses it by 26.47 percentage points. This result suggests that task-specific fine-tuning alone is insufficient to effectively handle diverse TSQA queries and highlights the effectiveness of TiMER’s design in constructing appropriate temporal representations adaptively to each query.

In summary, existing proprietary LLMs, open-weight LLMs, and time-series LLMs have limitations in query-aware temporal reasoning and generalization across diverse QA settings required in TSQA. In contrast, TiMER selectively constructs temporal granularity information required by the question intent and aligns it with the LLM input space using only a small number of alignment modules, corresponding to 0.08% of the total parameters. As a result, TiMER effectively generalizes across diverse domains, task, and question types.

6.2 Case Study

In Figure 3, we present the answer generation process of TiMER for an MC question in the characterization task. The question requires jointly assessing changing seasonality, trend strength, and variance shift, making global temporal structure more important than local variations. The first time series shows changing seasonality and variance shifts without a clear monotonic trend, so the correct answer is option B. TiMER assigns the highest routing weight to the global expert and correctly predicts B. This shows that query-adaptive routing selectively utilizes the temporal granularity re-

Model	Method	Overall (%)	Train. (h)	Infer. (s)
LLaMA3.2-1B	Inst.	73.48	30.96	2.21
	TiMER	77.04 (↑3.56)	16.0 (↓48.3%)	1.04 (↓52.9%)
Qwen3-0.6B	Inst.	78.32	51.8	1.39
	TiMER	78.45 (↑0.13)	29.3 (↓43.4%)	1.25 (↓10.1%)

Table 2: Comparison between instruction tuning and TiMER across different LLMs. *Inst.* denotes LoRA-based instruction tuning. *Train.* is the total training time in hours for 10 epochs, and *Infer.* is the per-sample inference time in seconds.

Setting	Components				Overall Accuracy
	MTE	GTE	QGR	TLA	
(a)	✓				52.59
(b)	✓	✓			56.09
(c)	✓	✓	✓		58.96
(d)	✓			✓	71.25
(e)	✓	✓		✓	74.60
Ours	✓	✓	✓	✓	78.45

Table 3: Effect of TiMER components on overall accuracy. Each component provides complementary performance gains, with the full model achieving the best overall accuracy.

quired by the question intent, enabling time-series interpretation and temporal reasoning aligned with the query. Additional case studies and task-level routing analyses are provided in Appendix D.1.

6.3 Ablation Study

We conduct ablation studies to verify the effectiveness of each component of TiMER and the impact of detailed configurations on performance. All experiments are conducted based on TiMER-0.6B, and detailed results for each experiment are provided in Appendix E.

Effect of Time-to-Language Alignment. In TSQA, LLMs must jointly understand time-series inputs and natural language queries. However, conventional instruction tuning mainly adjusts response generation to the training data, rather than explicitly aligning time-series representations. We therefore compare TiMER with LoRA (Hu et al., 2022)-based instruction tuning to evaluate the effectiveness and efficiency of time-to-language alignment (Table 2). TiMER achieves comparable overall accuracy than instruction tuning on both LLaMA3.2-1B and Qwen3-0.6B, while substantially reducing training and inference costs. In particular, on LLaMA3.2-1B, it reduces both training and inference time by approximately 50% while improving accuracy. These results suggest that aligning time-series representations with the LLM input

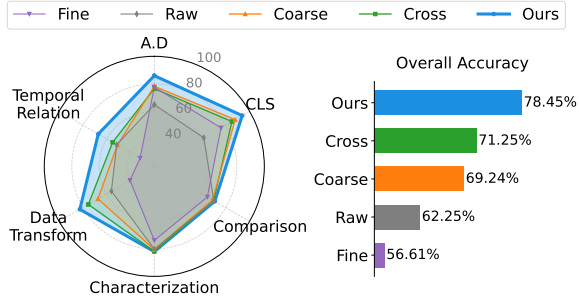


Figure 4: Effect of temporal granularity on TSQA performance. The left plot shows task-wise accuracy, and the right plot shows overall accuracy.

space is more effective and efficient for TSQA than directly tuning the LLM’s response generation. The consistent trend across different LLMs also shows that TiMER generalizes reliably without depending on a specific backbone.

Effect of TiMER Components. Table 3 presents the ablation results for each component of TiMER. Compared with the baseline using only MTE, performance gradually improves as GTE and QGR are added, with the largest gain observed when TLA is applied. This shows that temporal granularity-based representation modeling and query-adaptive routing effectively organize time-series information, while aligning it with the LLM input space is crucial for improving TSQA performance. The additional improvement from jointly using GTE and TLA further indicates that temporal granularity modeling and time-to-language alignment work complementarily. Overall, these results verify that each component of TiMER effectively contributes to time-series information construction and alignment with the LLM input space.

Effect of Temporal Granularity Modeling. To analyze the effect of temporal granularity modeling, we compare TiMER with variants that individually use raw representations and each granularity representation (Figure 4). The results show that temporal granularity-based representations consistently improve performance over raw representations, with the cross-granularity representation achieving the best performance among single-representation variants. This indicates that structuring time series from the perspective of temporal granularity is effective. However, using only a single granularity representation still underperforms TiMER, with particularly large drops in data transformation and temporal relationship tasks. This suggests that selectively

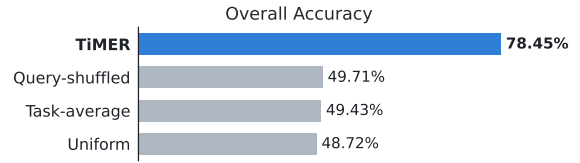


Figure 5: Effect of query-conditioned routing interventions on overall accuracy. Each intervention modifies only the routing strategy during inference.

Task	Type	ITFormer	TiMER	Δ
A.D	–	60.90	65.16	$\uparrow$ 4.26
CLS	–	26.54	32.53	$\uparrow$ 5.99
Reasoning QA	MC	28.35	32.17	$\uparrow$ 3.82
	T/F	49.49	55.25	$\uparrow$ 5.76
	Open-Ended	0.08	0.20	$\uparrow$ 0.12

Table 4: Effect of TiMER across diverse Time-MQA tasks. Results are reported in Macro-F1, except for Open-Ended, which is evaluated using BERTScore.

integrating fine-grained, coarse-grained, and cross-granularity information according to the query intent is important for complex temporal reasoning.

Effect of Query-Conditioned Routing To examine whether QGR merely relies on task-level routing patterns or selects temporal representations based on the semantics of individual queries, we conduct diagnostic experiments that modify only the routing mechanism at inference time. As shown in Figure 5, *task-average routing*, which uses the average routing weights within each task, substantially reduces performance to 49.43 despite preserving task-level routing tendencies. This result indicates that the gains of TiMER cannot be explained simply by learning a fixed expert preference for each task. Similarly, *query-shuffled routing*, in which the router receives a query from another sample, reduces performance to 49.71, highlighting the importance of alignment between query semantics and temporal expert selection. A comparable degradation is also observed with *uniform routing*, which assigns equal weights to all experts. Taken together, these results support that QGR does not merely reproduce task-level routing patterns, but dynamically constructs temporal representations at an appropriate granularity according to the temporal evidence required by each individual query.

Effect of TiMER across Different Time-Series QA Tasks. To evaluate the generalizability of TiMER beyond the TSAQA benchmark, we further conduct experiments on Time-MQA (Kong et al.,

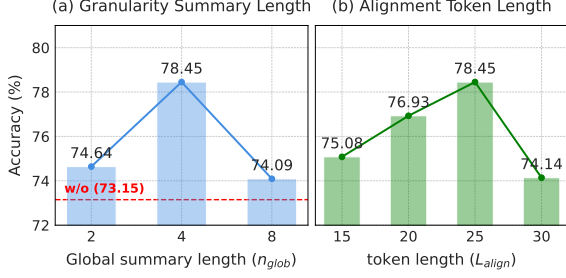


Figure 6: Effect of granularity summary length and alignment token length on overall accuracy. w/o denotes the setting without adaptive average pooling.

2025), an additional TSQA benchmark. We evaluate tasks that are directly aligned with the query-conditioned answer generation setting of TiMER, including anomaly detection, classification, and multiple forms of reasoning QA. For a fair comparison, both ITFormer-0.6B and TiMER-0.6B are trained on Time-MQA under the same setting. As shown in Table 4, TiMER consistently outperforms ITFormer across all evaluated tasks. These results demonstrate that the gains of TiMER are not limited to the specific data composition of TSAQA, but generalize to an external TSQA benchmark with different data sources and query formats. In particular, the ability of TiMER to select and integrate query-relevant temporal information is effective across diverse TSQA tasks, ranging from anomaly detection and time-series classification to natural language reasoning.

Effect of Granularity Summary Representation Length. GTE generates granularity-wise summary representations through adaptive pooling. To analyze the effect of summary representation length, we vary $n_{glob} \in \{2, 4, 8\}$ and set the local and cross summary lengths proportionally (Figure 6 (a)). The results show that all settings with adaptive pooling outperform the variant w/o pooling (73.15%), with $n_{glob} = 4$ achieving the highest overall accuracy. This indicates that summarizing temporal information at each granularity with an appropriate length is effective, and that balancing detail preservation and representation sequence length is important.

Effect of Alignment Token Length. To analyze the effect of alignment token length, we vary $L_{align} \in \{15, 20, 25, 30\}$ and compare the performance (Figure 6 (b)). The results show that overall accuracy improves to 78.45% as the alignment token length increases from 15 to 25, but decreases

to 74.14% when it is set to 30. This suggests that a short alignment token length may limit the incorporation of query-relevant time-series information, while an excessively long length may introduce redundant representations and reduce alignment efficiency.

7 Conclusion

In this study, we proposed TiMER, a time-series-language framework that selectively constructs temporal granularity information required by the question intent and effectively connects it with LLMs. Through query-adaptive multi-granularity routing, TiMER dynamically organizes time-series information and aligns it with the LLM input space, enabling effective natural language-based time-series understanding and temporal reasoning. Experiments on the TSAQA benchmark show that TiMER achieves strong performance and generalization compared with various LLM-based baselines using only a small number of trainable alignment modules. In addition, extensive ablation studies verify the effectiveness of each component. This study presents a scalable direction for future natural language-based time-series analysis and temporal reasoning by effectively connecting time-series information with the language understanding and reasoning capabilities of LLMs.

Limitations

This study focuses on the TSQA setting, where the goal is to understand and reason about the characteristics and relationships of time series based on natural language queries. Therefore, although TSAQA includes forecasting-related queries, regression-style forecasting that directly outputs continuous future values, as in conventional time-series forecasting models, is beyond the main scope of this study. In future work, the proposed time-series-language alignment structure can be extended to precise numerical prediction, probabilistic forecasting, and uncertainty-aware prediction.

In addition, although TiMER is designed to train only a small number of alignment modules based on a frozen LLM and a frozen MTE, the architecture and hyperparameters of each module still have room for further optimization. For example, temporal granularity configuration, routing strategy, summary representation length, and alignment token length can affect the trade-off between performance and efficiency. Future work will systematically op-

timize these design factors across more diverse model scales and computational environments, and explore more efficient routing and alignment structures.

Ethical Considerations

This study does not involve human subjects, personally identifiable information, or sensitive private data. We reviewed the dataset documentation and data fields to check for personally identifying information or offensive content. The datasets used in this study do not contain human-subject data, names, unique personal identifiers, or offensive content; therefore, no additional anonymization was required. We use publicly available research datasets and external artifacts, including datasets, pretrained models, and code, in accordance with their respective licenses and terms of use. Our use of these artifacts is consistent with their intended research purposes and original access conditions. We cite the original creators of these artifacts and do not redistribute any third-party artifacts beyond what is permitted by their licenses.

Acknowledgments

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2020-II201373, Artificial Intelligence Graduate School Program (Hanyang University)), the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education (No. RS-2025-25417811), and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2024-0040780).

References

The claude 3 model family: Opus, sonnet, haiku.

Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Bernie Wang. 2024. [Chronos: Learning the language of time series](#). *Transactions on Machine Learning Research*. Expert Certification.

Defu Cao, Furong Jia, Serkan O Arik, Tomas Pfister, Yixiang Zheng, Wen Ye, and Yan Liu. 2024. [TEMPO](#):

[Prompt-based generative pre-trained transformer for time series forecasting](#). In *The Twelfth International Conference on Learning Representations*.

Jialin Chen, Aosong Feng, Ziyu Zhao, Juan Garza, Gaukhar Nurbek, Cheng Qin, Ali Maatouk, Leandros Tassioulas, Yifeng Gao, and Rex Ying. 2026. [Mt-bench: A multimodal time series benchmark for temporal reasoning and question answering](#). *Preprint*, arXiv:2503.16858.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. 2024. A decoder-only foundation model for time-series forecasting. In *Proceedings of the 41st International Conference on Machine Learning, ICML’24*. JMLR.org.

Azul Garza, Cristian Challu, and Max Mergenthaler-Canseco. 2024. [Timegpt-1](#). *Preprint*, arXiv:2310.03589.

Google Gemini Team. 2025. [Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities](#). *Preprint*, arXiv:2507.06261.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. 2023. Large language models are zero-shot time series forecasters. In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS ’23*, Red Hook, NY, USA. Curran Associates Inc.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [LoRA: Low-rank adaptation of large language models](#). In *International Conference on Learning Representations*.

Yushan Jiang, Zijie Pan, Xikun Zhang, Sahil Garg, Anderson Schneider, Yuriy Nevmyvaka, and Dongjin Song. 2024. [Empowering time series analysis with large language models: a survey](#). In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI ’24*.

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

Baoyu Jing, Sanhorn Chen, Lecheng Zheng, Boyu Liu, Zihao Li, Jiaru Zou, Tianxin Wei, Zhining Liu, Zhichen Zeng, Ruizhong Qiu, Xiao Lin, Yuchen Yan, Dongqi Fu, Jingchao Ni, Jingrui He, and Hanghang

- Tong. 2026. [TSAQA: Time series analysis question and answering benchmark](#). In *Proceedings of the Fifth Workshop on Generation, Evaluation and Metrics (GEM)*, pages 944–979, San Diego, California, USA. Association for Computational Linguistics.
- 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](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 29736–29753, Vienna, Austria. Association for Computational Linguistics.
- Hongyu Li, Liang Ding, Meng Fang, and Dacheng Tao. 2024. [Revisiting catastrophic forgetting in large language model tuning](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4297–4308, Miami, Florida, USA. Association for Computational Linguistics.
- Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. 2024. [Foundation models for time series analysis: A tutorial and survey](#). In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '24*, page 6555–6565. ACM.
- Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, Alexandre Sablayrolles, Amélie Hélioü, Amos You, Andy Ehrenberg, Andy Lo, Anton Eliseev, Antonia Calvi, Avinash Sooriyarachchi, Baptiste Bout, and 101 others. 2026. [Ministral 3](#). *Preprint*, arXiv:2601.08584.
- Xu Liu, Junfeng Hu, Yuan Li, Shizhe Diao, Yuxuan Liang, Bryan Hooi, and Roger Zimmermann. 2024. [Unitime: A language-empowered unified model for cross-domain time series forecasting](#). In *Proceedings of the ACM Web Conference 2024, WWW '24*, page 4095–4106, New York, NY, USA. Association for Computing Machinery.
- Mike A Merrill, Mingtian Tan, Vinayak Gupta, Thomas Hartvigsen, and Tim Althoff. 2024. [Language models still struggle to zero-shot reason about time series](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3512–3533, Miami, Florida, USA. Association for Computational Linguistics.
- Yuqi Nie, Yaxuan Kong, Xiaowen Dong, John M. Mulvey, H. Vincent Poor, Qingsong Wen, and Stefan Zohren. 2024. [A survey of large language models for financial applications: Progress, prospects and challenges](#). *Preprint*, arXiv:2406.11903.
- OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, and 401 others. 2024. [Gpt-4o system card](#). *Preprint*, arXiv:2410.21276.
- OpenAI. 2025. [Introducing GPT-4.1 in the API](#). Accessed: 2026-05-21.
- Jinwoo Park, Hyeonwon Kang, Seunghun Han, and Pilsung Kang. 2025. [Granularity fusion transformer: Learning multi-granularity patterns for time-series forecasting](#). *Knowledge-Based Systems*, 320:113644.
- 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.
- Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, and 89 others. 2024. [Gemma: Open models based on gemini research and technology](#). *Preprint*, arXiv:2403.08295.
- 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 *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI'25/IAAI'25/EAAI'25*. AAAI Press.
- Yilin wang, Peixuan Lei, Jie Song, Yuzhe Hao, Tao Chen, Yuxuan Zhang, LEI JIA, Yuanxiang Li, and zhongyu wei. 2025. [ITFormer: Bridging time series and natural language for multi-modal QA with large-scale multitask dataset](#). In *Forty-second International Conference on Machine Learning*.
- 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](#). *Proceedings of the VLDB Endowment*, 18(8):2385–2398.
- Hao Xue and Flora D. Salim. 2024. [Promptcast: A new prompt-based learning paradigm for time series forecasting](#). *IEEE Trans. on Knowl. and Data Eng.*, 36(11):6851–6864.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.

- Yiyuan Yang, Chaoli Zhang, Tian Zhou, Qingsong Wen, and Liang Sun. 2023. [Dcdetector: Dual attention contrastive representation learning for time series anomaly detection](#). In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 3033–3045. ACM.
- Yuan Yuan, Jingtao Ding, Chenyang Shao, Depeng Jin, and Yong Li. 2023. [Spatio-temporal diffusion point processes](#). In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 3173–3184. ACM.
- Zhirui Zhang, Changhua Pei, Tianyi Gao, Zhe Xie, Yibo Hao, Zhaoyang Yu, Longlong Xu, Tong Xiao, Jing Han, and Dan Pei. 2025. [Timesense:making large language models proficient in time-series analysis](#). *Preprint*, arXiv:2511.06344.
- Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, and 3 others. 2026. [A survey of large language models](#). *Preprint*, arXiv:2303.18223.
- Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proc. 39th International Conference on Machine Learning (ICML 2022)*.
- Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. 2023. One fits all: power general time series analysis by pretrained lm. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Contents

A Multi-Granularity Time Encoder	13
A.1 MTE Architecture	13
A.2 Self-supervised Pre-training of MTE	14
B TiMER Training Objective	14
C Experimental Setup Details	15
C.1 TSAQA Dataset	15
C.2 Comparison models	15
C.3 Implementation Details	15
D Qualitative and Routing Analysis	16
D.1 Qualitative Case Study	16
D.2 Query-Adaptive Granularity Routing Analysis	17
E Additional Experimental Results	18
E.1 Domain-wise Performance Analysis	18
E.2 Performance Stability across Random Seeds	18
E.3 Experimental Results Detail . . .	18

A Multi-Granularity Time Encoder

This section describes the architecture and pre-training procedure of MTE, which is used as the pretrained time-series encoder in Section 4.1. MTE is built based on the Granularity Fusion Transformer (GFT) (Park et al., 2025). TiMER uses it as a frozen feature extractor that extracts multi-granularity signal representations from the input time series, without updating it during downstream training. Specifically, before directly learning question-answer pairs, MTE is pretrained through a separate self-supervised prediction task to capture the temporal structure of the time series itself.

A.1 MTE Architecture

MTE takes a variable-length univariate time series X_i as input and generates two token representations with different temporal granularities:

$$H_i^f \in \mathbb{R}^{L_i^f \times d}, \quad H_i^c \in \mathbb{R}^{L_i^c \times d}. \quad (1)$$

where H^f denotes fine-grained tokens that represent short-term fluctuations and local patterns, and H^c denotes coarse-grained tokens that capture long-term trends and global-level changes.

First, RevIN normalization is applied only to valid timesteps, excluding padded positions in the

input time series. This process enables stable statistics to be computed for TSAQA time series with varying lengths. The normalized time series $\hat{x}$ is then decomposed into a coarse trend component τ and a fine residual component r through moving average-based decomposition:

$$\tau = \text{MA}(\hat{x}), \quad r = \hat{x} - \tau. \quad (2)$$

τ represents global-level changes and long-term trends, while r contains fine-grained information such as short-term fluctuations, local patterns, and periodic variations.

For the fine residual component r , MTE performs FFT-based period estimation. Specifically, FFT is applied to r , and dominant frequencies with large amplitudes are selected after excluding the DC component, which represents the mean level. The selected frequencies indicate the dominant repetitive patterns in the residual time series, and the series is reconstructed into a 2D temporal map that reflects the periodic axis. A convolutional upsampling block is then applied to enhance fine temporal resolution while preserving local periodic patterns. In contrast, the coarse trend component τ is aligned to the same temporal resolution as the fine sequence through interpolation.

MTE then tokenizes the fine and coarse sequences using different patch sizes and strides. The fine stream consists of short patches to capture short-term changes and local patterns, while the coarse stream consists of longer patches to capture smooth trends and global-level changes. Each patch is transformed into a d -dimensional token through linear projection, and positional embeddings and RoPE are applied to incorporate positional information. The fine and coarse streams are then passed through separate Transformer encoders:

$$H_i^f = \text{Enc}_f(E_i^f), \quad H_i^c = \text{Enc}_c(E_i^c), \quad (3)$$

where E_i^f and E_i^c denote the fine-grained and coarse-grained patch embeddings, respectively.

During MTE pretraining, a cross-scale fusion layer is additionally used to learn interactions between fine-grained and coarse-grained tokens. This layer performs cross-attention in which coarse tokens attend to fine tokens, producing the fused representation H^{fc} . However, H^{fc} is used only as the input to the pretraining decoder for future time-series prediction and is not directly used in the downstream training stage of TiMER. In the downstream stage, TiMER extracts only H^f and

H^c from MTE, and the subsequent local, global, and cross experts reconstruct the granularity information required under the query condition.

Therefore, instead of compressing the input time series into a single vector, MTE provides multi-granularity token representations that separate local temporal variations and global trends. This structure provides a foundation for selectively utilizing fine-grained or coarse-grained information depending on the nature of the query in the downstream question answering stage.

A.2 Self-supervised Pre-training of MTE

Before being used for downstream TSQA, MTE is pretrained in a self-supervised manner using only time-series data, without question or answer annotations. The objective of pretraining is to enable MTE to represent periodicity, trends, local fluctuations, and cross-granularity interactions in time series before being connected to a language model.

In this study, MTE is pretrained with a variable-horizon forecasting task. Given a time series x of length L , the first 80% of the sequence is used as input, and the remaining future segment is used as the prediction target. MTE generates the fine-grained representation H^f and the coarse-grained representation H^c from the input segment. During pretraining, the fused representation H^{fc} , which combines the two representations, is used as the memory of the decoder to predict the future segment. Since the future prediction length can vary across time series, learnable horizon queries Q_{horizon} are used to generate variable-length future sequences.

The pretraining loss is defined as a masked forecasting loss computed only over valid future timesteps. To handle different prediction lengths within a mini-batch, padded timesteps are excluded from the loss computation. Let $\Omega = \{(i, t) \mid m_{i,t} = 1\}$ denote the set of valid future timesteps, and let $e_{i,t} = \hat{y}_{i,t} - y_{i,t}$ denote the prediction error. The pretraining objective is defined as follows:

$$\mathcal{L}_{\text{pre}} = \frac{1}{|\Omega|} \sum_{(i,t) \in \Omega} (e_{i,t}^2 + \lambda |e_{i,t}|). \quad (4)$$

where $m_{i,t}$ is a binary mask indicating whether the corresponding timestep belongs to the valid future segment, and λ is a coefficient that controls the contribution of the MAE term. In this study, we set $\lambda = 1$. The MSE term strongly penalizes large errors between the predicted and ground-truth values, while the MAE term provides a relatively stable

learning signal against outliers. By combining these two terms, MTE learns both overall prediction accuracy and stable temporal representations.

After pretraining, we select the checkpoint with the lowest validation loss and use it for downstream TiMER training. In the downstream stage, all parameters of MTE are kept frozen, and MTE is used only as a fixed signal encoder that extracts H^f and H^c from each time series. This training separation decouples time-series representation learning from language model alignment, enabling stable use of time-series-specific representations even with a small-scale question-answering training process.

B TiMER Training Objective

TiMER is trained with a causal language modeling objective. The parameters of the LLM and MTE are kept frozen, and only the trainable parameters ϕ in QGR, GTE, and TLA are updated. Let the final LLM input be $\tilde{H}_i^{\text{in}}(\phi)$ and the answer sequence be $a_i = [a_{i,1}, \dots, a_{i,K_i}]$. The alignment loss is computed only over the answer tokens as follows:

$$\mathcal{L}_{\text{align}}(\phi) = -\frac{1}{K_i} \sum_{k=1}^{K_i} \log p_{\Theta}(a_{i,k} \mid \tilde{H}_i^{\text{in}}(\phi), a_{i,<k}), \quad (5)$$

where Θ denotes the parameters of the LLM, K_i is the length of the answer sequence, and $a_{i,<k}$ denotes the token sequence before the k -th answer token. The loss gradient is propagated to the alignment modules through $\tilde{H}_i^{\text{in}}(\phi)$, thereby aligning the time-series representations with the LLM input space. In addition, TiMER uses a balance loss as an auxiliary regularization term to mitigate collapse in QGR, where routing becomes overly biased toward a specific granularity expert. The balance loss encourages the batch-averaged routing distribution to be close to the uniform distribution:

$$\mathcal{L}_{\text{bal}} = \left\| \frac{1}{B} \sum_{b=1}^B p_b - \frac{1}{N_{\text{exp}}} \mathbf{1}_{N_{\text{exp}}} \right\|_2^2, \quad (6)$$

where B is the batch size, N_{exp} is the number of experts, $\mathbf{1}_{N_{\text{exp}}} \in \mathbb{R}^{N_{\text{exp}}}$ is a vector whose elements are all one, and p_b denotes the routing weights of QGR for the b -th sample. The final training objective is therefore defined as follows:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{align}} + \lambda_{\text{bal}} \mathcal{L}_{\text{bal}}, \quad (7)$$

where λ_{bal} is a hyperparameter that controls the strength of balance regularization. Through this

objective, TiMER efficiently aligns time-series representations with the LLM input space while preserving the language understanding and generation capabilities of the LLM and the time-series representation capability of MTE.

C Experimental Setup Details

C.1 TSAQA Dataset

To evaluate the performance of TiMER, we use the Time Series Analysis Question Answering Benchmark (TSAQA) (Jing et al., 2026), a large-scale TSQA benchmark. TSAQA covers 13 domains, 6 time-series analysis tasks, and 3 question types, and contains a total of 210K QA pairs consisting of time-series inputs, natural language queries, and answers (Figure 2). The entire dataset is split into train, validation, and test sets with a ratio of 7:1:2, enabling broad evaluation of query-based time-series understanding and generalization across diverse time-series tasks. The six tasks include anomaly detection (A.D), classification (CLS), characterization, comparison, data transformation, and temporal relationship, while the question types include true/false (TF), multiple-choice (MC), and puzzling (PZ).

In terms of domains, TSAQA includes 13 domains: healthcare, energy, nature, transport, sales, finance, web, biomedical, IT, environmental sensing, manufacturing, synthetic, and robotics. The dataset consists of healthcare with 30,101 samples (14.3%), energy with 27,615 samples (13.2%), nature with 23,485 samples (11.2%), transport with 22,485 samples (10.7%), sales with 21,604 samples (10.3%), finance with 21,419 samples (10.2%), and web with 21,013 samples (10.0%), showing that it is not overly concentrated in a single domain. It also includes relatively smaller domains, such as biomedical, IT, environmental sensing, manufacturing, synthetic, and robotics, allowing evaluation of time-series understanding across diverse real-world application scenarios. This domain composition makes TSAQA suitable for verifying whether TiMER can generalize to time-series question answering problems across diverse domains, rather than being limited to a specific data type or application area.

To further examine the characteristics of the dataset, we analyze the distributions of input time-series length and question token length by task. The length of each input time series ranges from 6 to 345. As shown in Figure A.1, TSAQA ex-

hibits different length distributions depending on the task. Anomaly detection tends to involve relatively short time series and fixed question-length distributions, whereas characterization and comparison are more broadly distributed across different time-series lengths. Classification shows a tendency for samples to concentrate in specific time-series length ranges, while data transformation and temporal relationship include relatively long natural language queries with wide question-length ranges. These distributions show that TSAQA is not limited to fixed-length inputs or a single question format, but instead covers diverse temporal resolutions and question complexities.

C.2 Comparison models

To evaluate time-series understanding and natural language-based reasoning from multiple perspectives, we categorize the comparison models into three groups with different types and scales.

(i) **Proprietary LLMs.** We use GPT-4.1 (OpenAI, 2025), GPT-4o (OpenAI et al., 2024), Claude-3.5-Sonnet (ant), and Gemini-2.5-Flash (Google Gemini Team, 2025). These are API-based large-scale models with strong instruction-following, reasoning, and multimodal understanding capabilities.

(ii) **Open-weight LLMs.** We use models from the LLaMA (Grattafiori et al., 2024), Qwen (Qwen et al., 2025; Yang et al., 2025), Gemma (Team et al., 2024), and Ministral (Liu et al., 2026) families. These are general-purpose LLMs with publicly available weights, covering a wide range of scales from small models under 1B parameters to larger models with 8B, 32B, and 70B parameters.

(iii) **Time-series LLMs.** We use ChatTime (Wang et al., 2025) and ITFormer (wang et al., 2025). These are time-series-specific LLMs that transform or align time-series information into forms that can be processed by LLMs. Through this comparison setup, we comprehensively compare TiMER with large-scale multimodal API models, open-weight LLMs of various scales, and time-series LLMs.

C.3 Implementation Details

In this study, we use Qwen3 as the LLM backbone of TiMER, with the following hyperparameter settings. The hidden dimension of MTE is set to $d = 256$. In GTE, the numbers of summary representations generated through adaptive pooling are set to $n_{\text{loc}} = 8$, $n_{\text{glob}} = 4$, and $n_{\text{cross}} = 8$. The learnable alignment token length of TLA is set to

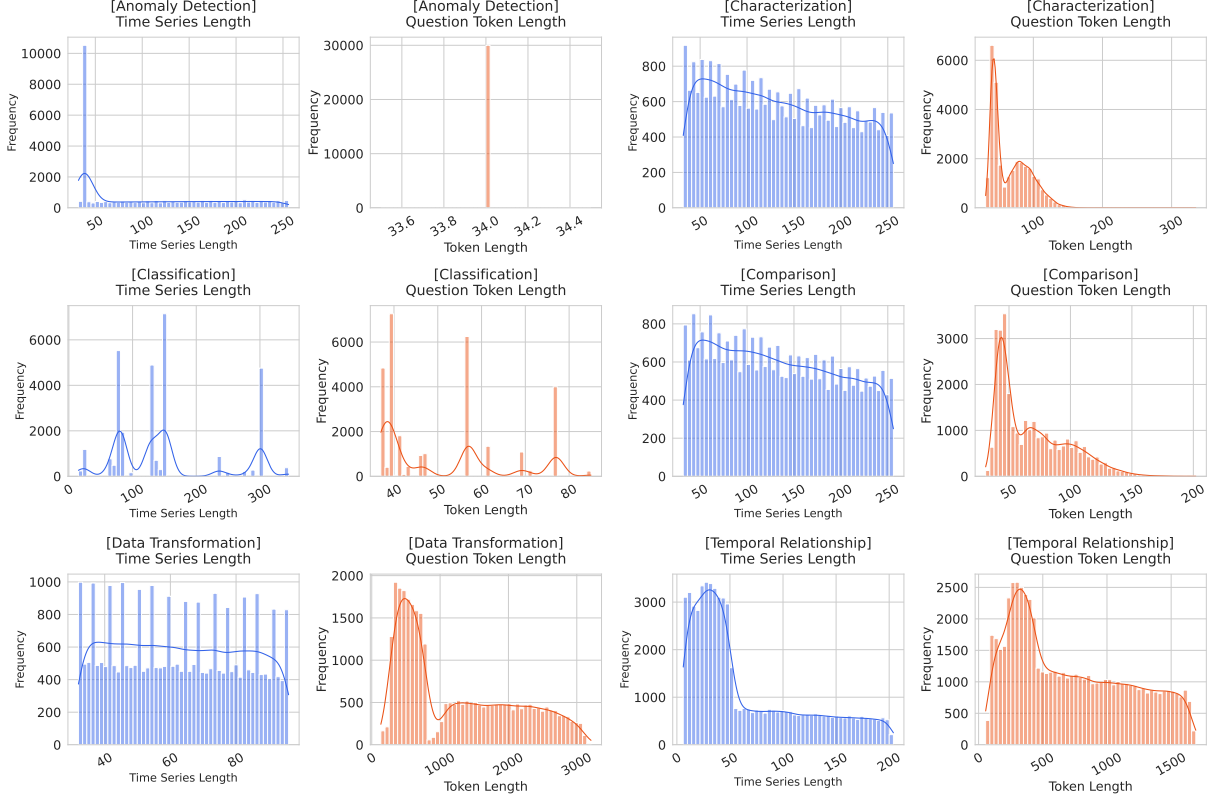


Figure A.1: Task-wise distributions of time series length and question token length in TSAQA. Each pair of subplots shows the distribution of input time series length and corresponding question token length for one task type.

$L_{\text{align}} = 25$, and the weight of the balance loss for mitigating routing collapse toward a specific expert is set to $\lambda_{\text{bal}} = 10^{-3}$. All models are trained for up to 30 epochs, with a gradient accumulation step of 4 and an early stopping patience of 3. The maximum input length is set to $\text{max_seq_len} = 4096$ for all models. The batch size and learning rate are adjusted according to model size: Qwen3-0.6B is trained with a batch size of 8 and a learning rate of 1×10^{-4} , while Qwen3-8B is trained with a batch size of 4 and a learning rate of 5×10^{-5} . The time-series representation is inserted into the LLM input using the special tokens `<signal_start>` and `<signal_end>`, resulting in the following input format:

```
<signal_start>{input_ts}<signal_end>{query}
```

where `{input_ts}` denotes the time-series representation provided to the LLM. During evaluation, all models use deterministic greedy decoding with `do_sample=False`, `temperature=0.0`, `num_beams=1`, and the default repetition penalty. Performance is evaluated using accuracy following the TSAQA evaluation protocol. TF and MC questions are evaluated using exact-match-based 0/1 scores, while PZ questions use the position-wise

matching ratio between the predicted patch order and the ground-truth order as the score. All experiments are conducted on a single NVIDIA H200 GPU with mixed-precision BF16 training.

D Qualitative and Routing Analysis

D.1 Qualitative Case Study

Figure A.2 illustrates how TiMER uses temporal granularity information to interpret time series and generate correct answers. In the first Comparison T/F example, the model needs to determine whether the first of the two time series shows a global upward trend over the entire interval. TiMER correctly predicts that there is no clear global upward trend in the series, while assigning a high weight to global-level information required for trend judgment.

In the second Data Transformation MC example, the model needs to select the most appropriate candidate after applying first-order differencing to the given time series. Since first-order differencing computes changes between adjacent timesteps, local-level information that captures short-term variation patterns is important. At the same time, the model must compare whether each answer can-

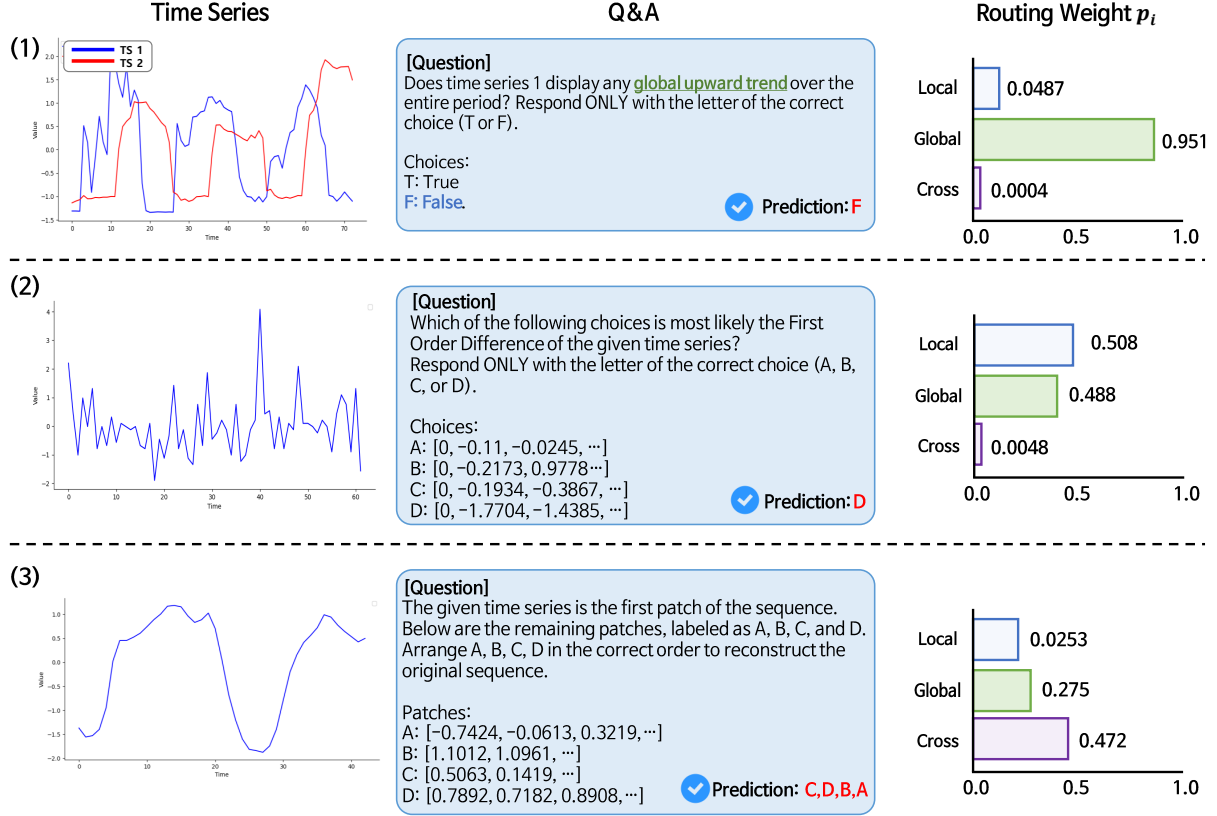


Figure A.2: Qualitative case studies of TiMER on TSAQA. The examples correspond to (1) Comparison T/F, (2) Data Transformation MC, and (3) Temporal Relationship patch PZ. Each example shows the input time series, natural language query, model prediction, and expert routing distribution.

didate is consistent with the overall change, sign, scale, and distributional pattern of the original series, requiring global-level information as well. Accordingly, TiMER assigns high gate values to both the local and global experts, showing that this problem requires considering both local changes and the overall consistency of the transformed pattern.

In the final Temporal Relationship PZ example, the model needs to rearrange the candidate patches in the correct temporal order after the first patch. TiMER correctly predicts the ground-truth order, C, D, B, A, and the cross-granularity expert plays a relatively important role by jointly considering local continuity between patches and the overall sequence structure. These results suggest that TiMER selectively utilizes the temporal granularity required by each question type and task, enabling consistent interpretation across diverse time-series reasoning problems.

D.2 Query-Adaptive Granularity Routing Analysis

Table A.1 shows the average routing weights of the Local, Global, and Cross experts for each task type

Task Type	Local	Global	Cross
Anomaly Detection	0.189	0.001	0.810
Characterization	0.067	0.932	0.001
Classification	0.113	0.077	0.810
Comparison	0.055	0.945	0.001
Data Transformation	0.338	0.660	0.002
Temporal Relationship	0.371	0.379	0.250

Table A.1: Task-wise average routing weights of the Query-Adaptive Granularity Router.

in the Query-Adaptive Granularity Router (Section 4.2). Overall, different experts tend to be selected depending on the reasoning characteristics of each task, indicating that the proposed model adaptively utilizes task-relevant temporal granularity rather than relying on a single expert.

In Anomaly Detection and Classification, the Cross expert receives the highest routing weight. This indicates that these tasks require considering interactions between local temporal variations and global sequence context, rather than focusing only on local patterns. In particular, Anomaly Detection requires determining how much a local deviation

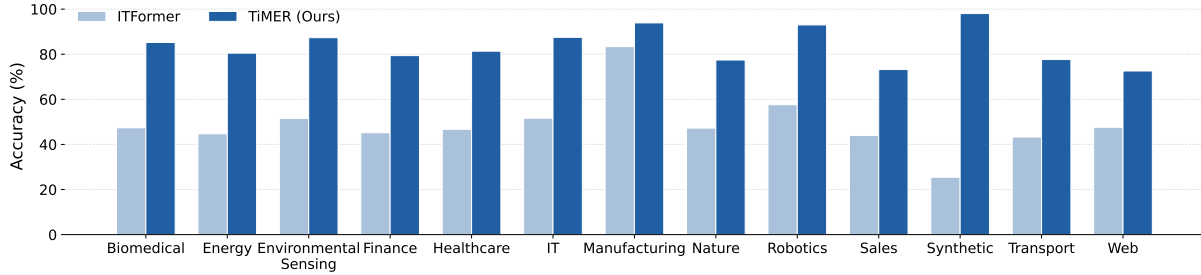


Figure A.3: Domain-wise comparison between ITFormer and TiMER.

differs from the overall temporal flow, while Classification requires both detailed patterns and global time-series structure. Therefore, the Cross expert can be interpreted as playing an important role in these tasks.

In contrast, Characterization and Comparison assign higher weights to the Global expert. Characterization involves identifying overall properties of the time series, such as trend, seasonality, and variance shift, while Comparison requires comparing the overall shapes and trends of two time series. Thus, the Global expert, which provides sequence-level representations, is used more strongly than experts focusing on local changes.

In Data Transformation, both the Global and Local experts tend to be utilized. This is because the model needs to capture the overall change before and after transformation while also considering local pattern changes induced by the transformation. Finally, Temporal Relationship shows relatively balanced routing weights without excessive reliance on a specific expert. This suggests that local, global, and cross-level information are all required to infer temporal order or continuity between patches.

E Additional Experimental Results

E.1 Domain-wise Performance Analysis

Figure A.3 compares the performance of ITFormer and TiMER across 13 domains. TiMER consistently achieves higher accuracy than ITFormer in every domain, demonstrating that its performance gains are not limited to a particular application area. Notably, the largest improvement is observed in the Synthetic domain, where accuracy increases from 25.45 to 98.03. Substantial gains are also observed in Biomedical, Environmental Sensing, and Robotics. Even in Manufacturing, where ITFormer already achieves relatively strong performance, TiMER further improves accuracy from

83.35 to 93.89. These results indicate that the query-conditioned temporal representation mechanism of TiMER generalizes effectively across diverse domains with different time-series characteristics.

E.2 Performance Stability across Random Seeds

To assess the stability and robustness of TiMER with respect to random initialization, we repeated the experiments using three different random seeds (42, 777, and 123). TiMER achieved overall accuracies of 78.45%, 77.07%, and 76.68%, yielding an average accuracy of 77.40%. Despite variations in random initialization, the model maintained consistently strong performance across all runs, with only limited fluctuations in overall accuracy and most individual tasks. These results indicate that the effectiveness of TiMER is not dependent on a favorable random seed, demonstrating stable and reproducible behavior across repeated training runs.

E.3 Experimental Results Detail

In the main paper, we evaluate each analysis mainly in terms of overall performance. This appendix provides the full results across task and question types for the analyses discussed in Section 6.3, including the effect of TiMER components, temporal granularity, granularity summary length, and alignment token length. The results are presented in Tables A.3-A.6.

Seeds	A.D	CLS	Characterization		Comparison		Data Transform		Temporal Relation			Overall
	MC	MC	TF	MC	TF	MC	TF	MC	TF	MC	PZ	
42	85.90	93.82	89.88	74.21	79.93	62.03	85.64	79.42	86.72	80.01	58.75	78.45
777	83.02	91.35	90.31	75.91	80.14	64.09	83.91	75.48	85.6	76.91	57.01	77.07
123	84.15	91.22	90.34	74.94	80.58	64.45	83.31	76.42	84.82	76.81	54.33	76.68
Average	84.36	92.13	90.18	75.02	80.22	63.52	84.29	77.11	85.71	77.91	56.70	77.40

Table A.2: Model performance across different tasks and seeds.

Components	A.D	CLS	Characterization		Comparison		Data Transform		Temporal Relation			Overall
	MC	MC	TF	MC	TF	MC	TF	MC	TF	MC	PZ	
MTE	80.83	86.57	62.97	35.07	60.44	42.36	51.87	25.45	53.21	25.33	26.51	52.59
+ GTE	<u>84.62</u>	<u>91.87</u>	79.09	40.44	73.65	43.90	52.03	26.42	56.86	25.79	23.09	56.09
+ GTE + QGR	83.30	89.28	82.12	44.64	73.75	45.27	63.16	27.45	68.73	32.50	28.56	58.96
+ TLA	76.50	85.17	89.01	<u>74.71</u>	78.95	61.01	80.31	70.45	81.20	71.21	44.22	71.25
+ GTE + TLA	83.60	88.55	88.54	75.48	79.05	66.15	<u>83.13</u>	<u>74.02</u>	<u>84.53</u>	<u>75.69</u>	<u>47.57</u>	<u>74.60</u>
TiMER	85.90	93.82	<u>89.88</u>	74.21	<u>79.93</u>	<u>62.03</u>	85.64	79.42	86.72	80.01	58.75	78.45

Table A.3: Results for the effect of TiMER components on TSAQA.

Temporal Granularity	A.D	CLS	Characterization		Comparison		Data Transform		Temporal Relation			Overall
	MC	MC	TF	MC	TF	MC	TF	MC	TF	MC	PZ	
Raw	64.85	61.57	87.58	72.81	78.51	<u>61.73</u>	64.92	47.20	76.46	67.90	41.08	62.25
Fine	77.57	76.02	84.22	63.43	75.72	53.81	54.03	26.82	63.86	30.21	24.14	56.61
Coarse	<u>78.29</u>	<u>87.87</u>	87.81	73.04	78.81	59.41	74.52	60.11	74.09	61.14	42.30	69.24
Cross	76.50	85.17	<u>89.01</u>	74.71	<u>78.95</u>	61.01	<u>80.31</u>	<u>70.45</u>	<u>81.20</u>	<u>71.21</u>	<u>44.22</u>	<u>71.25</u>
Ours	85.90	93.82	89.88	<u>74.21</u>	79.93	62.03	85.64	79.42	86.72	80.01	58.75	78.45

Table A.4: Results for the effect of temporal granularity on TSAQA.

Summary Length			A.D	CLS	Characterization		Comparison		Data Transform		Temporal Relation			Overall
n_{local}	n_{glob}	n_{cross}	MC	MC	TF	MC	TF	MC	TF	MC	TF	MC	PZ	
4	2	4	79.77	87.95	89.81	76.74	80.92	62.42	82.51	73.45	83.71	74.11	<u>51.91</u>	<u>74.64</u>
8	2	8	84.18	90.55	89.41	<u>75.48</u>	79.90	61.60	77.95	66.78	81.81	69.53	49.86	73.89
8	4	8	85.90	93.82	89.88	74.21	79.93	62.03	85.64	79.42	86.72	80.01	58.75	78.45
16	8	16	80.93	89.02	<u>89.84</u>	75.44	<u>80.17</u>	<u>63.76</u>	<u>84.58</u>	<u>76.18</u>	<u>86.38</u>	<u>77.01</u>	48.61	74.09
w/o Pooling			73.67	88.32	89.18	65.50	79.80	65.60	83.04	74.42	82.68	73.70	48.26	73.15

Table A.5: Results for the effect of granularity summary length on TSAQA.

Token Length	A.D	CLS	Characterization		Comparison		Data Transform		Temporal Relation			Overall
	MC	MC	TF	MC	TF	MC	TF	MC	TF	MC	PZ	
15	81.78	92.00	88.78	<u>75.48</u>	79.32	64.94	83.94	<u>76.38</u>	83.95	75.69	48.07	75.08
20	<u>85.12</u>	<u>91.93</u>	90.21	75.84	80.85	<u>63.60</u>	82.84	74.72	<u>86.09</u>	<u>77.52</u>	<u>54.67</u>	<u>76.93</u>
25	85.90	93.82	<u>89.88</u>	74.21	79.93	62.03	85.64	79.42	86.72	80.01	<u>58.75</u>	78.45
30	76.53	86.92	89.41	75.71	<u>80.27</u>	62.13	<u>84.71</u>	75.18	84.58	74.72	51.51	74.14

Table A.6: Results for the effect of granularity summary length on TSAQA.