DOES TIME-SERIES QUESTION ANSWERING NEED A LARGE LANGUAGE MODEL TO REASON?

Anonymous authors
Paper under double-blind review

ABSTRACT

Does time-series question answering need a large language model (LLM) as its reasoning engine? Many existing methods rely on an LLM both to interpret the signal and to reason over it. This is computationally expensive and makes it difficult to determine whether better answers come from better representations or better reasoning. We introduce GLEAN (**G**rounded **L**atent **E**vidence **A**ggregation **N**etwork), which separates pretrained input encoding from a dedicated recurrent reasoning engine. Frozen text and time-series encoders process each input once, supplemented by numerical features that retain details such as signal endpoints and slopes. GLEAN then reasons over these representations through recurrent latent updates. Its current state guides each new read of the inputs, so relevant details can be revisited and combined as the answer develops. A small shared network performs this computation, and structured output heads select answers or order signal fragments without generating text. The separation makes each part measurable: frozen encoders alone already lead every time-series LLM on perception questions, while the recurrent reasoner raises relational accuracy from 58.03% to 69.16% on the same representations. With 0.46B total parameters, of which only 8.2M are trained, GLEAN achieves 77.33% accuracy on TSAQA, outperforming all five evaluated fine-tuned time-series LLMs we evaluated while using 87–693$\times$ fewer estimated inference FLOPs per question. These results suggest that closed-ended temporal question answering does not require a large language model as its reasoning engine: pretrained representations coupled with a small recurrent network offer a strong and far cheaper alternative.

1 INTRODUCTION

Time series record how systems evolve, from physiological signals to industrial sensors. Beyond forecasting, users increasingly ask questions about them, e.g., whether a signal contains an anomaly, which of two signals is more volatile, or how fragments should be ordered (Merrill et al., 2024; Cai et al., 2024; Jing et al., 2026). Answering requires two abilities, i.e., *perceiving* the relevant properties of a signal and *reasoning* over them in light of a question posed in language. The question decides which properties matter: the same recording may need to be examined for a local anomaly, a global trend, or compatibility with a candidate continuation.

Existing approaches typically make a large language model (LLM) responsible for both. Text-based methods serialize values into digits or value tokens (Gruver et al., 2023; Wang et al., 2025a; Guan et al., 2026); embedding-based methods encode the series separately and insert or align its features with the text (Jin et al., 2024; Xie et al., 2025; Lu et al., 2026). In both, the LLM generates the answer (Figure 1a). This has two costs. Every question passes a long multimodal prompt through a multi-billion-parameter backbone; and because perception and reasoning are routed through a single generated answer, it is difficult to determine whether a better answer comes from a better representation of the signal or from better reasoning over it. Recent work strengthens the LLM itself by treating the series as a new modality (Xie et al., 2025) or by reinforcing explicit reasoning (Guan et al., 2026), and leaves this entanglement in place.

Several findings question whether the LLM is where this capability should reside. Zero-shot LLMs reason about time series barely above chance and up to 30 points below humans (Merrill et al., 2024), and in forecasting, removing the LLM from LLM-based forecasters or replacing it with a sin-

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

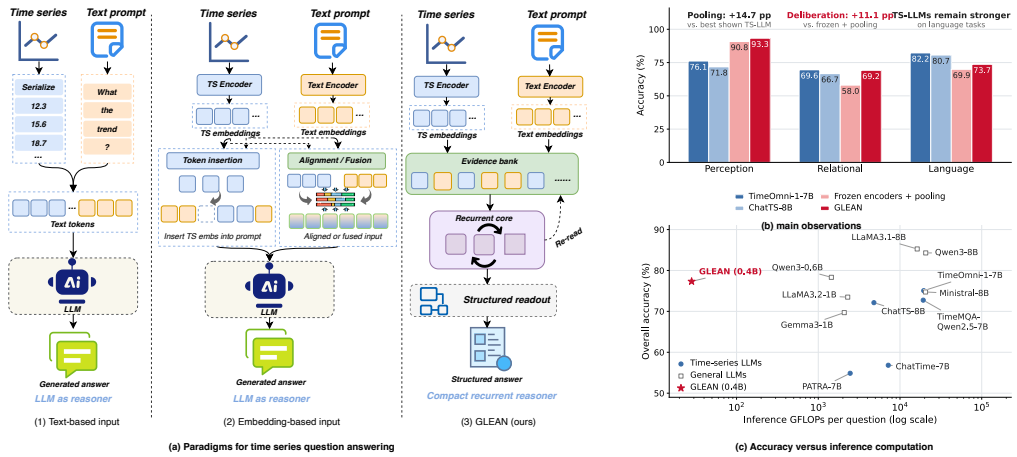


Figure 1: **Motivation.** (a) Existing paradigms serialize the series into text (left) or insert encoder embeddings into the prompt (middle), and in both the LLM answers by generating text; GLEAN (right) perceives once with frozen encoders and lets a compact recurrent core deliberate over an evidence bank before a structured readout. (b) Accuracy by question group: frozen encoders with a pooled readout lead on perception relative to the shown baselines; deliberation improves relational accuracy. (c) Overall accuracy against inference GFLOPs per question on the 42,000 test questions.

gle attention layer does not reduce accuracy (Tan et al., 2024). On TSAQA, zero-shot LLMs score 49–56% on true/false anomaly detection, near chance, yet up to 93% on characterization questions that the question text largely answers on its own (§3.3). Forecasting, however, involves no language; question answering does, which makes it the setting where an LLM reasoning engine should matter most. Meanwhile, small recursive networks rival LLMs on structured reasoning over fixed answer spaces (Jolicoeur-Martineau, 2025), and recursive latent computation can replace token-level chains of thought (Panwar et al., 2026). This raises our central question: *does time-series question answering need a large language model as its reasoning engine?*

Separating perception from reasoning makes each measurable. Frozen time-series and text encoders with a pooled readout and no reasoning module reach 90.83% on TSAQA's perception questions, at least 14.7 points above every time-series LLM we evaluate, yet only 58.03% on relational questions that require comparing contexts, candidates, or fragments (Figure 1b). Strong representations are thus available; what is missing is a way to reason over them. Such a reasoner must retain access to individual observations, since a later comparison may need a different part of the signal; integrate retrieved evidence across contexts, candidates and fragments; preserve exact values that normalization and patching can obscure; and respect the structure of the answer, since an ordering must use each fragment exactly once.

We introduce GLEAN (**G**rounded **L**atent **E**vidence **A**ggregation **N**etwork, Figure 1a, right), which separates pretrained perception from a dedicated recurrent reasoning engine. Frozen encoders build a token-level evidence bank once per instance, supplemented by boundary features that retain raw values such as endpoints and slopes. A small shared network then refines latent answer slots over repeated steps: the current state guides each new read of the bank, and slot interaction combines what is retrieved. Content matching selects answers and a Sinkhorn-based assignment head orders fragments, so no text is generated. GLEAN outperforms all five fine-tuned time-series LLMs at 87–693× fewer FLOPs per question. General-purpose LLMs fine-tuned on TSAQA remain more accurate overall, at 51–733× the computation, and we report where and why they lead (§3.2).

Our contributions are:

- We separate perception from reasoning and use this separation to identify a gap in frozen representations: a pooled readout already leads time-series LLMs on perception, while a recurrent reasoner substantially raises relational accuracy using the same frozen features.
- We introduce an evidence-grounded recurrent reasoner with value-sensitive inputs and structured readouts, training 8.2M parameters while keeping both encoders frozen.

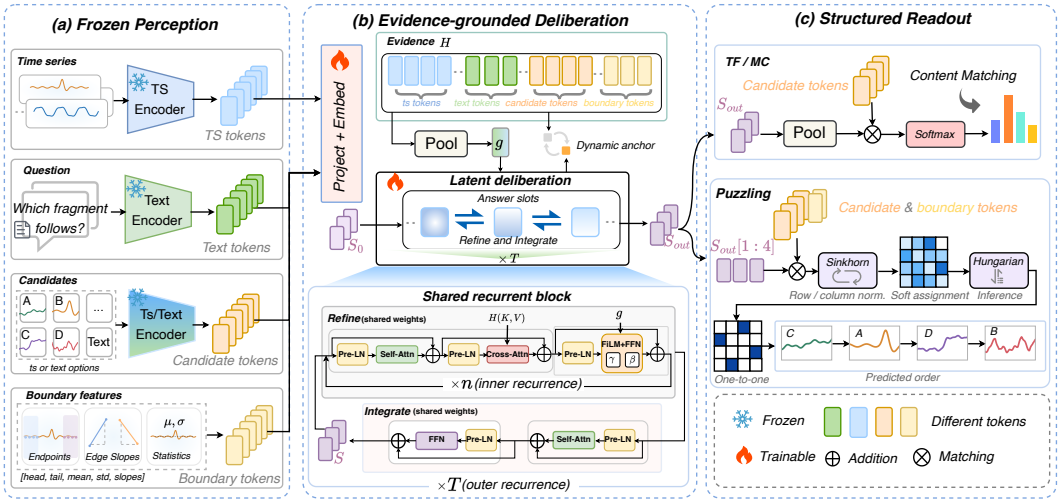


Figure 2: **Overview of GLEAN. (a)** Frozen encoders and raw-value boundary features form an evidence bank computed once per instance. **(b)** A shared recurrent core alternates evidence-guided *Refine* steps with *Integrate* steps on latent answer slots. **(c)** Content matching selects candidates; Sinkhorn assignment and Hungarian decoding order fragments. Encoders are frozen; projections, embeddings, core and readouts are trained.

- On TSAQA, GLEAN outperforms five time-series LLMs under a common fine-tuning protocol with 87–693× fewer FLOPs per question and the highest perception accuracy among the compared systems. Complementary ablations and encoder and schedule sweeps characterize its mechanisms and limitations.

2 Methodology

2.1 Problem Formulation

An instance is a tuple $(\mathcal{X}, q, \mathcal{A}, y^\star)$. The context $\mathcal{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(C)}\}$ contains C series with $\mathbf{x}^{(c)} \in \mathbb{R}^{L_c}$, q is a question in natural language, $\mathcal{A}$ is a structured answer space, and $y^\star$ is the correct answer. Two answer spaces cover the benchmark we study. In *selection* questions (true/false and multiple choice), $\mathcal{A} = \{a_1, \ldots, a_m\}$ and $y^\star \in \{1, \ldots, m\}$, where a candidate is either a short text or, for numeric questions, a short series. In *ordering* questions, $\mathcal{A}$ is the set of permutations of n_p fragments $\{\mathbf{p}_1, \ldots, \mathbf{p}_{n_p}\}$, and $y^\star = \pi^\star$ assigns one fragment to each chronological position.

We learn a single scoring function $f_\theta(\mathcal{X}, q, \mathcal{A})$ over $\mathcal{A}$ for all tasks and formats, with the task specified by language and the readout matched to the answer structure (Figure 2). Throughout, ϕ_{ts} and ϕ_{tx} denote pretrained time-series and text encoders whose parameters remain *frozen*, and θ collects only the parameters introduced below.

2.2 Perceive Once: The Evidence Bank

The first stage converts every input into tokens that the reasoner can address individually (Figure 2a).

Frozen perception at token level. The time-series encoder maps the context to patch tokens and the text encoder maps the question to token states,

$$\mathbf{Z}_{\text{ts}} = \phi_{\text{ts}}(\mathcal{X}) \in \mathbb{R}^{P \times d_{\text{ts}}}, \qquad \mathbf{Z}_q = \phi_{\text{tx}}(q) \in \mathbb{R}^{L_q \times d_{\text{tx}}}. \tag{1}$$

Textual candidates yield pooled vectors from ϕ_{tx}. Numeric candidates and ordering fragments instead pass through ϕ_{ts}, yielding both pooled vectors for the readout and patch tokens $\mathbf{Z}_{\text{cand}}$ for the bank. Textual candidates do not add patch tokens (Algorithm 1).

Boundary features. Normalization and patch encoding can obscure exact values needed to compare fragment boundaries. For every raw series $\mathbf{x} \in \mathbb{R}^L$, whether a context channel or an ordering

fragment, we therefore compute a descriptor *before* any padding or normalization,

$$\mathbf{b}(\mathbf{x}) = \big[\, x_{1:4} \,\|\, x_{L-3:L} \,\|\, \mu(\mathbf{x}) \,\|\, \sigma(\mathbf{x}) \,\|\, x_2 - x_1 \,\|\, x_L - x_{L-1} \,\big] \in \mathbb{R}^{12}, \tag{2}$$

which records endpoint values, level, spread and endpoint slopes. Each descriptor becomes one *boundary token*; the matrix $\mathbf{B}$ stacks them.

Assembling the bank. Each source is mapped to the working width d by its own projection $\mathrm{Proj}(\cdot) = \mathrm{LN}(\mathbf{W}\,\cdot)$, and the projected tokens are concatenated along one axis:

$$\mathbf{H}_0 = \big[\, \mathrm{Proj}_{\mathrm{ts}}(\mathbf{Z}_{\mathrm{ts}}) \parallel \mathrm{Proj}_{q}(\mathbf{Z}_{q}) \parallel \mathrm{Proj}_{\mathrm{c}}(\mathbf{Z}_{\mathrm{cand}}) \parallel \mathrm{Proj}_{\mathrm{b}}(\mathbf{B}) \,\big] \in \mathbb{R}^{L_H \times d}. \tag{3}$$

A shared axis requires every token to state where it comes from, so we add a modality embedding (series, text, candidate), a positional embedding, and a segment embedding naming the context channel or candidate it belongs to, which keeps tokens of different series distinguishable after concatenation. A task embedding is broadcast over the sequence, and a global condition is pooled from the result:

$$\mathbf{H} = \mathbf{H}_0 + \mathbf{E}_{\mathrm{mod}} + \mathbf{E}_{\mathrm{pos}} + \mathbf{E}_{\mathrm{seg}} + \mathbf{1}\,\mathbf{e}_{\mathrm{task}}^{\top}, \qquad \mathbf{g} = \mathrm{AttnPool}(\mathbf{H}) \in \mathbb{R}^{d}. \tag{4}$$

Evidence and answers are kept apart. The answer space is projected once into the same width: a candidate matrix $\mathbf{C} \in \mathbb{R}^{m \times d}$ for selection, and a fragment matrix $\mathbf{P} \in \mathbb{R}^{n_p \times d}$ for ordering whose rows add the boundary token of each fragment to its projected pooled representation. The bank $\mathbf{H}$ holds what the model may *look at*; $\mathbf{C}$ and $\mathbf{P}$ hold what counts as an *answer*. This separation makes the modality interventions of Section 3.3 clean, since they remove tokens from $\mathbf{H}$ while leaving the answer space unchanged.

2.3 Evidence-Grounded Deliberation: the Recurrent Core

With the evidence in place, the second stage reasons over it (Figure 2b). Reasoning operates on K latent answer slots $\mathbf{S} \in \mathbb{R}^{K \times d}$, initialized from a learned embedding $\mathbf{S}_0$ that gives each slot a distinct identity. The same slots read the evidence and are read out at the end, so retrieval and integration act on a single state.

Shared recurrent block. The core alternates two weight-shared residual steps. The *Refine* step relates the slots to one another, re-reads the entire bank, and transforms the result under the global condition:

$$\mathbf{S} \leftarrow \mathbf{S} + \mathrm{SelfAttn}(\mathbf{S}), \tag{5}$$

$$\mathbf{S} \leftarrow \mathbf{S} + \mathrm{CrossAttn}(\mathbf{S},\, \mathbf{H}), \tag{6}$$

$$\mathbf{S} \leftarrow \mathbf{S} + \mathrm{FFN}\big(\mathrm{FiLM}_{\mathbf{g}}(\mathrm{LN}(\mathbf{S}))\big), \qquad \mathrm{FiLM}_{\mathbf{g}}(\mathbf{h}) = \boldsymbol{\gamma}(\mathbf{g}) \odot \mathbf{h} + \boldsymbol{\beta}(\mathbf{g}). \tag{7}$$

Here $\boldsymbol{\gamma}(\cdot)$ and $\boldsymbol{\beta}(\cdot)$ are linear maps of the global condition, initialized to $\boldsymbol{\gamma} \equiv \mathbf{1}$ and $\boldsymbol{\beta} \equiv \mathbf{0}$ so modulation starts as the identity. *Integrate* step consolidates the slots without consulting the evidence:

$$\mathbf{S} \leftarrow \mathbf{S} + \mathrm{SelfAttn}'(\mathbf{S}), \qquad \mathbf{S} \leftarrow \mathbf{S} + \mathrm{FFN}'(\mathbf{S}). \tag{8}$$

Every attention layer normalizes its queries and its keys and values separately before attending, and every FFN is a two-layer GELU network with its own pre-normalization. One pass of the core runs T outer loops, each consisting of n Refine steps followed by one Integrate step, and hence applies $T(n{+}1)$ residual blocks with a single set of parameters. Depth is obtained by iteration rather than by parameters.

Evidence access and integration. Cross-attention provides a *dynamic anchor*: each step queries the full bank using the current slots. Unlike reasoning solely over a pooled summary, this preserves access to token-level evidence throughout refinement (Appendix G). Because queries depend on the current slots, successive reads can emphasize different evidence without rerunning the encoders. Slot self-attention then integrates retrieved evidence. For example, evaluating a continuation may require comparing the context's endpoint with the candidate's beginning: retrieving both is useful only if their relationship can affect the answer. Slot interaction provides a mechanism for such comparisons without prescribing a fixed semantic role for each slot. The Integrate step updates this joint state without another bank read, separating internal consolidation from new evidence access. Section 3.3 tests these mechanisms separately by withholding bank access (*static anchor*) or slot interaction (*access-only reader*).

2.4 Answer in Structure: Structured Readouts

Refined slots $\mathbf{S}_{\text{out}}$ are then mapped to the answer space using format-specific readouts (Figure 2c).

Selection by content matching. For true/false and multiple-choice questions, a query is pooled from the slots and matched against each candidate,

$$\mathbf{u} = \text{AttnPool}(\mathbf{S}_{\text{out}}), \qquad \ell_i = \frac{\langle \mathbf{u}, \mathbf{c}_i \rangle}{\sqrt{d}}, \qquad p(a_i \mid \mathcal{X}, q) = \text{softmax}_i(\ell), \tag{9}$$

with absent candidates masked. For fixed slots, reordering the candidate vectors reorders the logits and preserves the relabeled loss (Proposition 1). This property concerns the readout; it does not guarantee invariance to reordering candidate tokens in the upstream evidence bank.

Ordering as a soft permutation. An ordering assigns each fragment to exactly one position. An independent per-position head cannot express non-degenerate uncertainty while assigning probability only to valid permutations (Proposition 2). Two positions can otherwise prefer the same fragment, leaving another unused. A final matching algorithm can resolve this conflict, but training the positions independently does not expose their competition through the normalization itself.

GLEAN instead couples the positions. The first n_p slots act as position queries and form the score matrix $M_{jk} = \langle \mathbf{s}_j, \mathbf{p}_k \rangle / \sqrt{d}$, which is relaxed toward the Birkhoff polytope by I log-domain Sinkhorn iterations at temperature τ (Sinkhorn & Knopp, 1967; Mena et al., 2018), alternating row and column normalization:

$$\log \Gamma \leftarrow M/\tau, \quad \log \Gamma \leftarrow \log \Gamma - \operatorname*{logsumexp}_k \log \Gamma, \quad \log \Gamma \leftarrow \log \Gamma - \operatorname*{logsumexp}_j \log \Gamma. \tag{10}$$

The limiting matrix is doubly stochastic and hence a convex combination of permutation matrices (Corollary 1); finite iterations approximate this constraint. Training minimizes $\mathcal{L}_{\text{ord}} = -\frac{1}{n_p} \sum_j \log \Gamma_{j,\pi^\star(j)}$, coupling positions through normalization. This is a per-position objective, not an exact likelihood over permutations. At inference, Hungarian decoding (Kuhn, 1955) returns a valid permutation, $\hat{\pi} = \arg\max_\pi \sum_j \log \Gamma_{j,\pi(j)}$. The two operations serve distinct purposes: Sinkhorn introduces competition during learning, while Hungarian decoding guarantees that the final answer satisfies the assignment constraint.

2.5 Optimization

The core is unrolled for N_{seg} segments. After each segment the readout is applied and contributes a loss, and the slots are detached before entering the next segment, $\mathbf{S}^{(s)} = \text{Core}\big(\text{sg}[\mathbf{S}^{(s-1)}], \mathbf{H}, \mathbf{g}\big)$. Gradients therefore flow through all $T(n{+}1)$ blocks *within* a segment but never across segments, and each segment receives task supervision. A light auxiliary head predicts whether the current answer is already correct, $h^{(s)} = \sigma\big(\mathbf{w}^\top \text{AttnPool}(\mathbf{S}^{(s)}) + b\big)$, trained with binary cross-entropy against the realized correctness. The objective averages over segments,

$$\mathcal{L} = \frac{1}{N_{\text{seg}}} \sum_{s=1}^{N_{\text{seg}}} \Big(\mathcal{L}_{\text{task}}^{(s)} + \lambda \, \mathcal{L}_{\text{aux}}^{(s)} \Big), \tag{11}$$

where $\mathcal{L}_{\text{task}}$ is the selection cross-entropy or $\mathcal{L}_{\text{ord}}$. We set $\lambda = 0.1$ and use the final segment's answer after running the full schedule; the auxiliary head does not trigger early stopping. Pseudocode and implementation details are in Appendices D and E.

3 Experiments

3.1 Experimental Setup

Benchmark. We evaluate on TSAQA (Jing et al., 2026), which pairs series from 13 domains with questions in six tasks and three answer formats (true/false, multiple choice and ordering), giving eleven task–format cells. We use the official splits and report accuracy on the full test split of $42{,}000$ questions, aggregated as a sample-weighted mean over cells; ordering is scored by per-position

Table 1: **Accuracy (%) on the full TSAQA test set.** Cells are grouped into perception (AD, CLS), relational (DT, TR) and language-heavy (Char, Comp); the overall score is the sample-weighted average over all cells. General-purpose LLM results are those reported by Jing et al. (2026); time-series LLMs are fine-tuned under our protocol. **Blue**: best among time-series reasoning methods.

Model	Perception		Relational					Language-heavy				Overall
	AD TF	CLS MC	DT TF	DT MC	TR TF	TR MC	TR PZ	Char TF	Char MC	Comp TF	Comp MC	
General-purpose LLMs: zero-shot												
GPT-4.1	55.85	50.38	54.36	51.13	65.90	79.09	45.77	92.97	89.36	83.57	76.99	62.82
GPT-4o	54.32	47.20	60.66	53.24	62.25	75.58	45.61	88.15	84.15	78.61	69.07	60.73
Claude-3.5-Sonnet	51.27	41.23	65.79	57.07	82.05	82.15	54.56	74.39	78.45	66.59	74.14	61.19
Gemini-2.5-Flash	52.08	49.07	63.62	60.17	75.05	84.49	60.84	85.48	81.08	77.79	72.21	65.08
Qwen3-8B	50.60	50.52	52.43	34.46	65.22	67.14	21.93	77.35	66.87	71.04	63.21	51.04
LLaMA3.1-8B	54.92	50.20	51.90	36.56	54.82	40.95	6.80	68.10	62.26	67.84	49.98	44.93
Ministral-8B	53.35	34.08	50.70	25.28	50.58	33.88	30.77	71.06	63.93	47.54	52.90	44.65
Qwen3-0.6B	50.40	35.83	49.03	23.62	51.99	37.33	13.38	62.00	48.78	58.03	37.51	39.06
LLaMA3.2-1B	49.47	39.48	48.87	4.20	48.97	5.44	6.76	63.74	52.55	61.02	36.82	35.70
Gemma3-1B	49.15	49.83	49.37	24.88	49.42	25.84	23.97	63.74	47.71	61.19	43.37	43.03
General-purpose LLMs: instruction-tuned												
Qwen3-8B	87.70	90.05	89.84	84.99	96.84	97.56	66.21	92.37	85.42	86.55	79.08	84.29
LLaMA3.1-8B	91.02	91.27	90.17	86.62	96.94	97.41	67.68	92.44	83.68	86.72	79.31	85.26
Ministral-8B	71.56	74.28	75.15	71.61	94.07	94.15	56.82	91.31	80.78	84.14	74.63	74.74
Qwen3-0.6B	83.68	85.78	80.51	73.28	93.92	93.79	63.34	89.38	74.87	80.65	64.84	78.32
LLaMA3.2-1B	83.08	83.83	68.09	51.67	91.39	88.81	57.53	87.71	74.37	78.61	60.88	73.48
Gemma3-1B	83.10	84.05	64.06	45.23	91.00	88.05	42.92	87.88	72.54	78.61	59.31	69.70
Time-series reasoning models												
TimeOmni-1-7B	76.82	75.47	71.69	66.64	**94.60**	**92.78**	57.68	89.81	**81.45**	84.21	**73.68**	75.06
TimeMQA-7B	79.82	69.42	66.46	60.44	94.26	90.34	56.60	88.88	78.91	82.28	68.90	72.74
ChatTS-8B	78.70	64.88	73.22	68.61	85.55	82.66	54.68	89.28	78.85	82.61	72.05	72.13
ChatTime-1-7B	73.37	77.63	54.36	23.85	58.61	26.60	29.31	84.12	64.63	76.71	54.83	56.83
PATRA-7B	58.33	60.45	51.00	24.75	50.63	26.09	28.39	**90.48**	80.88	**84.75**	71.72	54.87
GLEAN (Ours)	**93.40**	**93.12**	**78.45**	**79.49**	79.18	74.52	**57.87**	88.21	69.64	78.78	58.36	**77.33**

agreement, as in the benchmark. To read results by competence, we group the tasks by their definition into *perception* (anomaly detection, classification), *relational* (data transformation, temporal relationship) and *language-heavy* (characterization, comparison) questions, again weighting by sample count. Dataset statistics and the aggregation rule are given in Appendix B.

Baselines. We compare with three groups. *Zero-shot* results of four commercial and six open LLMs, and *instruction-tuned* results of the six open LLMs, are quoted from the benchmark paper. Our main comparison is with five recent *time-series LLMs* that cover both input paradigms of Figure 1: TimeOmni-1-7B (Guan et al., 2026), TimeMQA-Qwen2.5-7B (Kong et al., 2025a), ChatTime-1-7B (Wang et al., 2025a), ChatTS-8B (Xie et al., 2025) and PATRA-7B (Lu et al., 2026). We fine-tune all five on the TSAQA training split under one protocol that follows the benchmark, and score them with the same parsing and aggregation code as GLEAN. Details are given in Appendix C.

Implementation. Unless stated otherwise, GLEAN uses a frozen MOMENT-1-large time-series encoder (Goswami et al., 2024) and a frozen BGE-base text encoder (Xiao et al., 2024), width d=384, K=12 answer slots and the schedule $(N_{\text{seg}}, T, n) = (2, 3, 4)$. One joint model is trained for 30k steps on all tasks, updating 8.2M of the 0.46B parameters it holds at inference. Hyperparameters and the protocol for measuring cost are given in Appendices E and F.

3.2 Main Results

Accuracy and perception. GLEAN reaches 77.33% overall accuracy (Table 1), exceeding the strongest time-series LLM, TimeOmni-1-7B (75.06%), and the other four by 4.6–22.5 points. It achieves the highest perception accuracy among all compared systems (93.26%), including instruction-tuned LLaMA3.1-8B (91.14%), and leads the time-series LLMs on both data-transformation cells. It also exceeds every zero-shot model by at least 12.2 points overall.

Table 2: **Ablation study on TSAQA** (accuracy, %). Variants are grouped by the design choice they test; Δ is the change in overall accuracy relative to the full model, and the last column lists the most affected cells. Per-cell results in Appendix H.

	Variant	Perc.	Rel.	Lang.	Overall (Δ)	Most affected cells
Sources	*text only*	53.58	37.16	70.86	51.48 (-25.85)	AD -42.5, DT MC -47.8
	series only	92.22	66.11	60.36	71.93 (-5.40)	Char TF -25.7, Comp TF -21.6
Core	*pooled readout, no core*	90.83	58.03	69.89	70.79 (-6.54)	PZ -14.9, DT TF -12.4
	static anchor	90.75	64.93	72.57	74.50 (-2.83)	DT TF -8.8, DT MC -8.4
	access-only reader	92.67	65.79	74.00	75.81 (-1.52)	DT TF -6.5, TR TF -5.5
	no recursion (T=n=1)	92.24	65.74	72.88	75.35 (-1.98)	DT TF -4.7, PZ -4.5
	truncated gradients	92.66	64.97	73.99	75.46 (-1.87)	PZ -6.9, DT TF -3.6
Bank	*no numeric-candidate encoding*	93.03	63.94	73.47	74.98 (-2.35)	DT MC -20.5, TR MC -8.2
	no boundary or segment IDs	92.05	60.98	74.16	73.62 (-3.71)	PZ -12.6, DT TF -7.4
	segment IDs only	92.14	60.99	74.74	73.82 (-3.51)	PZ -12.8, DT TF -8.1
	boundary features only	92.88	67.63	73.78	76.60 (-0.73)	PZ -2.5, TR MC -1.8
	series read as text	87.53	57.82	71.89	70.32 (-7.01)	DT MC -18.0, PZ -12.3
Readout	*no Sinkhorn coupling*	85.65	54.22	72.68	68.47 (-8.86)	PZ -19.2, DT TF -15.2
	argmax decoding	92.86	66.99	73.48	76.24 (-1.09)	PZ -3.6, DT TF -3.0
	GLEAN (ours)	93.26	69.16	73.66	77.33	

Efficiency and remaining gaps. Including both frozen encoders and the recurrent core, GLEAN costs 28.2 GFLOPs per question, using 87–693$\times$ fewer FLOPs and 15–18$\times$ fewer total parameters than the five time-series LLMs (Appendix F). Three general LLMs score higher overall (Qwen3-0.6B, Qwen3-8B and LLaMA3.1-8B), but each requires 51–733$\times$ more computation (Figure 1c). The remaining gaps concern exact numeric continuation and language-heavy multiple-choice questions; Appendix J examines their relation to the frozen representations.

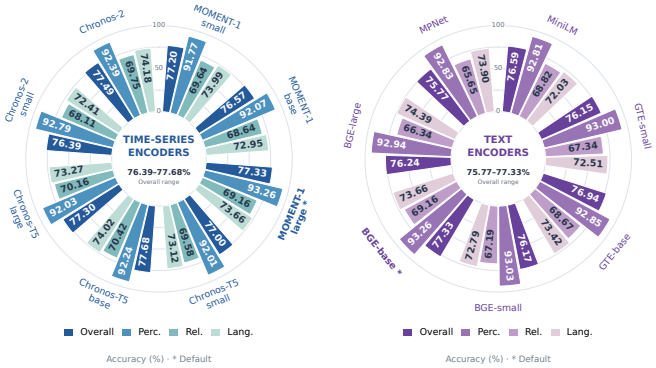
3.3 Ablation Studies

Each ablation removes one design choice and retrains the model under the same protocol (Table 2; per-cell results in Appendix H).

Combining the Two Evidence Sources. If the core combines evidence rather than relying on one source, removing either source should hurt, and hurt different questions. Text only drops accuracy to 51.48, collapsing perception and relational accuracy while language-heavy questions are largely retained; series only, with the question and the task identity withheld, keeps perception at 92.22 but drops language-heavy questions to 60.36. The full model is best on every group and exceeds the better single source by 5.4 points, so both sources are used.

Evidence-Grounded Deliberation. Complementary ablations test evidence access and slot interaction (§2.3): a pooled readout without a core reaches 70.79, a core that sees only the pooled condition g (*static anchor*) 74.50, a core that re-reads the bank but whose slots never attend to one another (*access-only reader*) 75.81, and the full core 77.33. The two intermediate arms fail on different cells: removing re-reading costs most on data transformation, where the answer depends on comparing specific values, whereas removing slot interaction costs most on the relational cells and leaves the language-heavy cells within 0.61 points. The two arms are complementary rather than successive stages of one nested architecture: the static anchor keeps slot interaction but removes repeated bank access, the access-only reader the reverse. Both also change active capacity, so they do not isolate connectivity alone (Appendix H); their distinct error patterns nonetheless support retaining both mechanisms. Four blocks that re-read the bank (75.35) outperform 30 blocks without re-reading (74.50), and back-propagating through all blocks within a segment matters mainly for ordering (-6.90 on TR PZ).

Token-Level Evidence and Boundary Features. Two choices in the bank target the values that frozen encoders abstract away. Encoding numeric candidates as signals rather than text is worth 2.35 points overall and 20.52 on DT MC, the cell whose candidates are numbers; boundary features carry most of the structural gain, since removing them together with segment identities costs 3.71



(a) (b)

Figure 3: **Frozen-encoder sensitivity on TSAQA.** Overall and question-group accuracy on a shared 0–100 radial scale. Asterisks mark defaults; centers show the overall range. (**a**) Time-series encoders, with BGE-base fixed. (**b**) Text encoders, with MOMENT-1-large fixed.

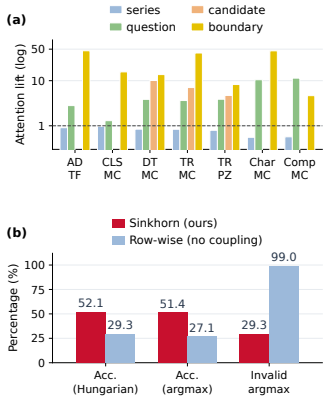


Figure 4: **(a)** Attention lift by token type (dashed: uniform). **(b)** Per-position accuracy and invalid argmax rate, with and without Sinkhorn coupling.

points, keeping only the segment identities recovers almost nothing (73.82) and keeping only the boundary features recovers most of it (76.60). The two channels act on different cells and are nearly additive (joint removal: 71.64). Replacing the time-series encoder by the frozen text encoder, which reads each patch as serialized numbers while everything else stays fixed, costs 7.01 points, most of them on relational questions (57.82 against 69.16) and 1.77 on language-heavy ones: perceiving the signal as text is not sufficient, and a dedicated encoder matters most where the answer turns on comparing values and shapes.

Structured Readouts. Replacing the Sinkhorn coupling by independent per-position classifiers, with the same parameters and scores, lowers the TR PZ cell by 19.21 points. Because this arm also degrades cells without an ordering head (for example −7.7 on AD TF), part of the drop likely reflects less stable joint training, so we read the ordering cell rather than the overall score (68.47). Decoding by per-position argmax instead of the Hungarian algorithm costs 3.64 points on ordering in a separately trained model. Six task-specific models reach 78.69 (Appendix H).

3.4 Model Analysis

We test how much the result depends on the frozen encoders and the recurrence schedule, and inspect what the core reads; cost accounting and further analyses are in the Appendix.

3.4.1 Frozen Encoders

Time-series encoders. With BGE-base fixed, overall accuracy ranges from 76.39% to 77.68% across eight encoders, with no monotonic relation to model size (Figure 3a). All retain strong perception (91.77–93.26%), although TR MC varies substantially: the alternatives reach 86.27–87.95% versus 74.52% for MOMENT-1-large (Table 7), a gap we discuss in Appendix J.

Text encoders. With MOMENT-1-large fixed, overall accuracy spans 75.77–77.33% across seven encoders (Figure 3b). BGE-base gives the highest overall and relational scores; BGE-large improves language-heavy accuracy but lowers overall accuracy to 76.24%. Every pairing exceeds the strongest time-series LLM overall. Full results are in Appendix I.

3.4.2 Visualizing Deliberation

What the slots read. We measure attention *lift*: a token type's share of cross-attention divided by its share of the bank, averaged over Refine steps on a stratified sample of 300 questions per cell (Figure 4a). A lift of one denotes uniform reading. Question-token lift is higher on language-heavy cells (10.5–15.9) than on perception cells (1.3–2.8), indicating question-dependent evidence use.

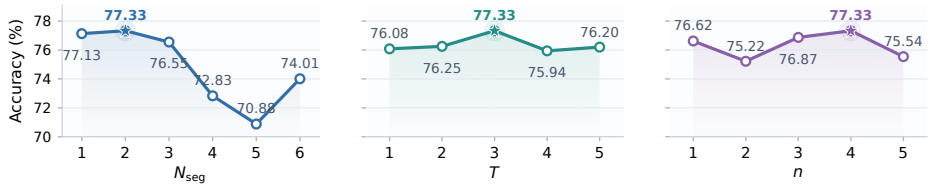


Figure 5: **Schedule sensitivity.** Overall accuracy when one of N_{seg}, T and n is varied with the other two at their defaults. Panels share a vertical scale; full results in Appendix K.

Boundary tokens attract high attention even where their removal does not hurt accuracy; attention alone therefore does not establish importance. Appendix L gives per-cell values.

What the coupling changes. We analyze the 5,978 full-reordering questions; Table 1's TR PZ score also includes 2,000 selection questions, so the two accuracies are not directly comparable (Appendix B). On this subset, Sinkhorn reaches 52.1% per-position accuracy versus 29.4% with row-wise normalization (Figure 4b), and invalid argmax assignments fall from 98.9% to 29.3% while Hungarian decoding adds only 0.7 points. Coupling therefore helps during training, and Hungarian decoding guarantees validity; exact-match accuracy also rises from 8.0% to 33.5%, so the benefit extends to complete fragment orders and not only to individual positions. Appendix L reports the error breakdowns.

3.4.3 Recurrence Schedule

The schedule (N_{seg}, T, n) sets how many residual blocks the core applies without changing its parameters (Figure 5). Accuracy varies within 75.22–77.33% when the number of outer loops or Refine steps changes around the default, and all eleven near-default schedules remain above the strongest time-series LLM. Depth is best spent inside a segment: with a single segment, growing the core from 2 to 15 blocks raises accuracy from 75.59 to 77.13%, whereas adding supervision segments beyond three is harmful (72.83, 70.88 and 74.01% for four to six).

4 Related Work

Time-series question answering typically supplies serialized values or encoder embeddings to an LLM for evidence integration and answer generation (Wang et al., 2025a; Guan et al., 2026; Xie et al., 2025). GLEAN instead performs these steps with a compact recurrent core and structured readouts. Latent cross-attention is established by Perceiver (Jaegle et al., 2021; 2022), and learned queries also connect frozen encoders to LLMs (Li et al., 2023). We use the latent state as the reasoner itself, repeatedly accessing a separate time-series–text evidence bank. Shared recurrence builds on compact latent reasoning (Wang et al., 2025b; Jolicoeur-Martineau, 2025); our focus is how evidence access and integration support language-specified temporal reasoning. Content matching and Sinkhorn assignment (Mena et al., 2018) are established operators, not new contributions; Appendix A extends this discussion.

5 Conclusion

In this work, we introduced GLEAN, a compact framework that separates frozen perception from evidence-grounded reasoning in time-series question answering. A recurrent core repeatedly accesses and integrates token-level evidence, while structured readouts produce answers without text generation. With 8.2M trainable parameters, GLEAN achieves 77.33% accuracy on TSAQA, outperforming five fine-tuned time-series LLMs with 87–693× fewer FLOPs per question. These results support reasoning directly over frozen evidence as an efficient alternative to an LLM backbone in this setting. Nevertheless, exact numeric continuation and language-heavy questions remain limitations, and the answer spaces we study are closed. These findings therefore do not establish sufficiency for unrestricted language-based reasoning. Future work should improve numerical fidelity and language representations, extend the readouts beyond closed answer spaces, and evaluate transfer across datasets and to open-ended questions.

AI USE STATEMENT

The authors developed the research questions, methodology and experimental design, and take responsibility for the scientific claims and interpretation of the results. Generative AI tools solely assisted in improving the clarity and readability of the manuscript. The authors reviewed and verified the resulting code and text, and retain full responsibility for the accuracy and integrity of this work.

ETHICS STATEMENT

This work uses TSAQA, a public benchmark of time series and closed-ended questions, and publicly released pretrained encoders and language models; it involves no human subjects, no personal data and no newly collected data. Time-series reasoning systems may be applied in medicine, finance and industrial monitoring, where an incorrect answer carries real cost, so we report accuracy by question type rather than as a single aggregate and state explicitly where our model is weaker than language models. Our results should not be read as evidence that such a system is ready for deployment in those domains without task-specific validation.

REPRODUCIBILITY STATEMENT

The method is specified in §2, with every operator written out, stated as pseudocode in Appendix D together with the edit that defines each ablation, and with the default configuration in Appendix E, which also describes batch construction, optimization and model selection. The proofs of all formal statements are in Appendix G. The benchmark, its splits and the scoring rule, including the composition of the ordering cell, are described in Appendix B; the baselines, their fine-tuning protocol and their input interfaces are in Appendix C; and the cost accounting, including the FLOP convention, is in Appendix F. Per-cell results for every ablation and every encoder pairing are reported in Appendices H and I so that each number in the main text can be traced. Source code for the model, the evaluation harness and the analysis scripts will be released.

REFERENCES

Ryan Prescott Adams and Richard S Zemel. Ranking via Sinkhorn propagation. *arXiv preprint arXiv:1106.1925*, 2011.

Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. Don't just assume; look and answer: Overcoming priors for visual question answering. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 4971–4980, 2018.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob L. Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karén Simonyan. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

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

ARC Prize Team. The hidden drivers of HRM's performance on ARC-AGI. `https://arcprize.org/blog/hrm-analysis`, 2025. Accessed 2026-09-16.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 688–699, 2019.

Andrea Banino, Jan Balaguer, and Charles Blundell. PonderNet: Learning to ponder. In *8th ICML Workshop on Automated Machine Learning (AutoML 2021)*, 2021.

Yifu Cai, Arjun Choudhry, Mononito Goswami, and Artur Dubrawski. TimeSeriesExam: A time series understanding exam. In *TSALM Workshop at the Conference on Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End object detection with Transformers. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, volume 12346 of *Lecture Notes in Computer Science*, pp. 213–229, 2020.

Ching Chang, Yidan Shi, Defu Cao, Wei Yang, Jeehyun Hwang, Haixin Wang, Jiacheng Pang, Wei Wang, Yan Liu, Wen-Chih Peng, and Tien-Fu Chen. A survey of reasoning and agentic systems in time series with large language models. *Transactions on Machine Learning Research (TMLR)*, 2026, 2026.

Mingyue Cheng, Jiahao Wang, Daoyu Wang, Xiaoyu Tao, Qi Liu, and Enhong Chen. Can slow-thinking LLMs reason over time? empirical studies in time series forecasting. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining*, WSDM '26, pp. 99–110, New York, NY, USA, 2026.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10148–10167, 2024.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal Transformers. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A family of open time-series foundation models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 16115–16152, 2024.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 6325–6334, 2017.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Tong Guan, Zijie Meng, Dianqi Li, Shiyu Wang, Chao-Han Huck Yang, Qingsong Wen, Zuozhu Liu, Sabato Siniscalchi, Ming Jin, and Shirui Pan. TimeOmni-1: Incentivizing complex reasoning with time series in large language models. In *International Conference on Learning Representations*, volume 2026, pp. 152139–152170, 2026.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Proceedings of the 2025 Conference on Language Modeling (COLM)*, 2025.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and João Carreira. Perceiver: General perception with iterative attention. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4651–4664, 2021.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J. Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver IO: A general architecture for structured inputs & outputs. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time series forecasting by reprogramming large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

Baoyu Jing, Sanhorn Chen, Lecheng Zheng, Boyu Liu, Zihao Li, Jiaru Zou, Tianxin Wei, Zhining Liu, Zhichen Zeng, Ruizhong Qiu, et al. TSAQA: Time series analysis question and answering benchmark. In *Proceedings of the Fifth Workshop on Generation, Evaluation and Metrics (GEM)*, pp. 944–979, 2026.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Yaxuan Kong, Yiyuan Yang, Yoontae Hwang, Wenjie Du, Stefan Zohren, Zhangyang Wang, Ming Jin, and Qingsong Wen. 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), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 29736–29753, 2025a.

Yaxuan Kong, Yiyuan Yang, Shiyu Wang, Chenghao Liu, Yuxuan Liang, Ming Jin, Stefan Zohren, Dan Pei, Yan Liu, and Qingsong Wen. Achieving time series reasoning requires rethinking model design, tasks formulation, and evaluation. *arXiv preprint arXiv:2502.01477*, 2025b.

Harold W Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19730–19742, 2023.

Haoxin Liu, Chenghao Liu, and B Aditya Prakash. A picture is worth a thousand numbers: Enabling LLMs reason about time series via visualization. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7486–7518, 2025.

Junkai Lu, Peng Chen, Xingjian Wu, Yang Shu, Chenjuan Guo, Christian S Jensen, and Bin Yang. PATRA: Pattern-aware alignment and balanced reasoning for time series question answering. In *Forty-third International Conference on Machine Learning, ICML 2026, Seoul, South Korea, July 6-11, 2026*, Proceedings of Machine Learning Research, pp. 1–21, 2026.

Gonzalo E. Mena, David Belanger, Scott W. Linderman, and Jasper Snoek. Learning latent permutations with Gumbel-Sinkhorn networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

Mike A. Merrill, Mingtian Tan, Vinayak Gupta, Thomas Hartvigsen, and Tim Althoff. Language models still struggle to zero-shot reason about time series. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, volume EMNLP 2024 of *Findings of ACL*, pp. 3512–3533, 2024.

Abhishek Panwar, Maheep Singh, and Saksham Bansal. Think deep, speak once: Relit, a recursive latent implicit Transformer framework. In *Workshop on Latent & Implicit Thinking – Going Beyond CoT Reasoning at the International Conference on Learning Representations (ICLR)*, 2026.

Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

Mingtian Tan, Mike A. Merrill, Vinayak Gupta, Tim Althoff, and Tom Hartvigsen. Are language models actually useful for time series forecasting? In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Chengsen Wang, Qi Qi, Jingyu Wang, Haifeng Sun, Zirui Zhuang, Jinming Wu, Lei Zhang, and Jianxin Liao. ChatTime: A unified multimodal time series foundation model bridging numerical and textual data. In *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pp. 12694–12702, 2025a.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025b.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53140–53164, 2024.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-Pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pp. 641–649, 2024.

Zhe Xie, Zeyan Li, Xiao He, Longlong Xu, Xidao Wen, Tieying Zhang, Jianjun Chen, Rui Shi, and Dan Pei. ChatTS: Aligning time series with LLMs via synthetic data for enhanced understanding and reasoning. *Proc. VLDB Endow.*, 18(8):2385–2398, 2025.

Junru Zhang, Lang Feng, Xu Guo, Yuhan Wu, Yabo Dong, and Duanqing Xu. TimeMaster: Training time-series multimodal LLMs to reason via reinforcement learning. In *BERT2S workshop at the Conference on Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. One fits all: Power general time series analysis by pretrained LM. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

APPENDIX

A EXTENDED RELATED WORK

This section expands §4. It covers five lines of work at greater length, and states for each one what we take from it and where we depart.

Time-series foundation models. Pretraining on large corpora of signals has produced general-purpose encoders that transfer across forecasting, classification and anomaly detection (Goswami et al., 2024; Ansari et al., 2024; Das et al., 2024; Woo et al., 2024). These models are strong perceivers, patching, normalizing and embedding raw measurements, but they answer a fixed set of numerical queries and expose no interface for questions stated in language, so adapting them to a new task usually means attaching a head and fine-tuning the backbone. We take the opposite stance. The encoders stay frozen and are used once per instance as a source of token-level evidence, so what is learned is a way of combining evidence rather than a better representation.

Language models as time-series reasoners. The dominant route to language-specified reasoning over signals places a language model at the center, and the approaches differ mainly in how the signal reaches it. Some serialize values into digit text and prompt the model directly (Gruver et al., 2023), optionally with slow-thinking inference (Cheng et al., 2026) or with reinforcement learning that elicits explicit rationales (Guan et al., 2026; Zhang et al., 2025; Kong et al., 2025a). Others extend the vocabulary with discretized value tokens (Wang et al., 2025a). A third group encodes the series with a dedicated encoder and inserts or aligns the resulting embeddings in the input space of the model (Xie et al., 2025; Lu et al., 2026; Jin et al., 2024; Zhou et al., 2023), and a fourth renders the series as a plot for a multimodal model (Liu et al., 2025). Different as these interfaces are, they delegate evidence integration and answer generation to the language model, even when a separate encoder handles signal perception. Processing the resulting input through a large backbone contributes substantial cost. We instead keep both encoders frozen and learn a compact module to combine their evidence without generating text.

What time-series reasoning benchmarks measure. A growing set of benchmarks targets this capability, from synthetic examinations to multi-task question answering and large unified suites (Cai et al., 2024; Kong et al., 2025a; Liu et al., 2025; Jing et al., 2026), alongside surveys and position papers that ask for evaluation beyond aggregate accuracy (Chang et al., 2026; Kong et al., 2025b). Several analyses question what such scores reflect: zero-shot language models answer no better with the series than without it (Merrill et al., 2024), and the language component of LLM-based forecasters can be removed without loss of accuracy (Tan et al., 2024). The concern is familiar from visual question answering, where much of the task turned out to be solvable from question priors, which motivated balanced datasets and changing-prior splits (Goyal et al., 2017; Agrawal et al., 2018). We take this as a reason to report accuracy by competence rather than as a single number: we group questions into perception, relational and language-heavy, and verify with modality controls that the groups behave as their names suggest (§3.3). The grouping is a reading aid for our own analysis, not a claim about the benchmark's construction.

Iterative and latent reasoning. Reasoning by repeatedly transforming a latent state, rather than by emitting tokens, has a long lineage: weight-tied recurrence in Transformers (Dehghani et al., 2019), adaptive computation and halting (Graves, 2016; Banino et al., 2021), and equilibrium formulations that iterate to a fixed point (Bai et al., 2019). Recent small recursive models revive the idea for structured puzzles with a few million parameters (Wang et al., 2025b; Jolicoeur-Martineau, 2025), and an independent replication attributes much of the gain to the outer refinement loop and to deep supervision rather than to the hierarchical architecture (ARC Prize Team, 2025). In parallel, latent chain-of-thought methods let language models think in continuous space instead of words (Hao et al., 2025). Two differences characterize our setting. First, these methods reason over inputs that *are* the state, a grid or a prompt, whereas our evidence is external, multimodal and far larger than the answer, so the design question is how iteration stays anchored to it; our ablations show that re-reading the evidence matters more than depth (§3.3). Second, we find that iterating within a supervised segment helps while stacking further segments does not (§3.4.3), so we make no claim that depth monotonically buys accuracy.

Querying frozen encoders, and structured readouts. Architecturally, our recurrent core belongs to the family of latent arrays that cross-attend a fixed input, introduced by Perceiver and Perceiver IO (Jaegle et al., 2021; 2022) and widely used to connect frozen encoders to downstream models through a small set of learned queries (Alayrac et al., 2022; Li et al., 2023). We adopt this interface but use it differently: instead of compressing perception into a fixed-length prefix that a language model then consumes, the latent array is itself the reasoner, and it re-reads the full evidence at every step. On the output side we treat answering as structured prediction. Scoring candidates by content is standard in retrieval-style readouts and is permutation equivariant for a fixed query, while ordering is handled with a Sinkhorn relaxation of the assignment problem (Adams & Zemel, 2011; Mena et al., 2018) decoded by the Hungarian algorithm (Kuhn, 1955), related to the bipartite matching used in set prediction for detection (Carion et al., 2020). Our contribution is not these operators but the finding that aligning the readout with the structure of each answer format is necessary for a compact reasoner to be competitive, and that with the coupled relaxation the soft assignment is already close to a permutation before any hard decoding (§3.4.2).

B Benchmark Details

TSAQA (Jing et al., 2026) pairs time series from 13 domains with closed-ended questions in six tasks and three answer formats, and provides official train, validation and test splits. We train on the training split, select checkpoints on the validation split, and report the official test split of $42{,}000$ questions. Table 3 lists the eleven task-format cells and their sizes.

The six tasks. *Anomaly detection* asks whether a recording contains an anomalous segment, given a domain description of what counts as anomalous. *Classification* asks which class a recording belongs to, with the candidate classes listed in the question. *Characterization* asks about a property of a single series, such as its trend, seasonality or volatility. *Comparison* contrasts two series on such a property, for instance which of them is more volatile. *Data transformation* asks which operation maps an input series to an output, or which output a stated operation produces; its multiple-choice candidates are themselves numeric series. *Temporal relationship* asks how fragments relate in time: whether a fragment continues a context (true/false), which of four fragments continues it (multiple choice), or in which order four fragments occur (the ordering format).

Scoring. True/false and multiple-choice cells are scored by accuracy. The ordering cell (TR PZ) is scored by per-position agreement with the ground-truth permutation, as defined by the benchmark. This cell is not homogeneous: of its $7{,}978$ questions, $5{,}978$ ask for a full reordering of four fragments, $1{,}013$ ask which fragment continues the context in multiple-choice form, and 987 ask the same in true/false form. GLEAN routes the first group to the Sinkhorn readout and the other two to the selection readout, and all three contribute to the TR PZ score, for every system. The reordering analyses of §3.4.2 cover the first group only, which is why their accuracy is not the TR PZ cell score.

Aggregation. Every sample receives a score $s_i \in [0, 1]$: exact match for true/false and multiple choice, so that $s_i \in \{0, 1\}$, and the per-position match rate for ordering, so that $s_i \in \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$ with four fragments. The overall score is the mean over all samples,

$$\text{Overall} \;=\; \frac{1}{N}\sum_{i=1}^{N} s_i, \qquad N = 42{,}000, \tag{12}$$

which follows the protocol of the benchmark, averaging over samples rather than over cells (Jing et al., 2026). Because each cell score is itself the mean over that cell's samples, this is identical to the mean of the eleven cell scores weighted by cell size, so the overall column of Table 1 can be recomputed from the per-cell values. The two aggregations are not interchangeable: a macro average over cells would give TR MC ($1{,}966$ samples) the same weight as TR PZ ($7{,}978$). The three question groups are aggregated in the same way and cover $12{,}000$ samples for perception (AD TF, CLS MC), $18{,}000$ for relational (DT TF, DT MC, TR TF, TR MC, TR PZ) and $12{,}000$ for language-heavy (Char TF, Char MC, Comp TF, Comp MC). Relational questions therefore carry 43% of the overall score, and the ordering cell alone carries 19%.

Table 3: The eleven task-format cells of the TSAQA test split. TF: true/false; MC: multiple choice; PZ: ordering, scored per position.

Cell	Task	n	Cell	Task	n
AD TF	Anomaly detection	6,000	TR PZ	Temporal relationship	7,978
CLS MC	Classification	6,000	Char TF	Characterization	3,003
DT TF	Data transformation	3,002	Char MC	Characterization	2,997
DT MC	Data transformation	2,998	Comp TF	Comparison	2,945
TR TF	Temporal relationship	2,056	Comp MC	Comparison	3,055
TR MC	Temporal relationship	1,966			

C BASELINES

Quoted results. Zero-shot results of four commercial and six open LLMs, and instruction-tuned results of the six open LLMs, are the values published by Jing et al. (2026). We never mix them with our own runs in the same block of Table 1.

Time-series LLMs we fine-tune. The five time-series LLMs are fine-tuned on the TSAQA training split with one protocol: LoRA of rank 16 on the attention projections, two epochs, effective batch size 128, learning rate 1×10^{-5} with a cosine schedule, bf16, single A100. Time-MQA is distributed as an adapter over a quantized base model, so we merge it into a full-precision checkpoint before fine-tuning.

Each model receives the benchmark prompt through its own input interface, and training and evaluation use the same interface: chat templates for TimeOmni-1 and ChatTS, the native `<QUE>/<ANS>` format for Time-MQA, discretized value tokens for ChatTime, and the time-series encoder of ChatTS and PATRA for their embedded series. The series values are always those of the benchmark field that the text baselines also read, so no model sees more of the signal than another. Responses are parsed and aggregated by the same code for every system, including GLEAN.

Reproduction of ChatTS and ChatTime. The benchmark paper reports 30.10 for ChatTS and 55.19 for ChatTime. Under our protocol both score higher (72.13 and 56.83); we report our values, which is the conservative choice for GLEAN. We attribute the difference mainly to the input interface, since both models require their own processor and answer in their own format.

D ALGORITHMS

This section states GLEAN as pseudocode, in the notation of §2. Algorithm 1 builds the evidence bank of §2.2, Algorithm 2 is one segment of the recurrent core of §2.3, Algorithm 3 gives the two structured readouts of §2.4, and Algorithm 4 assembles them into the training loop of §2.5. The division of labor is visible in where θ appears: the encoder outputs and the boundary descriptors of Algorithm 1 are a function of the frozen encoders alone and are computed once per instance and cached, while everything that follows them is a function of the trained parameters.

Every attention normalizes its query and its key-value inputs separately before attending, and every FFN carries its own pre-normalization, so the layer normalizations left implicit in Algorithm 2 are those of the main text. The cost of one segment is linear in the length of $\mathbf{H}$ through the cross-attention and quadratic only in K through the self-attention.

A single row normalization, $\log \Gamma \leftarrow \log \Gamma - \mathrm{logsumexp}_k \log \Gamma$ applied once, is exactly the first half of the first Sinkhorn iteration and is also exactly a log-softmax over candidates for each position independently. It is therefore the parameter-matched factorized head of Proposition 2, and it is the arm reported as "no Sinkhorn coupling" in §3.3: the same score matrix and the same per-position loss, with the column normalization and the iteration removed.

Inference. Inference runs the same loop without the loss terms. All N_{seg} segments are executed and the answer of the last one is reported; the auxiliary head is recorded but never used to stop early, so no reported number depends on it. Because nothing is produced token by token, the cost of a

Algorithm 1 Perceive once: building the evidence bank

Require: context $\mathcal{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(C)}\}$, question q, answer space $\mathcal{A}$, task identity t
Require: frozen encoders $\phi_{\text{ts}}, \phi_{\text{tx}}$; projections and embeddings, all in θ
$\mathbf{Z}_{\text{ts}} \leftarrow \phi_{\text{ts}}(\mathcal{X})$ ▷ P patch tokens
$\mathbf{Z}_q \leftarrow \phi_{\text{tx}}(q)$ ▷ L_q token states
if $\mathcal{A} = \{a_1, \ldots, a_m\}$ is a candidate set **then** ▷ selection
  if the candidates are numeric series **then**
    $(\mathbf{C}_{\text{pool}}, \mathbf{Z}_{\text{cand}}) \leftarrow \phi_{\text{ts}}(\mathcal{A})$ ▷ pooled vectors and patch tokens
  else
    $\mathbf{C}_{\text{pool}} \leftarrow \phi_{\text{tx}}(\mathcal{A}), \quad \mathbf{Z}_{\text{cand}} \leftarrow \varnothing$
  end if
else ▷ ordering: $\mathcal{A}$ is the set of permutations of $\mathbf{p}_1, \ldots, \mathbf{p}_{n_p}$
  $(\mathbf{P}_{\text{pool}}, \mathbf{Z}_{\text{cand}}) \leftarrow \phi_{\text{ts}}(\{\mathbf{p}_k\}_{k=1}^{n_p})$
end if
for every raw series $\mathbf{x}$ among the context channels and the fragments **do**
  $\mathbf{b}(\mathbf{x}) \leftarrow \big[\, x_{1:4} \,\|\, x_{L-3:L} \,\|\, \mu(\mathbf{x}) \,\|\, \sigma(\mathbf{x}) \,\|\, x_2 - x_1 \,\|\, x_L - x_{L-1} \,\big]$ ▷ before padding or normalization
end for
$\mathbf{B} \leftarrow$ the boundary descriptors, stacked ▷ everything above depends only on $\phi_{\text{ts}}, \phi_{\text{tx}}$ and is cached
$\mathbf{H}_0 \leftarrow \big[\, \text{Proj}_{\text{ts}}(\mathbf{Z}_{\text{ts}}) \,\|\, \text{Proj}_q(\mathbf{Z}_q) \,\|\, \text{Proj}_{\text{c}}(\mathbf{Z}_{\text{cand}}) \,\|\, \text{Proj}_{\text{b}}(\mathbf{B}) \,\big]$
$\mathbf{H} \leftarrow \mathbf{H}_0 + \mathbf{E}_{\text{mod}} + \mathbf{E}_{\text{pos}} + \mathbf{E}_{\text{seg}} + \mathbf{1}\,\mathbf{e}_t^\top$ ▷ source, position, channel or candidate, task
$\mathbf{g} \leftarrow \text{AttnPool}(\mathbf{H})$ ▷ global condition for the FiLM modulation
$\mathbf{C} \leftarrow \text{Proj}_{\text{opt}}(\mathbf{C}_{\text{pool}}), \quad \text{or} \quad \mathbf{p}_k \leftarrow \text{Proj}_{\text{patch}}(\mathbf{P}_{\text{pool},k}) + \text{Proj}_{\text{b}}(\mathbf{b}(\mathbf{p}_k))$ ▷ answer space
return $\mathbf{H}$ with its validity mask, $\mathbf{g}$, and $\mathbf{C}$ or $\mathbf{P}$

Algorithm 2 One segment of the recurrent core. All steps share one set of weights.

Require: answer slots $\mathbf{S} \in \mathbb{R}^{K \times d}$, evidence bank $\mathbf{H}$ with its mask, condition $\mathbf{g}$
Require: outer loops T, refinement steps per loop n
for $t = 1$ to T **do**
  for $i = 1$ to n **do** ▷ *Refine*: relate, re-read, transform
    $\mathbf{S} \leftarrow \mathbf{S} + \text{SelfAttn}(\mathbf{S})$
    $\mathbf{S} \leftarrow \mathbf{S} + \text{CrossAttn}(\mathbf{S}, \mathbf{H})$ ▷ dynamic anchor: the full bank, at every step
    $\mathbf{S} \leftarrow \mathbf{S} + \text{FFN}\big(\text{FiLM}_{\mathbf{g}}(\text{LN}(\mathbf{S}))\big)$
  end for
  $\mathbf{S} \leftarrow \mathbf{S} + \text{SelfAttn}'(\mathbf{S})$ ▷ *Integrate*: consolidate without consulting $\mathbf{H}$
  $\mathbf{S} \leftarrow \mathbf{S} + \text{FFN}'(\mathbf{S})$
end for
return $\mathbf{S}$ ▷ $T(n{+}1)$ residual blocks applied with one set of parameters

question is fixed by the schedule rather than by the length of an answer, which is what Appendix F counts.

Where the ablations act. Each arm of §3.3 is one edit to the pseudocode, which is how they are implemented. *Text only* and *series only* drop the corresponding blocks from $\mathbf{H}$ in Algorithm 1, the latter also dropping $\mathbf{e}_t$, and neither touches $\mathbf{C}$ or $\mathbf{P}$. *No numeric-candidate encoding* always takes the text branch for candidates. *No boundary features* removes $\text{Proj}_{\text{b}}(\mathbf{B})$ from $\mathbf{H}$ and from the fragment rows, and *no segment ids* removes $\mathbf{E}_{\text{seg}}$. *Series read as text* replaces ϕ_{ts} by ϕ_{tx} applied to the serialized values. In Algorithm 2, the *pooled readout* skips the algorithm entirely and sets $\mathbf{S}$ to a masked mean of $\mathbf{H}$ broadcast over the K slots, running a single segment; the *static anchor* deletes the cross-attention line, so the slots see the evidence only through $\mathbf{g}$; the *access-only reader* deletes the two self-attention lines, so the slots re-read the bank but never meet; *no recursion* sets $T = n = 1$; and *truncated gradients* runs the first $T-1$ outer loops without gradient. In Algorithm 3, *no Sinkhorn coupling* replaces the iteration by the single row normalization described above, and

Algorithm 3 Structured readouts. No text is generated in either branch.

function SELECTION($\mathbf{S}_{\text{out}}, \mathbf{C}$) ▷ true/false and multiple choice
  $\mathbf{u} \leftarrow \text{AttnPool}(\mathbf{S}_{\text{out}})$
  $\ell_i \leftarrow \langle \mathbf{u}, \mathbf{c}_i \rangle / \sqrt{d}$ for each candidate, and $\ell_i \leftarrow -\infty$ for absent ones
  return $\text{softmax}(\ell)$ ▷ training loss: cross-entropy against $y^\star$
end function

function ORDERING($\mathbf{S}_{\text{out}}, \mathbf{P}$) ▷ the first n_p slots act as position queries
  $M_{jk} \leftarrow \langle \mathbf{s}_j, \mathbf{p}_k \rangle / \sqrt{d}$ for $j, k = 1, \ldots, n_p$
  $\log \Gamma \leftarrow M / \tau$
  for $i = 1$ to I **do** ▷ log-domain Sinkhorn
    $\log \Gamma \leftarrow \log \Gamma - \text{logsumexp}_k \log \Gamma$ ▷ rows
    $\log \Gamma \leftarrow \log \Gamma - \text{logsumexp}_j \log \Gamma$ ▷ columns
  end for
  return $\log \Gamma$ ▷ training loss: $-\frac{1}{n_p} \sum_j \log \Gamma_{j,\pi^\star(j)}$
end function

At inference, selection answers with $\arg\max_i \ell_i$ and ordering with $\hat{\pi} \leftarrow \text{HUNGARIAN}(-\log \Gamma)$, which is a permutation by construction.

Algorithm 4 Training with deep supervision

Require: cached evidence for the training split, segments N_{seg}, steps N_{steps}, auxiliary weight λ
initialize every submodule from a generator keyed by its own name ▷ see Appendix E
for step $= 1$ to N_{steps} **do**
  draw a (task, format) group with probability proportional to $N_{\text{group}}^{\alpha}$ ▷ one readout per batch
  draw a batch of instances from that group and apply the label-exact augmentations
  assemble $\mathbf{H}$, $\mathbf{g}$ and the answer space from the cache ▷ trained part of Algorithm 1
  $\mathbf{S} \leftarrow \mathbf{S}_0, \quad \mathcal{L} \leftarrow 0$
  for $s = 1$ to N_{seg} **do**
    $\mathbf{S} \leftarrow \text{CORE}(\mathbf{S}, \mathbf{H}, \mathbf{g})$ ▷ Algorithm 2
    $\hat{y}^{(s)} \leftarrow \text{SELECTION}(\mathbf{S}, \mathbf{C})$ or $\text{ORDERING}(\mathbf{S}, \mathbf{P})$ ▷ Algorithm 3
    $h^{(s)} \leftarrow \sigma\big(\mathbf{w}^\top \text{AttnPool}(\mathbf{S}) + b\big)$ ▷ auxiliary: is this answer already correct?
    $\mathcal{L} \leftarrow \mathcal{L} + \mathcal{L}_{\text{task}}(\hat{y}^{(s)}, y^\star) + \lambda \, \mathcal{L}_{\text{aux}}\big(h^{(s)}, \mathbb{I}[\hat{y}^{(s)} = y^\star]\big)$
    $\mathbf{S} \leftarrow \text{sg}[\mathbf{S}]$ ▷ state crosses segments, gradients do not
  end for
  update θ with AdamW on $\mathcal{L}/N_{\text{seg}}$, with warm-up, cosine decay and gradient clipping
  update the exponential moving average of θ
end for
return the checkpoint with the best validation score, evaluated with the averaged weights

argmax decoding replaces the Hungarian decoder by a per-position argmax, which is the only arm of the four that needs no retraining.

E IMPLEMENTATION DETAILS

Configuration. Table 4 lists the default configuration. The reported model is a single joint model trained on all tasks; per-task specialists appear only as an ablation.

Batching and sampling. Readouts are format-specific, so every batch is homogeneous in (task, format). Groups are drawn with probability proportional to $N_{\text{group}}^{\alpha}$ with $\alpha = 0.5$, which interpolates between frequency-proportional ($\alpha = 1$) and uniform ($\alpha = 0$) sampling and keeps the largest cells from dominating the update stream.

Table 4: Default configuration of GLEAN. The schedule applies $N_{\text{seg}}\,T(n{+}1) = 30$ residual blocks per forward pass.

Component	Value	Component	Value
working width d / heads	384 / 8	segments N_{seg}	2
answer slots K	12	outer loops T	3
FFN ratio	4.0	Refine steps n	4
ordering positions n_p	4	Sinkhorn τ / iterations I	0.5 / 8
question / option tokens	128 / 32	boundary descriptor	12 dims
optimizer	AdamW	batch size	64
lr (core / projections)	3×10^{-4} / 1×10^{-4}	steps	30k
weight decay / clipping	0.01 / 1.0	warm-up steps	2000
EMA decay	0.999	sampling α	0.5
auxiliary weight λ	0.1	trainable parameters	8,208,769

Optimization and model selection. We use AdamW with two parameter groups: a lower learning rate for the projections and embeddings that interface with the frozen features, and a higher one for the recurrent core and the readouts. The schedule is a linear warm-up followed by cosine decay, with gradient clipping. We keep an exponential moving average of the parameters and evaluate with the averaged weights. Checkpoints are selected on the official validation split with the metric used at test time, in particular per-position agreement for ordering, so that selection optimizes the reported quantity.

Cached perception. Both encoders are frozen, so their outputs and the boundary descriptors are computed once per instance and cached. Encoding runs in groups of shape-identical samples that are written back in the original order, which makes batched caching bitwise identical to per-sample caching.

Controlled ablations. Adding or removing a module changes the global random stream and therefore the initialization of unrelated parameters. Every submodule is instead initialized from a generator keyed by its own name, with the global stream restored afterwards, so that an ablation changes exactly the component it names. Ablations that change the cached features (numeric candidates, boundary features, segment identities, the time-series encoder) are trained from their own cache.

F Cost Accounting

Convention. All FLOP counts follow one convention (Kaplan et al., 2020): a forward pass costs $2N_{\text{non-emb}} + 2\,n_{\text{ctx}}\,d_{\text{attn}}$ per token and layer, plus $2\,d_{\text{model}}\,|V|$ at every position whose logits are computed. Causal prefill sums this over positions, decoding uses a key-value cache, layers with a sliding window use $\min(t, \text{window})$ as context, and encoders are bidirectional. Prompt and generated tokens are counted along the exact path each model uses for training and evaluation, that is, the benchmark prompt passed through the model's own template or processor, so the counts include the tokens a model spends on serialized numbers. For GLEAN, the two frozen encoders are counted analytically in the same way and the recurrent core is measured with a FLOP counter and extrapolated by its linear dependence on the length of the evidence bank. MOMENT processes $\lceil L/8 \rceil$ patch tokens per series, verified by instrumenting its encoder.

Per-model figures. Table 5 lists every system. Prompt length dominates the cost of the text-interface models, but it is not the whole story: ChatTS and PATRA use the shortest prompts of all language models and still spend $170\times$ and $87\times$ more computation per question than GLEAN, because every token passes through a multi-billion-parameter model.

Trainable parameters. Figure 6 compares trainable parameter counts and overall accuracy under the evaluated adaptation recipes. GLEAN achieves the highest accuracy among these six systems (77.33%) with the fewest trainable parameters (8.2M), compared with 10.1–16.8M for the five time-series LLMs. These counts depend on the adaptation recipe: GLEAN trains its projections, recurrent

Table 5: Cost per question on the TSAQA test split. Prompt tokens of GLEAN are text-encoder tokens; its series is encoded separately. ×: FLOPs relative to GLEAN.

Model	Params	Prompt tok.	GFLOPs/q	×	Overall
GLEAN	458.9M	69	28.2	1.0	77.33
TimeOmni-1-7B	7.62B	1,469	19,516	693	75.06
TimeMQA-Qwen2.5-7B	7.62B	1,447	19,252	683	72.74
ChatTS-8B	8.26B	340	4,798	170	72.13
ChatTime-1-7B	6.82B	548	7,188	255	56.83
PATRA-7B	7.98B	184	2,456	87	54.87
Qwen3-0.6B	596.0M	1,452	1,447	51	78.32
LLaMA3.2-1B	1.24B	1,145	2,291	81	73.48
Gemma3-1B	999.9M	1,456	2,080	74	69.70
Ministral-8B	8.02B	1,445	20,529	728	74.74
Qwen3-8B	8.19B	1,452	20,648	733	84.29
LLaMA3.1-8B	8.03B	1,145	16,246	576	85.26

core and readouts from scratch while keeping both encoders frozen; the baselines adapt pretrained language models with LoRA. The comparison measures the number of updated weights, excluding frozen parameters, rather than total model size or inference cost.

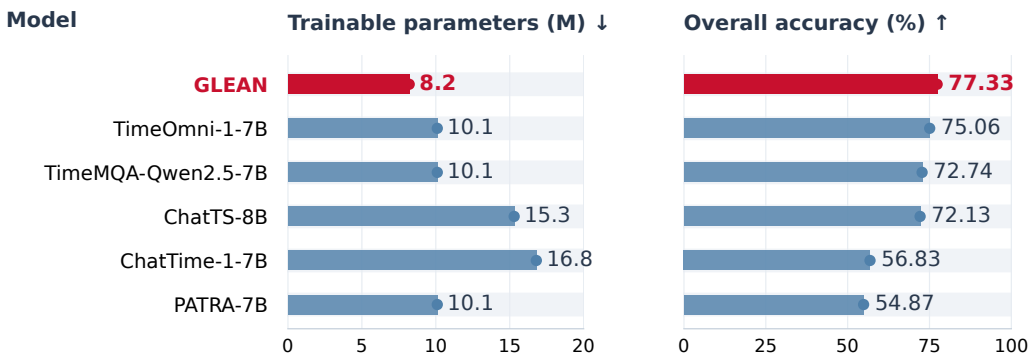


Figure 6: **Trainable parameters and accuracy on TSAQA.** Parameter counts and overall accuracy under the evaluated adaptation recipes. GLEAN is highlighted in red.

Where our cost goes. Of the 28.2 GFLOPs GLEAN spends per question, the frozen time-series encoder accounts for 11.9, the frozen text encoder for 13.7 and the recurrent core for 2.6. Reasoning is thus under a tenth of the cost, and the encoders are amortized whenever several questions concern the same series.

G Formal Statements and Proofs

This section states and proves the three results announced in §2.4.

Proposition 1 (Permutation equivariance of selection)**.** *Fix* $\mathbf{H}$ *and* $\mathbf{S}_{\text{out}}$*. For any permutation* ρ *of the* m *candidates, scoring* $\mathbf{C}^{\rho}$ *with rows* $\mathbf{c}^{\rho}_i = \mathbf{c}_{\rho(i)}$ *gives* $\ell^{\rho}_i = \ell_{\rho(i)}$*. Hence the set of maximizing indices transforms by* ρ^{-1}*, and relabeling the target as* $\rho^{-1}(y^\star)$ *leaves the cross-entropy unchanged.*

Proof. $\mathbf{u}$ is a function of $\mathbf{S}_{\text{out}}$ alone and does not depend on the order of the candidates. Each logit applies the same function to $(\mathbf{u}, \mathbf{c}_i)$ independently of the other candidates, so reindexing the candidates reindexes the logits. The softmax denominator is unchanged by reindexing, and the relabeled target has the same logit, $\ell^{\rho}_{\rho^{-1}(y^\star)} = \ell_{y^\star}$. Its softmax probability and cross-entropy are therefore unchanged. □

A unique maximizing index transforms in the same way; ties require an equivariant tie rule or a set-valued prediction. The statement is conditional on $\mathbf{H}$. When numeric candidates also enter the evidence bank, permuting the candidate matrix does not permute those tokens; the proposition is about how candidates are *scored*, not about reordering the evidence.

Proposition 2 (Factorized uncertainty cannot be confined to valid permutations)**.** *Let $n_p \geq 2$ and let $Q(\sigma) = \prod_{j=1}^{n_p} Q_j(\sigma(j))$ be a factorized distribution over assignments of fragments to positions. If every assignment in the support of Q is a permutation, then Q is a point mass. Equivalently, a factorized head that is uncertain between two orderings must assign positive probability to assignments that are not orderings.*

Proof. The support of a product distribution is the Cartesian product of the supports, $\mathrm{supp}(Q) = \prod_j \mathrm{supp}(Q_j)$. Suppose $\mathrm{supp}(Q)$ contains only permutations. If two positions $j \neq k$ shared a fragment $c \in \mathrm{supp}(Q_j) \cap \mathrm{supp}(Q_k)$, then the assignment sending both j and k to c would lie in $\mathrm{supp}(Q)$ and would not be injective, a contradiction. The supports are therefore pairwise disjoint, and n_p pairwise disjoint non-empty subsets of a set of n_p fragments must be singletons. Hence every Q_j is degenerate and Q is a point mass. □

Corollary 1 (What the Sinkhorn readout parameterizes)**.** *For finite scores and $\tau > 0$, as the number of iterations grows, Γ converges to a doubly stochastic matrix, which by the Birkhoff–von Neumann theorem is a convex combination of permutation matrices. The readout thus approaches the set of doubly stochastic matrices. This is a constraint on assignment marginals; it does not make independently sampled positions a valid joint distribution over permutations.*

Two caveats belong with this corollary. First, with a finite number of iterations the matrix is only approximately doubly stochastic. Our implementation ends with a column normalization, so column sums are one to numerical precision (we measure a deviation of 3×10^{-8}) while row sums deviate by about 7% at $I = 8$. Second, the training objective is a per-position negative log-likelihood on the rows of Γ, not the exact likelihood of a distribution over permutations; the corollary constrains the feasible set the readout can express, not the loss. Empirically, in §3.4.2, the coupled readout produces argmax decodings that are already permutations in 70.7% of the questions, against 1.1% without coupling.

Remark (information available to a reasoner). For fixed question, answer representations and task metadata, suppose a reasoner accesses the series only through $\mathbf{g} = \mathrm{AttnPool}(\mathbf{H})$. Then $\mathcal{X} \to \mathbf{H} \to \mathbf{g} \to \hat{y}$ is a Markov chain and the data-processing inequality bounds the information available after pooling. Information lost at this stage cannot be recovered by subsequent computation from $\mathbf{g}$ alone. Non-injective pooling does not, by itself, imply a strict loss of task-relevant information or lower accuracy. The static-anchor ablation (§3.3) therefore tests the practical value of token-level access rather than a predicted effect size.

H Per-Cell Ablation Results

Table 6 gives every ablation of §3.3 on all eleven cells. Two arms change active capacity as well as connectivity: the access-only reader activates a smaller core, and the static anchor leaves the disabled cross-attention untrained. The argmax-decoding arm is a separately trained model, so it mixes the decoding rule with run-to-run variation.

I Per-Cell Results across Frozen Encoders

Table 7 gives the fourteen encoder pairings of §3.4.1 on all cells. Two observations are worth stating explicitly. First, TR MC is the only cell that depends strongly on the time-series encoder: it reaches 86 to 88 with every encoder except MOMENT-1-large, which stays between 72.4 and 74.5 across all seven text encoders. Second, the spread across text encoders is concentrated on the language-heavy cells, and the encoder that is best on those cells is not the best overall.

Table 6: All ablations on the eleven cells of the TSAQA test split. Single run each.

	Perception		Relational					Language-heavy				
Variant	AD TF	CLS MC	DT TF	DT MC	TR TF	TR MC	TR PZ	Char TF	Char MC	Comp TF	Comp MC	Overall
Text only	50.88	56.27	48.20	31.69	50.34	30.67	33.26	84.45	64.70	77.83	56.82	51.48
Series only	92.22	92.22	71.32	73.95	76.70	69.58	57.63	62.47	65.33	57.22	56.43	71.93
Pooled readout, no core	88.77	92.88	66.06	78.75	66.93	66.02	42.96	83.62	66.37	75.01	54.93	70.79
Static anchor	90.33	91.18	69.65	71.11	75.88	69.28	56.94	88.05	67.73	79.25	55.68	74.50
Access-only reader	92.40	92.93	71.92	80.52	73.69	73.91	53.90	88.71	69.37	79.39	58.89	75.81
No recursion (T=n=1)	91.78	92.70	73.72	78.85	76.90	72.23	53.34	87.75	67.83	79.39	56.96	75.35
Truncated gradients	91.97	93.35	74.85	78.62	78.16	72.08	50.97	88.25	69.70	80.10	58.30	75.46
No numeric-candidate encoding	92.48	93.58	76.82	58.97	81.37	66.28	55.89	87.91	69.57	79.05	57.74	74.98
No boundary features, no segment ids	90.77	93.32	71.05	77.08	75.58	69.33	45.32	88.44	69.80	79.86	58.89	73.62
segment ids only	91.07	93.22	70.35	76.48	76.65	71.26	45.08	88.81	70.44	80.17	59.90	73.82
boundary features only	92.23	93.53	76.85	79.85	78.89	72.74	55.41	88.51	68.74	80.27	57.97	76.60
No boundary, segment ids, numeric enc.	90.80	93.62	70.82	57.84	76.46	59.16	44.79	88.31	69.80	79.52	57.91	71.64
Series read as text	85.17	89.88	67.46	61.51	74.37	69.63	45.62	85.15	67.27	77.56	57.91	70.32
No Sinkhorn coupling	85.73	85.57	63.29	70.25	65.76	66.99	38.66	87.11	67.37	78.71	57.91	68.47
Argmax decoding	92.28	93.45	75.48	79.39	79.86	73.40	54.23	87.61	69.77	78.91	58.00	76.24
GLEAN (Per-task specialists)	93.35	95.78	78.31	84.06	81.23	76.45	59.13	88.98	70.87	79.39	59.08	78.69
GLEAN (full)	93.40	93.12	78.45	79.49	79.18	74.52	57.87	88.21	69.64	78.78	58.36	77.33

Table 7: Frozen encoder pairings on the eleven cells. Single run each.

	Perception		Relational					Language-heavy				
Encoder	AD TF	CLS MC	DT TF	DT MC	TR TF	TR MC	TR PZ	Char TF	Char MC	Comp TF	Comp MC	Overall
Time-series encoder (text encoder: BGE-base)												
MOMENT-1-small	90.28	93.25	73.82	75.98	83.32	87.74	57.69	87.78	69.50	78.64	60.36	77.20
MOMENT-1-base	90.17	93.97	72.58	76.45	83.32	86.27	56.10	86.61	68.60	77.08	59.80	76.57
MOMENT-1-large (default)	93.40	93.12	78.45	79.49	79.18	74.52	57.87	88.21	69.64	78.78	58.36	77.33
Chronos-T5-small	90.32	93.70	73.72	77.45	79.77	87.33	58.07	87.08	68.87	76.98	59.84	77.00
Chronos-T5-base	90.78	93.70	74.05	81.09	81.47	87.95	57.87	87.58	69.54	78.06	61.21	77.68
Chronos-T5-large	90.33	93.73	78.21	78.19	84.00	87.49	56.28	87.31	68.64	78.47	58.99	77.30
Chronos-2-small	90.83	94.75	73.65	75.25	74.51	87.13	57.01	87.28	69.00	74.97	58.69	76.39
Chronos-2	90.53	94.25	75.18	77.65	84.00	86.98	56.83	87.65	69.70	78.57	61.11	77.49
Text encoder (time-series encoder: MOMENT-1-large)												
MiniLM	92.35	93.28	77.22	81.35	77.72	73.86	57.41	86.28	67.63	78.68	55.94	76.59
GTE-small	92.62	93.37	78.01	80.12	78.75	73.04	54.17	86.95	67.97	78.61	56.89	76.15
GTE-base	92.67	93.03	78.85	78.62	80.69	73.04	56.93	87.51	70.00	78.10	58.43	76.94
BGE-small	92.35	93.70	73.55	81.15	77.19	72.38	55.69	86.81	68.77	78.91	57.05	76.17
BGE-base (default)	93.40	93.12	78.45	79.49	79.18	74.52	57.87	88.21	69.64	78.78	58.36	77.33
BGE-large	92.42	93.47	69.59	80.09	75.15	73.60	55.89	88.48	70.14	79.73	59.57	76.24
MPNet	92.18	93.47	71.89	81.02	74.08	72.74	53.61	87.95	68.44	80.27	59.31	75.77

J Where the Separation Reaches Its Limit

Two families of cells remain in which language models lead; the main results and encoder analysis help explain these gaps (Sections 3.2 and 3.4.1).

Exact numeric continuation. The temporal-relationship TF and MC cells ask whether a candidate continues the context, which turns on exact values. Every model that reads the numbers as digit text (TimeOmni-1, TimeMQA and all instruction-tuned general LLMs) scores at least 88 on both cells, whereas GLEAN with its default encoder scores 79.18 and 74.52. Two observations qualify this gap. Boundary features and numeric-candidate encoding recover part of it (§3.3), and the choice of time-series encoder recovers more: with Chronos-T5-base, TR MC rises to 87.95 (Table 7). What remains, about 7 to 13 points to the digit-reading models on TR TF, is consistent with the normalization and patch pooling of frozen encoders, which can obscure exact magnitudes while retaining shape information.

Open-vocabulary language understanding. On the multiple-choice cells of characterization and comparison, every language model outperforms GLEAN. These are also the cells on which the question alone carries most of the answer (§3.3). Increasing the sentence-encoder size does not consistently improve language-heavy accuracy (§3.4.1). This suggests a limitation of the representation or its use by the reasoner, since each option is represented by one pooled sentence vector, but does not isolate the cause of the gap.

Table 8: **Complete schedule sensitivity results on TSAQA** (accuracy, %). The default configuration is shaded. The first block varies N_{seg}, the next two vary T and n with the default row shared, and the final row gives the minimal schedule.

N_{seg}	T	n	Blocks	AD TF	CLS MC	DT TF	DT MC	TR TF	TR MC	TR PZ	Char TF	Char MC	Comp TF	Comp MC	Overall
1	3	4	15	92.30	93.07	76.18	80.42	78.99	72.69	58.41	88.25	68.70	80.34	58.53	77.13
2	3	4	30	93.40	93.12	78.45	79.49	79.18	74.52	57.87	88.21	69.64	78.78	58.36	77.33
3	3	4	45	92.37	92.55	76.22	79.85	77.43	73.35	55.63	88.41	70.24	79.52	58.85	76.55
4	3	4	60	88.73	89.25	65.39	75.58	74.61	69.89	51.04	87.41	67.87	78.51	56.60	72.83
5	3	4	75	88.12	87.78	60.83	72.18	69.50	67.24	47.94	86.85	66.93	78.03	56.79	70.88
6	3	4	90	91.00	90.05	69.59	75.52	75.44	71.16	51.96	87.88	68.17	79.39	57.38	74.01
2	1	4	10	92.32	93.48	72.75	80.19	76.31	72.33	54.48	88.71	69.20	79.69	58.72	76.08
2	2	4	20	92.48	93.15	77.75	79.69	79.43	72.69	53.20	88.25	69.17	79.35	58.79	76.25
2	4	4	40	91.72	92.73	70.62	79.12	75.44	73.09	55.15	88.61	70.74	80.44	58.89	75.94
2	5	4	50	92.35	93.15	73.72	79.12	79.13	72.94	55.42	87.81	69.60	79.05	57.48	76.20
2	3	1	12	92.15	92.87	77.02	79.25	79.28	73.40	56.10	87.68	68.97	79.42	59.02	76.62
2	3	2	18	91.93	93.48	71.42	77.79	77.04	71.72	51.85	88.68	69.47	79.12	58.49	75.22
2	3	3	24	91.77	93.17	76.48	79.95	80.40	73.55	56.91	88.68	68.87	80.17	57.97	76.87
2	3	5	36	91.90	92.73	70.82	78.09	77.48	71.31	54.05	88.28	69.67	79.29	58.92	75.54
1	1	1	2	92.08	92.82	76.28	79.89	77.14	73.35	52.92	87.78	68.17	78.71	56.40	75.59

These gaps identify limits of the current combination of frozen representations and compact reasoning. The experiments do not establish whether richer representations, a different reasoner, or both would close them.

K Complete Recurrence Schedule Results

Table 8 reports all eleven task–format cells for the schedules in Figure 5. Each sweep changes one parameter while fixing the other two at $(N_{\text{seg}}, T, n) = (2, 3, 4)$. The shared default is listed once. We also include the single-segment, single-loop configuration discussed in Section 3.4.3. The block count is $N_{\text{seg}}T(n+1)$; increasing it does not change the number of trainable parameters. Overall accuracy is sample-weighted, and TR PZ follows the benchmark scoring described in Appendix B.

L Additional Visualization

Attention lift on all cells. Table 9 gives the values behind Figure 4a for all eleven cells. Candidate tokens exist only in the cells whose candidates are series. Question tokens are read well above uniform on the language-heavy cells and near uniform on the perception cells, series tokens stay below uniform per token everywhere because they are the most numerous type, and the single boundary token per series attracts high attention in every cell, including cells on which removing the boundary features does not lower accuracy.

Table 9: Attention lift per cell: share of the slots' cross-attention divided by share of tokens, averaged over all Refine steps. A value of one corresponds to uniform reading.

Cell	Series	Question	Candidate	Boundary
AD TF	0.91	2.80	–	46.04
CLS MC	0.98	1.30	–	15.66
DT TF	0.90	1.47	–	19.33
DT MC	0.84	3.85	10.09	13.65
TR TF	0.89	1.79	–	11.39
TR MC	0.84	3.63	7.04	41.08
TR PZ	0.79	3.88	4.71	8.21
Char TF	0.45	15.88	–	15.91
Char MC	0.56	10.50	–	45.88
Comp TF	0.51	13.83	–	6.59
Comp MC	0.57	11.39	–	4.65

Does attention land where the signal is informative? TSAQA does not annotate the location of anomalies or of the segments a question refers to, so we use a proxy for how informative a patch is: its *deviation*, the largest absolute difference between a value inside the patch and the median of its channel. Because patches are unevenly distributed over deviation levels, we report a *lift*: the attention a group of patches receives divided by the share of patches it contains, so that uniform reading gives one regardless of how many patches a series has.

Attention is allocated by salience, and it is allocated consistently (Figure 7). The 10% most deviant patches receive 1.3 to 2.1 times the attention they would receive under uniform reading, in every one of the ten cells. Resolving this by decile shows a step rather than a gradient: the flattest half of the patches is read *below* uniform, often at half the uniform rate, and the lift rises above one from the sixth decile on. The ordering across cells is interpretable. It is largest on the two characterization cells (about 2), where the questions ask about volatility and extreme behavior, and smallest on temporal-relationship true/false (about 1.3), where the answer depends on whether two endpoints match rather than on where the series deviates most. Two limits of this analysis are worth stating: deviation is a proxy for informativeness and not the region that answers the question, and the metric says where attention goes, not which evidence the answer depends on, which is what the ablations of §3.3 measure.

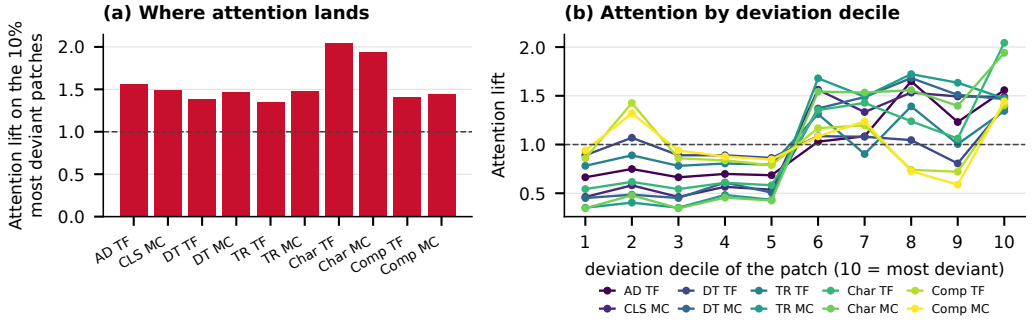


Figure 7: Attention against patch deviation, per cell, as attention per patch relative to uniform reading. (a) Lift of the 10% most deviant patches. (b) Lift per decile of deviation. Dashed: uniform reading. Reordering questions are analyzed separately and are excluded here.

Case studies. Figure 8 maps the attention of the answer slots back onto the time axis for one question per cell: each patch token covers a fixed window of the series, and the shading gives the attention of that window relative to uniform reading. The example of a cell is chosen by a rule fixed in advance: among the questions the model answers correctly, the one whose attention over the series is most concentrated, measured by normalized entropy. The four examples show attention on the parts of the signal the question is about: the step change on the anomaly-detection question, the falling and rising segments that a wavelet transform acts on, the peaks of the gait recording, and one high-variance burst on the characterization question. One further pattern is visible and should be read with care: the first and the last patch of a series often receive high attention regardless of their content, the same positional effect we observe for the single boundary token in §3.4.2.

Two ordering questions. Figure 9 concatenates the fragments of two reordering questions in the order the model predicts, so that the joins between fragments become visible. In the solved example the four fragments compose into a continuous signal whose joins are indistinguishable from the rest of the curve. In the failed example, two adjacent fragments are exchanged, and the two joins they create break the shape of the underlying cycle, which the ground-truth order in panel (c) restores. Errors of this kind, a swap of two adjacent fragments, account for one in four of the model's remaining ordering errors, and they are consistent with a readout that scores compatibility between endpoints: two fragments whose ends are similar are easy to exchange.

Error structure of ordering. Figure 10 classifies the outcome of Hungarian decoding on all $5{,}978$ reordering questions. With the Sinkhorn readout, 33.5% of the questions are solved exactly and a

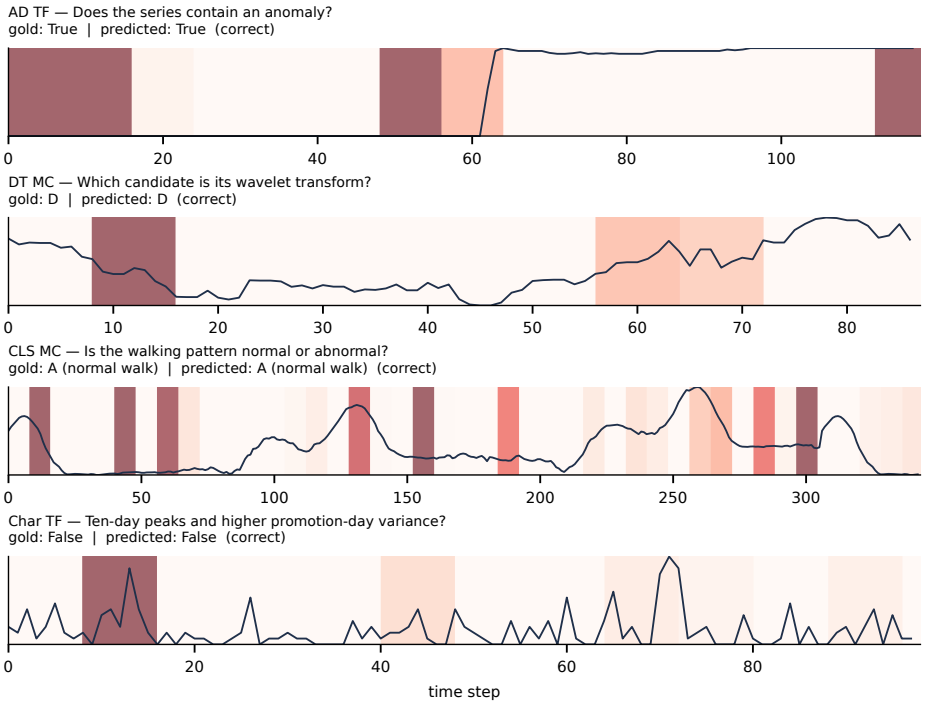


Figure 8: Attention over the time axis for one correctly answered question per cell (shading: attention relative to uniform reading). Titles paraphrase the question and give the ground truth and the prediction.

further 16.6% differ from the ground truth by a swap of two adjacent fragments, that is, one in four of the remaining errors is local. Without the coupling, only 8.0% are exact and most errors are larger misorderings.

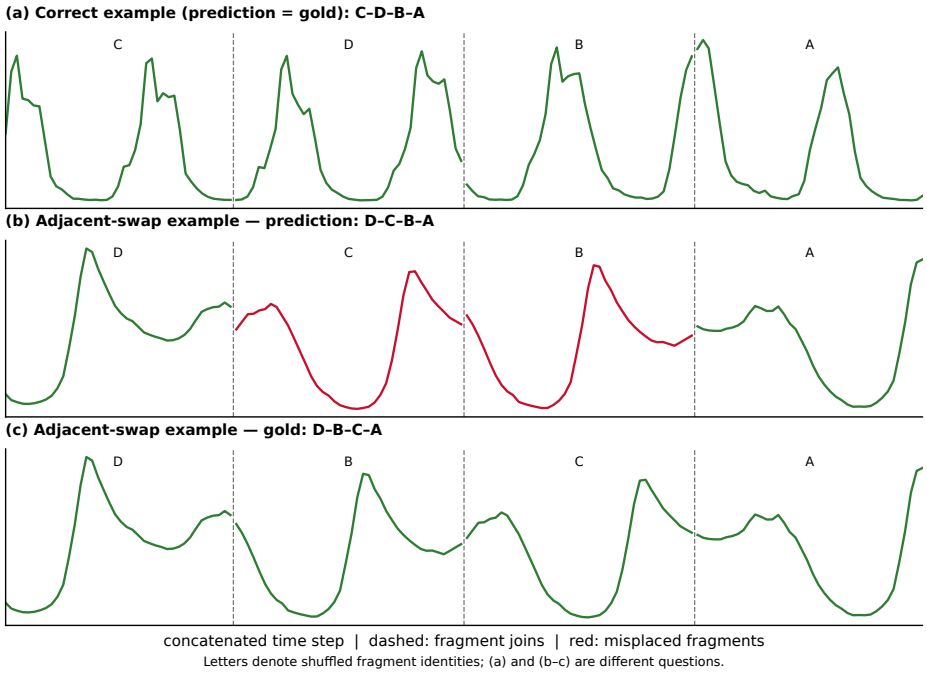


Figure 9: Two reordering questions with their fragments concatenated in the predicted order. (a) A solved question. (b) A question in which two adjacent fragments are exchanged; red marks the misplaced fragments. (c) The same question in the ground-truth order. Dashed lines mark the joins between fragments.

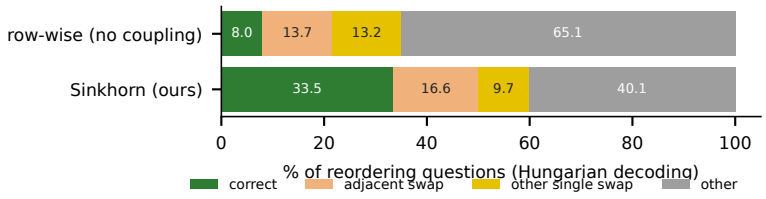


Figure 10: Outcome of Hungarian decoding on the reordering questions, for the Sinkhorn readout and a model trained with row-wise normalization only.