LEXIS: A SYMBOLIC VOCABULARY FOR TIME SERIES UNDERSTANDING IN PRETRAINED LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Pretrained large language models (LLMs) exhibit strong language understanding and reasoning capabilities, but applying these capabilities to time series understanding requires an interface that exposes temporal structure in a form legible to the model. Existing interfaces represent a series either as decimal strings or as embeddings produced by a series encoder. Decimal strings preserve the precise values but obscure their temporal relationships, while encoder embeddings are compact and information rich but lack semantics known to an LLM. We propose **LEXIS**, an interface that expresses a time series as language by extending the vocabulary of a pretrained LLM. A small set of new symbols describes the structure of a series, each symbol denoting one known component of a curve, and their embeddings are derived from the geometric relationships among the symbols. Across three time series understanding benchmarks, LEXIS-4B improves a Qwen3-4B backbone by 31.09%, 13.40%, and 4.45% and surpasses evaluated general-purpose LLMs and time series language models on most tasks. Controlled comparisons show that LEXIS outperforms the decimal and encoder embedding representations under the same training recipe. Ablations show that the model effectively derives temporal properties from the LEXIS representation and benefits consistently from the symbol geometry. These results establish LEXIS as an effective and principled interface between time series and pretrained LLMs.

1 INTRODUCTION

Multimodal large language models (MLLMs) extend the strong language understanding and reasoning capabilities of pretrained LLMs to images, audio, and other nontextual inputs. Through suitable interfaces, vision and audio language models can perform complex multimodal reasoning tasks (Liu et al., 2023b; Alayrac et al., 2022; Bai et al., 2025; Tang et al., 2024). Time series have not seen comparable progress. Understanding a series requires identifying temporal properties such as trends, periodic patterns, multivariate relationships, anomalous events, and responses to external events. It may also require relating these properties to text describing the measurement, the observation condition, and co-occurring events (Liu et al., 2024; Williams et al., 2025). This has motivated models that jointly process time series and text (Jin et al., 2024b; Xie et al., 2025; Kong et al., 2025; Langer et al., 2026; Guan et al., 2026a; Wang et al., 2026; Chang et al., 2026).

Existing models provide limited evidence of genuine alignment between time series and language. General LLMs perform only modestly above chance on questions about temporal structure (Merrill et al., 2024; Yu et al., 2026a). Time series language models (TSLMs) often retain much of their accuracy after the series is removed from the prompt (Merrill et al., 2024; Kong et al., 2025), a finding we reproduce in Appendix B.5. These results show that current models often fail to ground their answers in the input series and motivate a closer examination of how existing representations present temporal information to an LLM.

Two representations dominate current practice. The first serializes numerical values as decimal strings (Gruver et al., 2023; Kong et al., 2025). This preserves numerical precision, but standard subword tokenizers can expand a moderately long series into thousands of tokens (Spathis & Kawsar, 2024; Götz et al., 2026) with the temporal structure kept implicit. Recovering trends, periodicity,

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.

and abrupt changes requires arithmetic across tokenized numbers, a task at which LLMs remain unreliable (Liu et al., 2025a). In our controlled experiments, a model fine-tuned with only the mean and standard deviation of each short segment of the series performs within approximately 2% of a model fine-tuned with complete decimal series (section 4.2). The additional numerical precision therefore contributes little to temporal understanding.

The second representation projects the outputs of a series encoder into the token stream (Jin et al., 2024a; Xie et al., 2025; Wang et al., 2025b; Langer et al., 2026). These representations, often called soft tokens, are compact and information rich, but their encoded features are not grounded in the token space learned during language pretraining. Existing analyses find little evidence that such grounding succeeds (Tan et al., 2024; Zheng et al., 2025; Zhang et al., 2026b). Our matched evaluation reaches a similar conclusion: a model trained with projected representations that retain substantially more information about each patch performs no better than one trained with mean and standard deviation alone on two benchmarks (section 4.2). Decimal strings leave temporal structure implicit in the values, while projected representations place it in a space unfamiliar to the model. Analytic tools and manually specified features can expose selected properties, but they address prescribed questions and do not provide a general representation of the series (Kim et al., 2026; Liu et al., 2025b; Chang et al., 2026).

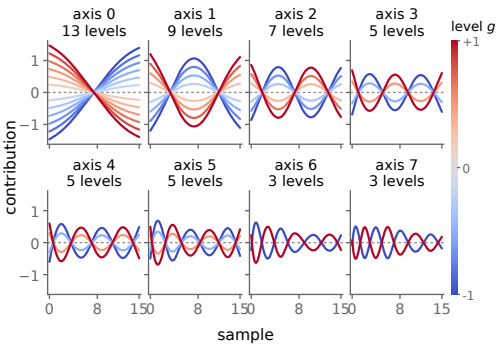


Figure 1: The curve component contributed by each shape symbol, one panel per axis and one curve per level (section 3.3).

We propose **LEXIS**, a representation that expresses time series as language through an extended vocabulary. LEXIS divides a series into short patches and represents each patch through two complementary channels, which we call rails: an envelope rail that records its level and scale as numerals, and a shape rail built from a compact alphabet of new symbols added to the LLM vocabulary (fig. 1). Each shape symbol denotes one component of a curve under a model independent decoder, and the symbols compose to describe a large space of curves. As with ordinary tokens, their meanings are learned from the contexts in which they occur. Bidirectional supervision over paired series and text grounds the symbols, while their repeated occurrence across the corpus provides dense supervision. The resulting token sequence exposes temporal structure explicitly and allows each component to be audited independently (fig. 2). To facilitate alignment with the pretrained LLM, LEXIS initializes the symbol embeddings according to the axis and level structure of the alphabet and preserves this geometry during training (fig. 3).

Contributions.

- **A series vocabulary for pretrained LLMs.** We introduce an extended vocabulary that represents series through a compact set of compositional symbols, each with a known referent.
- **Structured symbol alignment.** We develop an initialization and training scheme that utilizes the geometry of the shape symbols to facilitate symbol alignment.
- **Controlled attribution of understanding.** Through matched comparisons and component-wise interventions, we isolate the effect of the representation and verify that the model uses each component as intended.

2 RELATED WORK

Multimodal language models reach a new modality through an interface that brings its representation into the language space. Vision and audio models employ continuous projection or cross attention (Liu et al., 2023b; Alayrac et al., 2022; Bai et al., 2025; Tang et al., 2024), and most time series language models inherit this design (Jin et al., 2024a; Xie et al., 2025; Langer et al., 2026; Wang

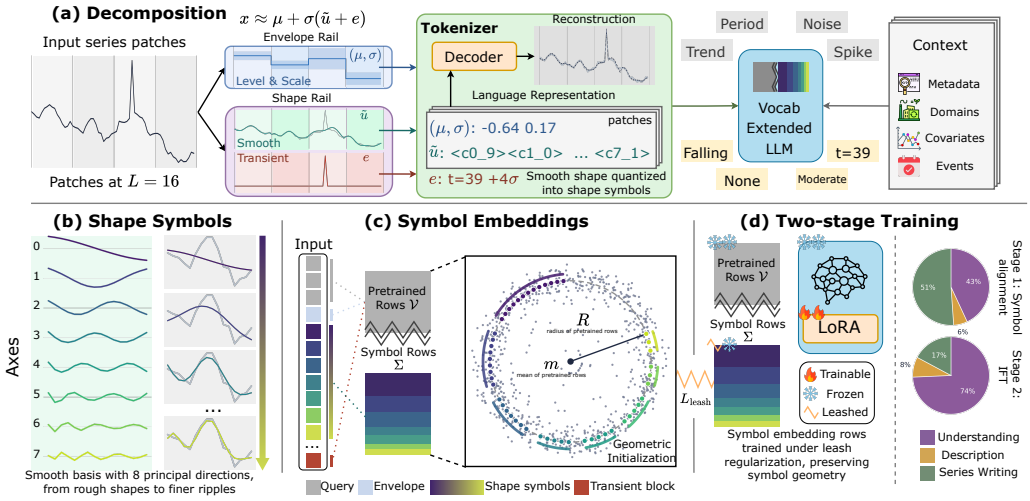


Figure 2: LEXIS overview. Each series is represented through complementary envelope and shape rails. A compact vocabulary of compositional symbols describes patch shape, while geometric initialization and training align the new symbols with the pretrained LLM.

et al., 2026), with decimal serialization and visual rendering as alternatives (Gruver et al., 2023; Kong et al., 2025; Liu et al., 2025a). These interfaces expose a central distinction between preserving information and making it legible: decimal strings retain the values and encoder embeddings retain rich signal, yet controlled analyses find weak numerical modeling, limited alignment with language, and predictions no better with the series than without it (Tan et al., 2024; Liu et al., 2025a; Zheng et al., 2025; Merrill et al., 2024). Discrete representations offer an alternative direction, but existing tokens generally denote scalar values, complete segments, or learned latent codes (Ansari et al., 2024; Talukder et al., 2024; Chen et al., 2026b). Recent work shows that discrete tokens can improve understanding, that their embedding order matters, and that local shape vocabularies can support generation (Quinlan et al., 2026; Li et al., 2026; Wei et al., 2026); research on vocabulary extension further shows that initialization affects whether new symbols acquire usable meanings (Chen et al., 2026a; Joshi et al., 2026). These studies leave open how to express temporal structure through symbols that denote known, composable curve components and install them reliably in a pretrained LLM. LEXIS addresses this gap by assigning each symbol one additive component of a curve and constructing its embedding from the same symbol geometry, yielding a representation that is compact, decodable, and auditable by component. Appendix C provides a full discussion.

3 METHOD

Pretrained LLMs operate on discrete token sequences, motivating three design requirements. First, the representation should be discrete so that the model can read and write it as ordinary language tokens. Second, each token should denote a known component of the series, allowing its interpretation to be learned from paired data and subsequently verified against that component. Third, the vocabulary should be compact and compositional, providing dense supervision for every symbol. LEXIS satisfies these requirements by encoding a series as a compact, decodable sequence of numerals and typed symbols that exposes its temporal structure.

The method comprises four components. First, the representation separates each patch into an envelope rail for level and scale and a shape rail for within patch structure (section 3.2). Second, a compositional alphabet represents each shape using symbols that each denote one curve component under a deterministic decoder (section 3.3). Third, geometric initialization places the alphabet in the embedding space of the LLM according to its axis and level structure (section 3.4). Finally, a two stage training pipeline aligns the symbols while preserving this geometry and adapts the model to time series understanding tasks (section 3.5).

3.1 Problem Formulation

Let f_θ be a pretrained LLM with vocabulary $\mathcal{V}$ and embedding matrix $E \in \mathbb{R}^{|\mathcal{V}| \times d}$. Given a multivariate series $x \in \mathbb{R}^{N \times V}$ with N observations and V variables, and textual context C containing an instruction, question, and any description of the series, the model outputs a textual answer y. We seek a mapping $\mathcal{S}$ from x to a token sequence $\mathcal{S}(x)$, which we call the surface of x, over an extended vocabulary $\mathcal{V}' = \mathcal{V} \cup \Sigma$, where Σ contains the new shape symbols and $E_\Sigma \in \mathbb{R}^{|\Sigma| \times d}$ contains their embeddings, tied to the output head. The model receives $(C, \mathcal{S}(x))$ and generates y.

3.2 Two-rail Decomposition

LEXIS represents a series with two complementary rails (fig. 2a). For clarity, we formulate the construction for a univariate series, as the same construction applies to each variable independently in the multivariate case. Each variable is first standardized using its mean and standard deviation over the full input span (the series as given in the prompt) and then divided into patches of length $L = 16$. For patch p, let μ_p and σ_p denote its mean and standard deviation, and define the normalized patch $u_p = \frac{x_p - \mu_p}{\sigma_p}$. Standardizing each patch removes its level and scale before the shape is symbolized, and the same shape then receives the same symbols at any level or scale. We further decompose u_p into a smooth shape $\tilde{u}_p$ and sparse transients e_p, giving

$$x_p = \mu_p + \sigma_p\big(\tilde{u}_p + e_p\big). \tag{1}$$

Envelope rail (μ_p, σ_p). The envelope rail is expressed as plain numerals. Across patches, the means describe changes in level, and the standard deviations describe changes in local scale. The envelope therefore provides a coarse trace of the series using tokens already present in the pretrained vocabulary (Levy & Geva, 2025). These statistics express level and scale directly. The shape within each patch is what a decimal serialization leaves implicit, and LEXIS introduces new symbols only for this residual structure.

Shape rail $(\tilde{u}_p, e_p)$. The normalized patch contains the structure not captured by its envelope. Its smooth component $\tilde{u}_p$ is represented by K typed symbols, one for each axis of a fixed basis (section 3.3) in axis order, giving a constant cost of K tokens per patch. Sparse excursions such as spikes and dips cannot be reliably captured by a smooth basis and are therefore isolated into e_p. Each transient is serialized by its within patch position, its extent, and its signed magnitude, from which the fixed decoder reconstructs e_p. Appendix A.2 gives the detection procedure and its justification.

Serialization and decoding. The envelope, shape symbols, and transient block are serialized under a fixed grammar given in Appendix A.1. Decoding reads the shape symbols into the smooth curve, or skeleton, $\hat{u}_p$ of equation 3, restores the recorded transients, applies equation 1 with $\hat{u}_p$ in place of $\tilde{u}_p$ in the standardized units, and then reverses the standardization of the full span (decoded examples in figs. 9 and 10). The resulting representation uses approximately 25% as many tokens as decimal serialization at three significant figures.

3.3 Shape Alphabet

The shape alphabet applies finite scalar quantization (Mentzer et al., 2024) to the coordinates of a patch in a fixed orthonormal basis: each coordinate is rounded independently, and every axis and level pair is one symbol. A fixed basis with per axis rounding, rather than a learned codebook, keeps every symbol decodable by a linear map and orders the levels of each axis.

Basis and quantization. Let $B \in \mathbb{R}^{L \times K}$ contain the leading K principal directions of the smooth, normalized patches. Its columns are orthonormal and ordered by the variance they explain (fig. 1; Appendix A.3). Because each retained axis adds one token to every patch, K controls the tradeoff between reconstruction fidelity and sequence length. We set $K = 8$: the first 8 axes retain 77.5% of the patch energy (squared norm), while each subsequent axis adds 3.1 to 3.4% at the cost of another token (fig. 7). Axis k has L_k quantization levels, with $(L_1, \ldots, L_8) = (13, 9, 7, 5, 5, 5, 3, 3)$, giving the leading axes finer resolution. A gain s_k determines the range covered by each axis. Let $q_k = \lfloor L_k/2 \rfloor$ denote the neutral level. For the coefficients $a = B^\top \tilde{u}$ of a patch, each a_k is quantized to a level index i_k and its normalized grid value g_k:

$$i_k = \mathrm{clip}\big(\,\mathrm{round}(q_k\, a_k / s_k + q_k),\, 0,\, L_k - 1\big), \qquad g_k = \frac{i_k - q_k}{q_k}. \tag{2}$$

The corresponding shape skeleton is decoded linearly:

$$\hat{u} \;=\; B\,(g \odot s) \;=\; \sum_{k} B_{:,k}\, s_k\, g_k. \tag{3}$$

Because B is orthonormal, the squared reconstruction error separates into projection error outside the retained subspace and independent quantization errors along the retained axes. We fit both the basis and the gains on the training corpus. The basis is obtained from the principal components of the smooth, normalized patches, and each gain s_k is selected to minimize the empirical rounding error along its axis (Appendix A.3). Rounding removes a further 3.8% of the energy.

Compositional vocabulary. A conventional factored quantizer combines the axis levels into a single index and assigns one token to each of the $\prod_k L_k$ combinations (Mentzer et al., 2024; Zhang et al., 2026c). LEXIS retains the factors in the token sequence: each axis and level pair receives one token, and every patch is represented by exactly K symbols in axis order. The resulting vocabulary uses $\sum_k L_k = 50$ symbols to express $\prod_k L_k = 921{,}375$ distinct shapes. A flat codebook for the same space would require an impractically large vocabulary extension. Because each symbol recurs across all shapes containing its axis and level pair, structurally similar patches share supervision. The representation also preserves additivity and order: each symbol contributes a known curve component under equation 3, and adjacent levels represent adjacent coefficients. These properties enable controlled interventions and define the geometry used for symbol installation.

3.4 INSTALLING SHAPE SYMBOLS IN THE LLM

The shape symbols add $\sum_k L_k = 50$ rows to the embedding matrix of the LLM (fig. 2c). Their installation requires two choices: where these rows begin and how freely they move during training (section 3.5). Common schemes initialize new rows near the vocabulary mean with small random perturbations, which encode no relationships among the symbols and leave their structure to be learned entirely from paired data (Chen et al., 2026a; Tao et al., 2026; Zhang et al., 2026b; Joshi et al., 2026). LEXIS initializes the rows from the axis and level structure of the alphabet and regularizes this geometry during alignment.

Geometric initialization. For each symbol n, we lift its normalized level coordinate g_n into a feature vector containing a one hot code for its family (its axis, for a shape symbol; Appendix A.4 lists the other families) and sinusoidal features of its level:

$$\phi_n = \big[\mathbf{1}_{\mathrm{family}(n)}; \alpha_1 \sin(\omega_1 g_n), \alpha_1 \cos(\omega_1 g_n), \ldots, \alpha_T \sin(\omega_T g_n), \alpha_T \cos(\omega_T g_n)\big]. \tag{4}$$

The frequencies decrease geometrically below π, and $\alpha_t = \omega_t^{-1/2}$ gives greater weight to lower frequencies. All lifted vectors have the same norm. Appendix A.4 gives the complete construction. Let m be the mean of the pretrained token embeddings, let $U \in \mathbb{R}^{d\times F}$ contain the leading F principal directions of the centered embeddings, where F is the length of ϕ_n, and let R be the median distance of the pretrained embeddings from m. We initialize the embedding of symbol n as

$$E_n = m + \kappa U \phi_n, \qquad \kappa = \frac{R}{\mathrm{median}_n \,\|\phi_n\|}. \tag{5}$$

The family code separates the basis axes, the sinusoidal features preserve the order of levels within each axis, and the scale κ places the new embeddings on the typical radius of the pretrained vocabulary. Rows at the vocabulary mean lie inside the convex hull of the pretrained rows, where the tied output head cannot select them under greedy decoding; the placement makes every new row a vertex of that hull at initialization (Appendix A.4).

3.5 TRAINING

We fine-tune LEXIS-4B on the Qwen3-4B backbone model with a two-stage pipeline (fig. 2d).

Symbol alignment. The first stage trains the new symbol embeddings E_Σ together with low rank adapters on the LLM backbone (Hu et al., 2022), with the original token embeddings frozen, under the objective

$$\mathcal{L} = \mathcal{L}_{\mathrm{CE}} + \lambda\, \mathcal{L}_{\mathrm{leash}}, \tag{6}$$

Table 1: Class-wise time series understanding accuracy on TSAQA, TimeSeriesExam, TemporalBench. We report four trained Qwen3-4B variants that differ only in series representation, as raw decimals, as soft tokens, as the envelope rail alone, or as the full surface. Δ denotes performance difference between LEXIS-4B and the base model. TR: temporal relationship, CH: characterization, CP: comparison, DT: data transformation, AD: anomaly detection, CL: classification, PR: pattern recognition, SA: similarity analysis, NU: noise understanding, CA: causality analysis. **Bold** denotes the best in each column.

	TSAQA							TimeSeriesExam						TemporalBench			
	TR	CH	CP	DT	AD	CL	All	PR	SA	AD	NU	CA	All	T1	T3	T4	All
Chance	26.39	37.50	37.50	37.50	50.00	44.58	36.25	38.40	46.39	35.49	46.43	37.96	40.13	31.58	30.35	25.00	29.36
Large Language Models																	
GPT-4o	52.67	79.50	74.00	53.50	52.00	45.00	60.82	61.33	59.17	50.00	**65.48**	51.39	58.85	**46.08**	38.36	37.52	40.64
GPT-4o-mini	35.67	76.50	62.00	45.50	51.00	49.00	52.27	48.34	61.67	39.81	63.10	44.44	50.54	21.29	35.30	**40.66**	32.15
Gemma-4-E4B-it	37.67	71.50	61.00	45.50	49.00	46.00	51.27	48.34	55.83	35.19	64.29	47.22	49.33	28.71	34.64	36.65	33.24
Qwen2.5-7B	36.00	69.50	57.00	41.50	48.00	47.00	49.00	47.51	50.00	36.11	59.52	45.83	47.45	27.59	**42.62**	38.92	36.78
Qwen3-1.7B	29.33	39.50	40.00	33.50	49.00	45.00	37.09	44.75	41.67	38.89	51.19	30.56	42.76	24.37	41.97	39.09	35.51
Mistral-7B	30.33	67.50	55.50	32.50	55.00	44.00	45.55	40.33	41.67	34.26	44.05	44.44	40.48	24.37	34.54	24.43	28.61
Llama-3.1-8B	29.00	67.50	57.00	42.50	48.00	40.00	46.27	37.57	47.50	38.89	47.62	40.28	40.75	17.09	31.80	31.06	26.84
Time Series Language Models																	
ChatTS-14B	27.33	37.50	48.00	38.50	30.00	45.00	36.82	58.56	60.83	39.81	58.33	48.61	55.23	45.10	40.87	36.13	**41.01**
ChatTS-8B	20.00	54.50	34.00	35.00	45.00	43.00	35.91	49.45	54.17	39.81	55.95	37.50	48.39	21.99	31.69	32.98	28.88
TimeOmni-1-7B	38.67	62.00	41.00	43.00	51.00	48.00	46.09	52.76	41.67	44.44	55.95	41.67	49.06	36.41	37.05	29.49	34.88
Qwen3-4B base	27.67	70.50	60.00	37.50	47.00	40.00	46.00	46.41	55.00	39.81	52.38	44.44	47.32	21.57	38.69	38.05	33.61
series trained as																	
Decimal	**63.00**	85.50	77.00	70.50	80.00	76.00	73.73	58.29	**65.00**	**55.56**	63.10	45.83	58.31	39.08	38.58	24.78	35.15
Soft token	53.00	85.00	78.00	68.50	83.00	84.00	71.73	55.52	59.17	54.63	57.14	52.78	55.90	42.02	39.78	28.80	37.65
Envelope-only	54.00	84.50	78.00	73.00	76.00	**85.00**	72.18	57.50	55.00	45.42	60.35	48.61	57.24	26.33	36.17	36.30	33.02
LEXIS-4B	61.33	**86.00**	**80.50**	**81.00**	**84.00**	**85.00**	**77.09**	**63.81**	60.00	51.85	**65.48**	**54.17**	**60.72**	35.01	39.34	39.79	38.06
Δ	+33.66	+15.50	+20.50	+43.50	+37.00	+45.00	+31.09	+17.40	+5.00	+12.04	+13.10	+9.73	+13.40	+13.44	+0.65	+1.74	+4.45

where $\mathcal{L}_{\text{CE}}$ is the next token cross entropy over the target tokens of each example and $\mathcal{L}_{\text{leash}}$ holds the symbol embeddings near the geometry of equation 5:

$$\mathcal{L}_{\text{leash}} = \frac{1}{|\Sigma|F} \sum_{n \in \Sigma} \left\| \kappa^{-1} U^{\top} (E_n - m) - \phi_n \right\|^2 + \frac{\beta}{|\Sigma|} \sum_{n \in \Sigma} \left(\|E_n - m\| - R \right)^2. \tag{7}$$

The first term preserves the coordinates assigned to each symbol in the placement subspace. The second maintains the typical embedding radius, with $\beta = 1/(\kappa^2 F)$ placing the terms on comparable scales. The remaining $d - F$ directions stay free for task specific adaptation within the radius constraint; Appendix A.4 gives λ.

The alignment data combine synthetic series, whose labels are computed directly from their generation parameters, with real series and paired text from GIFT-Eval (Aksu et al., 2024) and CAF-7M (Zheng et al., 2026). Supervision operates in both directions. Approximately half of the examples ask the model to answer questions about or describe a series, while the remainder ask it to generate a series representation (series writing), and the target tokens of $\mathcal{L}_{\text{CE}}$ are text in the first case and the surface $\mathcal{S}(x)$ in the second. This bidirectional objective connects each symbol both to the curve it decodes and to language describing that curve.

Instruction fine tuning. The second stage freezes E_Σ, keeping the symbol embeddings established in the first stage, and trains the low rank adapters under $\mathcal{L}_{\text{CE}}$ alone for instruction following and time series understanding. The training corpus combines synthetic questions about controlled series properties with examples from the training sets of TSAQA (Jing et al., 2026) and TimeMQA (Kong et al., 2025), and retains a share of series writing examples. Appendix B.1 reports the data sources, mixture proportions, and curation procedures.

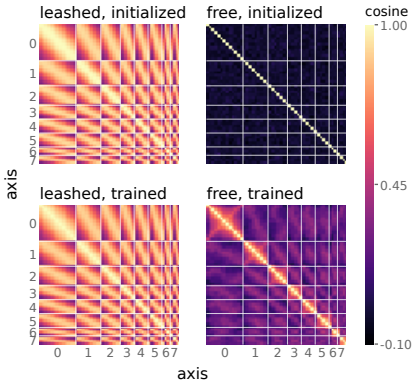


Figure 3: Cosine similarity among shape symbol embeddings under geometric installation with leash regularization (left) and mean initialization with unconstrained training (right), before training (top) and after training (bottom). Geometric installation preserves the ordered structure of the alphabet.

Table 2: Rail ablation of LEXIS-4B. Performance difference with each rail masked, while preserving the rest, and with the input series masked entirely. The shape rail consists of the smooth shape $\tilde{u}_p$ and the transient block e_p, and we report the effect of masking each individually.

	TSAQA		TimeSeriesExam		TemporalBench	
Rail masked	acc.	Δ	acc.	Δ	acc.	Δ
None (intact)	77.09		60.72		38.06	
Envelope	51.27	−25.82	41.82	−18.90	37.10	−0.95
Shape	66.73	−10.36	57.64	−3.08	36.10	−1.96
w/o Smooth	67.09	−10.00	58.71	−2.01	36.24	−1.82
w/o Transient	76.64	−0.45	59.29	−1.43	37.56	−0.50
Series masked	49.64	−27.45	39.14	−21.58	34.74	−3.32

4 EXPERIMENTS

We evaluate LEXIS on three time series understanding benchmarks and address three research questions. **RQ1:** How does the series representation affect understanding? **RQ2:** Does the model use each component in LEXIS? **RQ3:** How much does the geometry of the symbol embeddings contribute?

4.1 SETUP

Benchmarks. We evaluate models on TSAQA (Jing et al., 2026), TimeSeriesExam (TSE) (Cai et al., 2024), and TemporalBench (TB) (Weng et al., 2026). TSAQA and TSE assess properties such as pattern recognition, classification, comparison, and correlation. TB emphasizes analysis under contextual information and is organized in tiers: T1 asks context free questions about series properties, T3 and T4 directly test contextual and event understanding, and following the leaderboard standard we exclude the context free forecasting tier T2. For the component analysis in section 4.3, we additionally use a diagnostic set of 2,300 questions. Each question isolates one property of a series drawn from the benchmark datasets and has an analytically computed label (Appendix B.4).

Trained models. To isolate the effect of representation, we train four Qwen3-4B variants with the same data sources and optimization settings. *Decimal* exposes the scale and standardized numerical values directly. *Soft token* uses a frozen pretrained series encoder (Goswami et al., 2024) and a trained projector to insert continuous representations into the token stream. Because these representations cannot be generated as targets, this variant uses only the text target examples, and we compensate for the excluded series writing examples by training it for the same number of steps as LEXIS-4B (Appendix B.2). *Envelope-only* retains the envelope rail (μ_p, σ_p), discards the shape rail, and adds no new vocabulary. LEXIS-4B uses the complete surface $\mathcal{S}(x)$ described in section 3.

All reported results use held out evaluation questions and labels; the tokenizer and model are fitted and fine-tuned only on training data.

4.2 **RQ1:** REPRESENTATION COMPARISON

Overall performance. Table 1 reports class-wise accuracy. LEXIS-4B achieves 77.09% on TSAQA and 60.72% on TSE, outperforming every evaluated LLM and TSLM, including GPT-4o and ChatTS-14B. On TB, it scores 38.06%, behind only GPT-4o at 40.64% and ChatTS-14B at 41.01%. Among the four trained variants, LEXIS-4B performs best on all three benchmarks and improves the Qwen3-4B backbone by 31.09%, 13.40%, and 4.45%. The gains are largest on questions about series features and remain below 2% on the contextual tiers T3 and T4, a limitation discussed in section 5.

Matched representations. Decimal and Envelope-only achieve similar accuracy across the three benchmarks: 73.73% versus 72.18% on TSAQA, 58.31% versus 57.24% on TSE, and 35.15% versus 33.02% on TB. Compressing each patch to its mean and standard deviation therefore preserves most of the understanding obtained from the complete decimal sequence, while the additional numerical precision contributes little. Soft token also performs similarly to Envelope-only on TSAQA and TSE

(71.73% versus 72.18% and 55.90% versus 57.24%). Its advantage on TB (37.65% versus 33.02%) is driven primarily by T1. By explicitly representing the shape $(\tilde{u}_p, e_p)$, LEXIS-4B improves over Envelope-only by 4.91%, 3.48%, and 5.04%, and over Soft token by 5.36% and 4.82% on TSAQA and TSE, with a tie on TB (0.41%). It also exceeds Decimal on every benchmark, with its largest class-wise margins on data transformation, classification, causality analysis, and T4.

4.3 **RQ2:** RAIL ABLATION

The compositional structure of LEXIS permits controlled interventions on the envelope (μ_p, σ_p), the shape rail $(\tilde{u}_p, e_p)$, and individual shape axes $B_{:,k}$. We use three perturbations. **Masking** replaces a component with a neutral value, such as a typical envelope or symbols for a flat curve. **Resampling** replaces a component with one drawn from a random patch in the same dataset, removing its information while retaining a plausible value. **Shuffling** permutes a component across patches within a series, preserving its values but breaking their temporal structure.

Rail contribution. Table 2 reports the effect of masking each rail. The envelope makes the largest contribution on TSAQA and TSE: removing it reduces accuracy by 25.82% and 18.90%, respectively. Masking the shape rail costs 10.36% on TSAQA, more than 37% of the total contribution of the series, and 3.08% on TSE. On TB, the shape rail has the larger effect, reducing accuracy by 1.96% compared with 0.95% for the envelope. Within the shape rail, the smooth component accounts for most of the loss, while transients provide a smaller accuracy contribution. Isolating the transients, however, also keeps the smooth symbols faithful to the curve (Appendix A.2, table 5), and their value is therefore not measured by this accuracy alone.

Trait attribution. Because the components of LEXIS have known referents, their expected contributions to individual properties, or traits, can be tested directly. We perturb one rail or axis group at a time on questions drawn from all three benchmarks. Under resampling (fig. 4, left), the model exhibits selective dependence on the relevant components. The patch mean strongly affects *direction & trend* (-14%) and *level & spread* (-12%). The patch standard deviation contributes to *anomalies & transients* (-13%), *noise level* (-10%), and *continuation & order* (-11%). The leading shape axes, which encode slope and curvature, affect *local shape & turning points* (-15%) and *continuation & order* (-11%). The middle and later axes, which encode finer variation, contribute to *transforms* (-6.5% and -11%) and *noise level* (-6.9%). Composite traits such as *transforms* and *anomalies & transients* depend on several components together.

Shuffling tests whether the model uses the temporal position of each component. Breaking this alignment reduces performance on *transforms*, *direction & trend*, and *local shape & turning points* by 17%, 15%, and 24%, respectively (fig. 4, right). In contrast, *level & spread*, which does not depend on patch order, changes only slightly.

Together, these interventions show that the model uses the components selectively and relies on their temporal positions when the queried property requires order. The model therefore reads LEXIS as an ordered, component-wise description of the series.

4.4 **RQ3:** INSTALLATION ABLATION

Geometric initialization encodes the structure of the shape alphabet in the embedding space, and $\mathcal{L}_{\text{leash}}$ preserves that structure during training. We ablate both components under the same corpus and optimization settings. Without the leash, the symbols retain geometric initialization but move freely during training. Without either component, they are initialized at the mean of the pretrained token embeddings with a small Gaussian perturbation.

Table 3: Installation ablation of LEXIS over geometric initialization (GI) and the leash regularization loss $\mathcal{L}_{\text{leash}}$. Model performance on TSAQA, TimeSeriesExam (TSE), and TemporalBench (TB).

	TSAQA	TSE	TB
base model	46.00	47.32	33.61
+ LEXIS	77.09	60.72	38.06
w/o leash	76.91	58.98	37.51
w/o GI and leash	72.72	56.57	36.51

Relative to mean initialization with unconstrained training, the full installation improves accuracy by 4.37% on TSAQA, 4.15% on TSE, and 1.55% on TB. Geometric initialization accounts for most of this gain. The leash adds small further gains over geometric initialization alone. The installed

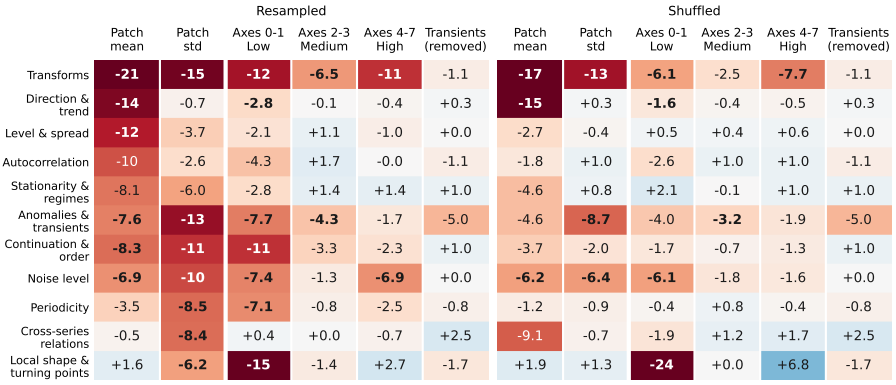


Figure 4: Performance change of LEXIS-4B with perturbed rails and axis groups. **Left:** each channel is resampled from a random patch of the same evaluation set, removing its content. **Right:** each channel is shuffled across the patches of the series, keeping its content and destroying its position. The transient block is emptied in both panels. Bold marks $p < 0.01$ under an exact McNemar test.

structure also remains stable: after alignment, all 50 symbol embeddings preserve their assigned coordinates and radius to within 0.6%, and every axis retains its initial level order. At the same time, training places 15% of each embedding's squared norm in the $d - F$ directions left unconstrained by the leash (Appendix A.4). Embeddings trained freely from the vocabulary mean recover the level order on every axis but end in a looser, higher rank geometry than the full installation (fig. 3; Appendix A.4).

5 LIMITATIONS

We identify two main limitations. First, the shape symbols are aligned primarily through general time series concepts, including trend, periodicity, level, scale, and transients. Their connection to specialized terminology in clinical, retail, or energy domains depends on the corresponding examples in the training corpus and may therefore be weaker. Future work could have the model first translate domain specific descriptions into general time series language, letting it reason in the space where its symbols are aligned.

Second, the improvement on TemporalBench is 4.45%, compared with 31.09% on TSAQA and 13.40% on TimeSeriesExam, and remains below 2% on its contextual tiers T3 and T4. These tiers contain longer records, more series, and substantially more text (Appendix B.4), placing greater demands on contextual and language reasoning. Although LEXIS makes the series structure accessible, its use remains constrained by the capabilities of the 4B backbone. Extending the interface to larger language models is therefore an important direction for future work.

6 CONCLUSION

We introduced LEXIS, a compact and auditable representation that expresses time series structure in a form accessible to pretrained LLMs. LEXIS separates level, scale, and shape, representing them through numerals and compositional symbols, each denoting one decodable curve component. Geometric installation preserves the structure of this vocabulary during training.

Across three understanding benchmarks, LEXIS-4B improves the Qwen3-4B backbone by 31.09%, 13.40%, and 4.45% and outperforms all evaluated general-purpose LLMs and TSLMs on two benchmarks. Controlled comparisons show that patch statistics nearly match complete decimals and projected encoder features, whereas explicit shape symbols provide consistent further gains. Component-wise interventions confirm that the model uses the intended components, and installation ablations establish the value of preserving symbol geometry. These results establish LEXIS as an effective and principled interface between time series and pretrained LLMs.

AI USAGE STATEMENT

In this work, we used generative AI tools to generate synthetic datasets and partially assist with method implementation. We used a vision language model to generate textual descriptions of time series plots. All generated samples are strictly gated by quality. We used coding agents to assist with a portion of implementation and test cases for the method. All code files are reviewed by at least two authors and verified for correctness. We additionally used LLMs to polish the language in human-written paper drafts. We did not use generative AI for other tasks that require disclosure. We reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. GIFT-Eval: A benchmark for general time series forecasting model evaluation. In *NeurIPS 2024 Workshop on Time Series in the Age of Large Models*, 2024.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, 2022.

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

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Yifu Cai, Arjun Choudhry, Mononito Goswami, and Artur Dubrawski. TimeSeriesExam: A time series understanding exam. In *NeurIPS Workshop on Time Series in the Age of Large Models*, 2024.

Defu Cao, Furong Jia, Sercan O. Arik, Tomas Pfister, Yixiang Zheng, Wen Ye, and Yan Liu. TEMPO: Prompt-based generative pre-trained transformer for time series forecasting. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=YH5w12OUuU.

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*, 2026.

Daiwei Chen, Zhoutong Fu, Chengming Jiang, Haichao Zhang, Ran Zhou, Tan Wang, Chunnan Yao, Guoyao Li, Rui Cai, Yihan Cao, Ruijie Jiang, Fedor Borisyuk, Jianqiang Shen, Jingwei Wu, and Ramya Korlakai Vinayak. Grounded token initialization for new vocabulary in LMs for generative recommendation. *arXiv preprint arXiv:2604.02324*, 2026a.

Xinye Chen, Erin Carson, and Cheng Kang. LLM-ABBA: Understanding time series via symbolic approximation. *IEEE Transactions on Signal Processing*, 74:1218–1232, 2026b. doi: 10.1109/TSP.2026.3662011.

Robert B. Cleveland, William S. Cleveland, Jean E. McRae, and Irma Terpenning. STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, 6(1):3–73, 1990.

Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. Moshi: a speech-text foundation model for real-time dialogue. *arXiv preprint arXiv:2410.00037*, 2024.

Konstantin Dobler, Desmond Elliott, and Gerard de Melo. Token distillation: Attention-aware input embeddings for new tokens. In *International Conference on Learning Representations (ICLR)*, 2026.

Steven Elsworth and Stefan Güttel. ABBA: Adaptive Brownian bridge-based symbolic aggregation of time series. *Data Mining and Knowledge Discovery*, 34:1175–1200, 2020. doi: 10.1007/s10618-020-00689-6.

Elizabeth Fons, Rachneet Kaur, Soham Palande, Zhen Zeng, Tucker Balch, Manuela Veloso, and Svitlana Vyetrenko. Evaluating large language models on time series feature understanding: A comprehensive taxonomy and benchmark. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Arushi Goel, Sreyan Ghosh, Jaehyeon Kim, Sonal Kumar, Zhifeng Kong, Sang-gil Lee, Chao-Han Huck Yang, Ramani Duraiswami, Dinesh Manocha, Rafael Valle, and Bryan Catanzaro. Audio Flamingo 3: Advancing audio intelligence with fully open large audio language models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 46789–46856, 2025.

Siavash Golkar, Mariel Pettee, Michael Eickenberg, Alberto Bietti, Miles Cranmer, Geraud Krawezik, Francois Lanusse, Michael McCabe, Ruben Ohana, Liam Parker, Bruno Régaldo-Saint Blancard, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. xVal: A continuous numerical tokenization for scientific language models. In *NeurIPS 2023 Workshop on AI for Science: from Theory to Practice*, 2023. URL https://arxiv.org/abs/2310.02989.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A family of open time-series foundation models. In *International Conference on Machine Learning*, volume 235 of *PMLR*, pp. 16115–16152, 2024.

Leon Götz, Marcel Kollovieh, Stephan Günnemann, and Leo Schwinn. Byte pair encoding for efficient time series forecasting. In *International Conference on Machine Learning*, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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*, volume 36, pp. 19622–19635, 2023.

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

Tong Guan, Sheng Pan, Johan Barthelemy, Zhao Li, Yujun Cai, Cesare Alippi, Ming Jin, and Shirui Pan. TimeOmni-VL: Unified models for time series understanding and generation. In *International Conference on Machine Learning*, 2026b.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

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

Ming Jin, Yifan Zhang, Wei Chen, Kexin Zhang, Yuxuan Liang, Bin Yang, Jindong Wang, Shirui Pan, and Qingsong Wen. Position: What can large language models tell us about time series analysis. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *PMLR*, pp. 22260–22276, 2024b.

Baoyu Jing, Sanhorn Chen, Lecheng Zheng, Boyu Liu, Zihao Li, Jiaru Zou, Tianxin Wei, Zhining Liu, Zhichen Zeng, Ruizhong Qiu, Xiao Lin, Yuchen Yan, Dongqi Fu, Jingchao Ni, Jingrui He, and Hanghang Tong. TSAQA: Time series analysis question and answering benchmark. In *Proceedings of the Fifth Workshop on Generation, Evaluation and Metrics (GEM)*, pp. 944–979. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.gem-main.74.

Raviraj Joshi, Utkarsh Vaidya, Sanjay Singh Chauhan, and Niranjan Wartikar. Beyond initialization loss: A systematic study of token embedding initialization strategies for LLM vocabulary extension. *arXiv preprint arXiv:2608.03494*, 2026.

Jaeho Kim, Changhun Oh, Seokhyun Lee, Irina Rish, and Changhee Lee. Representing time series as structured programs for LLM reasoning. *arXiv preprint arXiv:2606.12481*, 2026.

Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=cGDAkQo1C0p.

Tomoyoshi Kimura, Denizhan Kara, Jinyang Li, Hongjue Zhao, Yigong Hu, Yizhuo Chen, Xiaomin Ouyang, Shengzhong Liu, and Tarek Abdelzaher. Dywave: Event-aligned dynamic tokenization for heterogeneous IoT sensing signals. In *International Conference on Machine Learning*, 2026.

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)*, pp. 29736–29753. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1437.

Patrick Langer, Thomas Kaar, Max Rosenblattl, Maxwell A. Xu, Winnie Chow, Martin Maritsch, Robert Jakob, Ning Wang, Juncheng Liu, Aradhana Verma, Brian Han, Daniel Seung Kim, Henry Chubb, Scott Ceresnak, Aydin Zahedivash, Alexander Tarlochan Singh Sandhu, Fatima Rodriguez, Daniel McDuff, Elgar Fleisch, Oliver Aalami, Filipe Barata, and Paul Schmiedmayer. OpenTSLM: Time-series language models for reasoning over multivariate medical text- and time-series data. In *International Conference on Machine Learning*, 2026.

Amit Arnold Levy and Mor Geva. Language models encode numbers using digit representations in base 10. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pp. 385–395, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.naacl-short.33. URL https://aclanthology.org/2025.naacl-short.33/.

Musheng Li, Ziying Zhang, Cheng Jin, and Yuantao Gu. Continuity and ordinality matter: Constraining time series tokens for effective time series analysis with large language models. *arXiv preprint arXiv:2605.28866*, 2026.

Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing SAX: A novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2):107–144, 2007. doi: 10.1007/s10618-007-0064-z.

Jiafeng Lin, Yuxuan Wang, Jialong Wu, Huakun Luo, Zhongyi Pei, and Jianmin Wang. Thoth: Mid-training bridges LLMs to time series understanding. *arXiv preprint arXiv:2603.01042*, 2026.

Hao Liu, Wilson Yan, and Pieter Abbeel. Language quantized autoencoders: Towards unsupervised text-image alignment. In *Advances in Neural Information Processing Systems*, volume 36, pp. 4382–4395, 2023a. doi: 10.52202/075280-0195.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34892–34916, 2023b.

Haoxin Liu, Shangqing Xu, Zhiyuan Zhao, Lingkai Kong, Harshavardhan Kamarthi, Aditya B. Sasanur, Megha Sharma, Jiaming Cui, Qingsong Wen, Chao Zhang, and B. Aditya Prakash. Time-MMD: Multi-domain multimodal dataset for time series analysis. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, volume 37, pp. 77888–77933, 2024.

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, 2025a. doi: 10.18653/v1/2025.naacl-long.383.

Penghang Liu, Elizabeth Fons, Annita Vapsi, Mohsen Ghassemi, Svitlana Vyetrenko, Daniel Borrajo, Vamsi K. Potluru, and Manuela Veloso. TS-Agent: Understanding and reasoning over raw time series via iterative insight gathering. In *NeurIPS 2025 Workshop on Foundations of Reasoning in Language Models*, 2025b.

Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. In *Advances in Neural Information Processing Systems*, volume 35, pp. 9881–9893, 2022. doi: 10.52202/068431-0718.

Simon Malinowski, Thomas Guyet, René Quiniou, and Romain Tavenard. 1d-SAX: A novel symbolic representation for time series. In *Advances in Intelligent Data Analysis XII (IDA)*, volume 8207 of *Lecture Notes in Computer Science*, pp. 273–284, 2013. doi: 10.1007/978-3-642-41398-8_24.

Stéphane G. Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993. doi: 10.1109/78.258082.

Luca Masserano, Abdul Fatir Ansari, Boran Han, Xiyuan Zhang, Christos Faloutsos, Michael W. Mahoney, Andrew Gordon Wilson, Youngsuk Park, Syama Rangapuram, Danielle C. Maddix, and Yuyang Wang. Enhancing foundation models for time series forecasting via wavelet-based tokenization. In *International Conference on Machine Learning*, volume 267 of *PMLR*, pp. 43248–43275, 2025.

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: VQ-VAE made simple. In *International Conference on Learning Representations*, 2024.

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

Benjamin Minixhofer, Fabian Paischer, and Navid Rekabsaz. WECHSEL: Effective initialization of subword embeddings for cross-lingual transfer of monolingual language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, 2022.

OpenAI. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2020.

Felix Parker, Nimeesha Chan, Chi Zhang, and Kimia Ghobadi. Eliciting chain-of-thought reasoning for time series analysis using reinforcement learning. *arXiv preprint arXiv:2510.01116*, 2025.

Felix Parker, Nimeesha Chan, Chi Zhang, and Kimia Ghobadi. TsLLM: Augmenting LLMs for general time series understanding and prediction. In *International Conference on Machine Learning*, 2026.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models. In *Robotics: Science and Systems (RSS)*, 2025.

Paul Quinlan, Qingguo Li, and Xiaodan Zhu. Chat-TS: Enhancing multi-modal reasoning over time-series and natural language data. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5621–5647. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.eacl-long.263.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026. URL https://qwen.ai/blog?id=qwen3.6-35b-a3b.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H. Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. Recommender systems with generative retrieval. In *Advances in Neural Information Processing Systems*, 2023.

Patrick Schäfer and Mikael Högqvist. SFA: A symbolic fourier approximation and index for similarity search in high dimensional datasets. In *International Conference on Extending Database Technology (EDBT)*, pp. 516–527, 2012. doi: 10.1145/2247596.2247656.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, 2016. doi: 10.18653/v1/P16-1162.

Yu Shi, Zongliang Fu, Shuo Chen, Bohan Zhao, Wei Xu, Changshui Zhang, and Jian Li. Kronos: A foundation model for the language of financial markets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 25366–25373, 2026. doi: 10.1609/aaai.v40i30.39730.

Aaditya K. Singh and DJ Strouse. Tokenization counts: the impact of tokenization on arithmetic in frontier LLMs. *arXiv preprint arXiv:2402.14903*, 2024. URL https://arxiv.org/abs/2402.14903.

Dimitris Spathis and Fahim Kawsar. The first step is the hardest: Pitfalls of representing and tokenizing temporal data for large language models. *Journal of the American Medical Informatics Association*, 31(9):2151–2158, 2024. doi: 10.1093/jamia/ocae090.

Sabera Talukder, Yisong Yue, and Georgia Gkioxari. TOTEM: TOkenized time series EMbeddings for general time series analysis. *Transactions on Machine Learning Research*, 2024.

Mingtian Tan, Mike A. Merrill, Vinayak Gupta, Tim Althoff, and Thomas Hartvigsen. Are language models actually useful for time series forecasting? In *Advances in Neural Information Processing Systems*, volume 37, pp. 60162–60191, 2024.

Changli Tang, Wenyi Yu, Guangzhi Sun, Xianzhao Chen, Tian Tan, Wei Li, Lu Lu, Zejun Ma, and Chao Zhang. SALMONN: Towards generic hearing abilities for large language models. In *International Conference on Learning Representations*, 2024.

Xiaoyu Tao, Shilong Zhang, Mingyue Cheng, Daoyu Wang, Tingyue Pan, Bokai Pan, Changqing Zhang, and Shijin Wang. From values to tokens: An LLM-driven framework for context-aware time series forecasting via symbolic discretization. In *Proceedings of the Thirty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3043–3051, 2026. doi: 10.24963/ijcai.2026/338.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. Do NLP models know numbers? probing numeracy in embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 12694–12702, 2025a. doi: 10.1609/aaai.v39i12.33384.

Xinyue Wang, Jiacheng Pang, Kun Zhou, Kexin Zhang, Defu Cao, Fan Feng, Faisal, Songyao Jin, Yan Liu, and Biwei Huang. TimeBraid: Unifying time series and language for understanding and forecasting. *arXiv preprint arXiv:2609.29792*, 2026.

Yilin Wang, Peixuan Lei, Jie Song, Yuzhe Hao, Tao Chen, Yuxuan Zhang, Lei Jia, Yuanxiang Li, and Zhongyu Wei. ITFormer: Bridging time series and natural language for multi-modal QA with large-scale multitask dataset. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63324–63344, 2025b.

Subo Wei, Jianqi Gao, Mingyan Fan, Shaorong Xie, Xinzhi Wang, and Yongpeng Dong. ShapeLex: Decoupling local shape symbolization and global scale modeling for text-controlled time series generation. *arXiv preprint arXiv:2609.24003*, 2026.

Yunshi Wen, Tengfei Ma, Tsui-Wei Weng, Lam M. Nguyen, and Anak Agung Julius. Abstracted shapes as tokens – a generalizable and interpretable model for time-series classification. In *Advances in Neural Information Processing Systems*, volume 37, pp. 92246–92272, 2024. doi: 10.52202/079017-2929.

Muyan Weng, Defu Cao, Wei Yang, Yashaswi Sharma, and Yan Liu. TemporalBench: A benchmark for evaluating LLM-based agents on contextual and event-informed time series tasks. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD)*, pp. 9997–10008, 2026. doi: 10.1145/3770855.3817485.

Andrew Robert Williams, Arjun Ashok, Étienne Marcotte, Valentina Zantedeschi, Jithendaraa Subramanian, Roland Riachi, James Requeima, Alexandre Lacoste, Irina Rish, Nicolas Chapados, and Alexandre Drouin. Context is key: A benchmark for forecasting with essential textual information. In *International Conference on Machine Learning*, volume 267 of *PMLR*, pp. 66887–66944, 2025.

Junfeng Wu, Yi Jiang, Chuofan Ma, Yuliang Liu, Hengshuang Zhao, Zehuan Yuan, Song Bai, and Xiang Bai. Liquid: Language models are scalable and unified multi-modal generators. *International Journal of Computer Vision*, 134(1):39, 2026. doi: 10.1007/s11263-025-02639-5.

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. *Proceedings of the VLDB Endowment*, 18(8):2385–2398, 2025. doi: 10.14778/3742728.3742735.

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

Fan Yang and John Paparrizos. SPARTAN: Data-adaptive symbolic time-series approximation. *Proceedings of the ACM on Management of Data*, 3(3):1–30, 2025. doi: 10.1145/3725357.

Fangxu Yu, Xingang Guo, Lingzhi Yuan, Haoqiang Kang, Hongyu Zhao, Lianhui Qin, Furong Huang, Bin Hu, and Tianyi Zhou. TSRBench: A comprehensive multi-task multi-modal time series reasoning benchmark for generalist models. In *International Conference on Machine Learning*, 2026a.

Fangxu Yu, Hongyu Zhao, and Tianyi Zhou. TS-Reasoner: Aligning time series foundation models with LLM reasoning. *Transactions on Machine Learning Research*, 2026b.

Lijun Yu, Yong Cheng, Zhiruo Wang, Vivek Kumar, Wolfgang Macherey, Yanping Huang, David A. Ross, Irfan Essa, Yonatan Bisk, Ming-Hsuan Yang, Kevin Murphy, Alexander G. Hauptmann, and Lu Jiang. SPAE: Semantic pyramid autoencoder for multimodal generation with frozen LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 52692–52704, 2023.

Lijun Yu, José Lezama, Nitesh B. Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, Agrim Gupta, Xiuye Gu, Alexander G. Hauptmann, Boqing Gong, Ming-Hsuan Yang, Irfan Essa, David A. Ross, and Lu Jiang. Language model beats diffusion – tokenizer is key to visual generation. In *International Conference on Learning Representations*, 2024.

Haochuan Zhang, Chunhua Yang, Jie Han, Liyang Qin, and Xiaoli Wang. TempoGPT: Enhancing time series reasoning in multi-modal language model via quantizing embedding. *Neurocomputing*, 664:132035, 2026a. doi: 10.1016/j.neucom.2025.132035.

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 *NeurIPS 2025 Workshop on Recent Advances in Time Series Foundation Models (BERT2S)*, 2025.

Xinyu Zhang, Shanshan Feng, Xutao Li, Kenghong Lin, Fan Li, and Pengfei Jia. From tokenizer bias to backbone capability: A controlled study of LLMs for time series forecasting. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, pp. 1928–1939, 2026b. doi: 10.1145/3770854.3780318.

Yunhao Zhang, Ruiying Qi, Jiale Zheng, Jianfeng Zhang, Lujia Pan, and Junchi Yan. Time series as language: A universal tokenizer for general-purpose time series foundation models. *arXiv preprint arXiv:2606.09861*, 2026c.

Yue Zhao, Yuanjun Xiong, and Philipp Krähenbühl. Image and video tokenization with binary spherical quantization. In *International Conference on Learning Representations*, 2025.

Liangwei Nathan Zheng, Chang George Dong, Wei Emma Zhang, Lin Yue, Miao Xu, Olaf Maennel, and Weitong Chen. Understanding why large language models can be ineffective in time series analysis: The impact of modality alignment. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD)*, pp. 4026–4037, 2025. doi: 10.1145/3711896.3737169.

Yifu Zheng, Étienne Marcotte, Arjun Ashok, Andrew Robert Williams, Xiaoyin Sun, Alexandre Drouin, and Valentina Zantedeschi. Overcoming the modality gap in context-aided forecasting. In *International Conference on Machine Learning*, 2026.

Tianyi Zhou, Deqing Fu, Mahdi Soltanolkotabi, Robin Jia, and Vatsal Sharan. FoNE: Precise single-token number embeddings via fourier features. In *International Conference on Learning Representations*, 2026.

Zihao Zhou and Rose Yu. Can LLMs understand time series anomalies? In *International Conference on Learning Representations (ICLR)*, 2025.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Xuehui Wang, Yue Cao, Yangzhou Liu, et al. InternVL3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.

Appendix Contents

A Implementation Details **17**

A.1 Surface 17

A.2 Transient Detector 18

A.3 Tokenizer 19

A.4 Installation 21

A.5 Decoding 23

B Experimental details and additional results **24**

B.1 Training corpus 24

B.2 Trained Representation Baselines 26

B.3 Training Hyperparameters 26

B.4 Evaluation protocol 26

B.5 Accuracy without the series 28

C Full Related Work **29**

C.1 Multimodal Large Language Models 29

C.2 Time Series Language Models 29

C.3 Time Series Decomposition and Numeracy 29

C.4 Vocabulary Extension in Pretrained Language Models 30

A Implementation Details

A.1 Surface

```
System: You analyze time series.
User: Is the given time series a white noise process?
<TS_BEGIN> len=1024 scale=-0.0214 1.83
-0.13 1.29 <c0_5><c1_2><c2_4><c3_2><c4_2><c5_2><c6_1><c7_1> ...
<EV_START> t=625-626 +3 <EV_END> <TS_END>
Choices: A: Yes; B: No.
Answer with one option letter and its content.
Assistant: A: Yes
```

Figure 5: Example prompt containing the LEXIS representation. The envelope rail (blue) records the mean and standard deviation of each patch as numerals. The shape rail describes the smooth subpatch curve with symbols (green) and records sparse transients explicitly (red).

Grammar. LEXIS serializes a time series as a sequence over the pretrained vocabulary extended with 50 shape symbols across 8 axes (section 3). As shown in fig. 5, the special tokens <TS_BEGIN> and <TS_END> delimit the series. A header records its length, span mean, and span standard deviation. The series is then divided into patches of length $L = 16$ and serialized in temporal order. Each patch contains its envelope (μ_p, σ_p) followed by 8 shape symbols in axis order. After the patch sequence, the transient block lists each detected excursion by its position, extent, and signed magnitude in units of the local standard deviation.

Vocabulary. LEXIS adds 50 shape tokens, denoted <ck_i> for axes $k = 0, \ldots, 7$ (token names index axes from 0, whereas the text numbers them from 1), and four structural tokens that delimit the time series and transient blocks. The vocabulary of Qwen3-4B consequently increases from 151,669 to 151,723 tokens. The shape token embeddings are initialized to preserve the axis and level geometry described in section 3.4.

Token cost. Table 4 compares the number of tokens required by LEXIS with decimal serialization at three significant figures on the evaluation benchmarks. The envelope rail is slightly longer than the shape rail because each numeral is encoded digit by digit, while each shape symbol occupies one token. Overall, the complete LEXIS surface uses approximately 25% as many tokens as decimal serialization.

Table 4: Mean tokens per series span on the evaluation sets: the decimal rendering tokenizes raw numerals, and LEXIS tokenizes series into the envelope rail and the shape rail, where patch shapes are represented by 8 shape symbols. The ratio denotes token count of Decimal over that of LEXIS.

	TSAQA	TimeSeriesExam	TemporalBench
Decimal	477	5,744	1,259
LEXIS	134	1,260	305
envelope rail	67	653	155
shape rail	52	533	126
ratio	3.6	4.6	4.1

A.2 TRANSIENT DETECTOR

Why transients are separated. The shape basis spans 8 smooth directions within a patch. An isolated spike or dip lies largely outside this subspace. Projecting such an excursion onto the basis distributes its energy across multiple axes, distorting the encoded smooth curve. Subsequent quantization then loses information about both the curve and the excursion, as illustrated in fig. 6. LEXIS therefore separates the shape rail into a smooth component and an explicit transient component. Transients are detected and removed before the smooth shape is quantized, then recorded by position, extent, and signed magnitude in a dedicated block.

Detection. We detect transients through iterative matching pursuit (Mallat & Zhang, 1993) on each patch. At every pass, a running median estimates the local level, and the largest excursion from this level becomes the next candidate. The detector fits the candidate with a short localized template and selects the width that best matches its profile. A candidate is accepted only if its magnitude reaches 2.95 patch standard deviations and its profile is sufficiently narrow. These criteria isolate large, brief departures while leaving smaller or broader variation to the smooth basis. The candidate must also exceed 5% of the spread of the complete series, preventing small fluctuations in nearly constant patches from being promoted to transients. Each accepted transient is subtracted from the patch, and the next pass searches the residual.

Transient block. Each accepted transient is represented by a triple (t_j, w_j, η_j) containing its position, half width, and signed magnitude in units of the patch standard deviation. The transient component of patch p is the sum of the corresponding pulses:

$$e_p(t) \;=\; \sum_j \eta_j \, \bar{\tau}_{w_j}(t - t_j), \qquad \bar{\tau}_w = \tau_w - \tfrac{1}{L}\textstyle\sum_t \tau_w(t), \tag{8}$$

where

$$\tau_w(t) = \max(0,\, 1 - |t|/w)$$

is a triangular pulse with half width w. For $w = 0$, τ_0 denotes a unit impulse. Subtracting the mean from each pulse gives $\bar{\tau}_w$ zero mean over the patch, and removing e_p therefore leaves the envelope mean unchanged. The symbols quantize the remaining smooth component,

$$\tilde{u}_p = u_p - e_p,$$

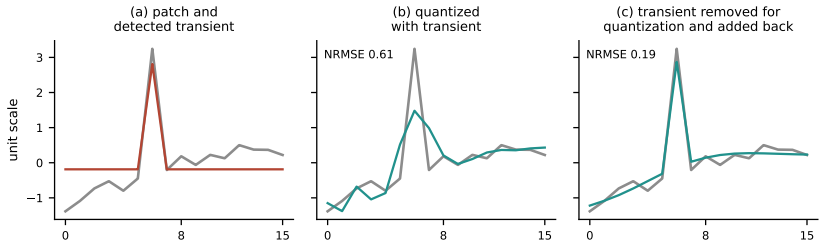


Figure 6: Effect of transient removal on shape quantization. An isolated transient cannot be represented faithfully by the 8 smooth basis directions (middle). Removing the transient (left) before quantization improves reconstruction of the smooth shape, and restoring the transient after quantization recovers the complete patch (right).

and the decoder restores e_p before applying equation 1.

In the serialized surface, each transient is written using the first and last samples affected by its pulse and its signed magnitude, as in `t=5-7 +3`. A one sample impulse is written with a single position. The transient block is included in every surface and remains empty when no transient is detected.

A.3 TOKENIZER

Shape alphabet. After removing its envelope and transients, a patch of length $L = 16$ is represented by a smooth normalized curve $\tilde{u}_p \in \mathbb{R}^L$. The tokenizer describes this curve through its coordinates along a fixed orthonormal basis $B \in \mathbb{R}^{L\times K}$ with $K = 8$. Each coordinate is rounded independently to one of a small number of levels.

Axis k has L_k levels, with

$$(L_1, \ldots, L_8) = (13, 9, 7, 5, 5, 5, 3, 3),$$

and a gain s_k that determines their range. Level i on axis k represents the coefficient

$$s_k g_k(i), \qquad g_k(i) = \frac{i - q_k}{q_k}, \qquad q_k = \lfloor L_k/2 \rfloor.$$

Thus, $g_k(i)$ lies on a uniform grid from -1 to 1. Every L_k is odd, so each axis has a neutral level $i = q_k$ whose contribution to the curve is zero.

The Cartesian product of the level sets defines

$$\prod_k L_k = 921{,}375$$

representable cells, equivalent to 19.81 bits. Each patch is encoded by 8 tokens, one per axis, drawn from a vocabulary of

$$\sum_k L_k = 50$$

shape symbols. The tokenizer is fully specified by the basis and 8 gains. After accounting for the orthonormality of B, these quantities contain 100 degrees of freedom.

Basis and gain selection. Because B is orthonormal, the reconstruction error separates into projection error outside the retained subspace and independent quantization errors along the retained axes:

$$\|\tilde{u}_p - \hat{u}_p\|^2 = \|\tilde{u}_p - BB^\top \tilde{u}_p\|^2 + \sum_k \big(a_k - s_k g_k(i_k)\big)^2, \qquad a = B^\top \tilde{u}_p. \tag{9}$$

The two terms can therefore be minimized separately. We choose B as the leading K eigenvectors of the second moment $\mathbb{E}[\tilde{u}\tilde{u}^\top]$ over the training series. This subspace maximizes the average retained energy, and its axes are ordered by the energy they capture. The leading axes consequently receive the finest quantization.

The 8 basis directions retain 77.5% of the energy of a normalized patch. For each axis, we then select s_k to minimize its scalar quantization error. With these gains, rounding reduces the retained energy by a further 3.8 percentage points. The remaining error primarily reflects fine texture within a patch that lies outside the resolution of the alphabet.

Axis count and ladder. Figure 7a shows the spectrum of the patch remainders. From axis 9 on, every axis adds a nearly constant 3.1 to 3.4%, the signature of texture without a preferred shape, and each would cost one more token per patch for a symbol with no nameable shape. The 3.8 points lost to rounding are small against the 22.5% outside the subspace, and the axis count rather than the level count therefore limits fidelity. Figure 7b shows the consequence on the benchmark series: median NRMSE falls steeply over the first four axes and flattens by the eighth, from 0.321, 0.157, and 0.204 at four axes to 0.213, 0.137, and 0.172 at 8 on TSAQA, TimeSeriesExam, and TemporalBench. The 50 symbols follow from the axis count and the ladder $(13, 9, 7, 5, 5, 5, 3, 3)$, 19.81 bits per patch allocated in energy order.

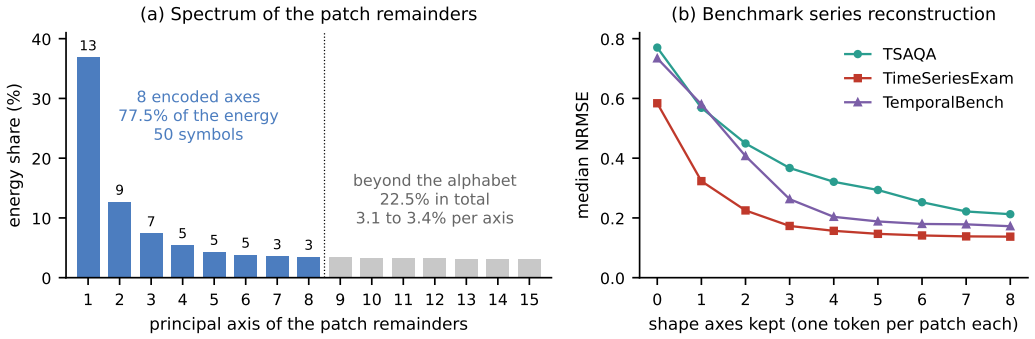


Figure 7: Axis count of the shape alphabet. (a) Energy share of each principal axis of the patch remainders over the training corpus, with the number of levels above each encoded axis. (b) Median NRMSE on the benchmark series when the decoded skeleton keeps only the first K axes, with $K = 0$ leaving the envelope and transients alone.

Patch length. A closed form fit at each length on the same series gives mean NRMSE of 0.311, 0.228, and 0.101 on TimeSeriesExam at $L = 32$, 16, and 8, for 644, 1,260, and 2,492 tokens per series. Periods shorter than 32 samples carry at least half the variance of 68% of TemporalBench series and lie beyond the smooth rail at $L = 32$, whereas $L = 8$ doubles the token count and fires the transient detector 2 to 12 times more often.

Construction. The basis and gains are estimated once from the training corpus (table 7), comprising 36.4 million patches, and remain fixed thereafter. Each span is standardized and divided into patches of length 16. Removing the envelope and detected transients from each patch yields its unit scale smooth remainder. B comprises the 8 leading eigenvectors of the second moment matrix over these remainders and captures 77.5% of their energy. For each axis, the gain is selected from 64 candidates geometrically spaced between 0.25 and 4 coefficient standard deviations. Candidates are evaluated against the empirical coefficient histogram, and the gain minimizing rounding error is selected. The resulting gains range from 1.5 to 2.3 coefficient standard deviations and clip between 4% and 14% of patches on their respective axes. The number of levels per axis is odd to provide a neutral level and decreases with the energy order of the axes. Under the same bit budget, alternative allocations change the reconstruction error energy by less than 0.1%.

Fidelity. Table 5 evaluates reconstruction of the benchmark series from their serialized surfaces. We report the median normalized root mean square error (NRMSE) between the decoded surface and the standardized original series. We also report the within patch variance explained by the rounded coordinates and by the unrounded projection $BB^\top \tilde{u}_p$, together with the fraction of patches containing a detected transient.

Rounding reduces R^2 by 4 to 6 percentage points beyond the loss from projection. The transient block has the greatest effect when excursions are frequent. On TemporalBench, transients occur in 18.6% of patches, and removing the block increases median NRMSE from 0.172 to 0.310.

Table 5: Reconstruction fidelity with patch length $L = 16$. NRMSE is the median across series between the decoded surface and the standardized original, reported with and without the transient block. The two R^2 columns give the median across patches for the rounded coordinates and the unrounded projection. The final column reports the fraction of patches containing a transient.

	series	NRMSE	w/o trans.	R^2 rounded	R^2 unrounded	transients
TSAQA	2,682	0.212	0.268	0.87	0.92	10.0%
TimeSeriesExam	992	0.137	0.150	0.89	0.93	3.0%
TemporalBench	2,775	0.172	0.310	0.89	0.95	18.6%
Synthetic validation	296	0.098	0.101	0.74	0.80	3.7%

Alphabet usage. The factorized alphabet produces dense supervision for individual symbols even though complete cells remain sparse. Across the training corpus of 36.4 million patches, the rarest of the 50 symbols occurs 1.06 million times. Only 58% of the 921,375 cells occur at all; among the observed cells, the median frequency is 22 and 7.8% occur once. The sum of the marginal level entropies across axes is 17.25 of the nominal 19.81 bits, or 87%.

Across the evaluation sets, the corresponding sum is 14.94 bits, or 75% of nominal capacity. The normalized entropy is 89% on TSAQA, 85% on TimeSeriesExam, and 70% on TemporalBench (fig. 8).

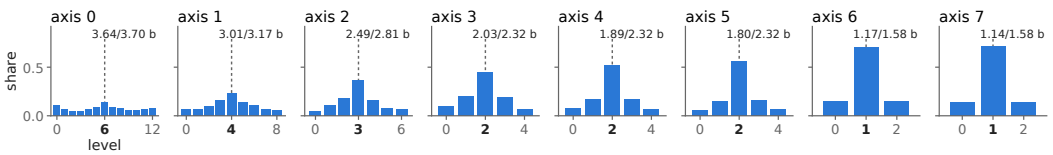


Figure 8: Usage of the shape alphabet on the evaluation sets. Each panel corresponds to one basis axis, and each bar gives the fraction of patches assigned to one level. Dashed lines mark the neutral levels, and annotations report level entropy relative to its maximum $\log_2 L_k$. All 50 symbols occur. Usage is broad along the leading axes and becomes increasingly concentrated at the neutral level on later axes.

A.4 Installation

Lift and placement. Each new symbol has a scalar coordinate and a family. For a shape symbol, the coordinate is $g_k(i)$ for level i on axis k, and the family is its axis. The two span sentinels form one additional family at coordinates ∓ 1, and the two transient delimiters form another, giving 10 families in total.

The lift in equation 4 maps each symbol to a feature vector of dimension

$$F = 10 + 2T = 26.$$

It concatenates a one hot family code with $T = 8$ sine and cosine pairs. Their frequencies,

$$\omega_t = \pi\, 10^{-t/T},$$

are geometrically spaced from 0.75π down to $\pi/10$ for $t = 1, \ldots, T$, and their amplitudes $\alpha_t = \omega_t^{-1/2}$ give greater weight to the lower frequencies.

This construction has two useful properties. First, every lifted vector has the same norm:

$$\|\phi\|^2 = 1 + \sum_t \alpha_t^2,$$

because $\sin^2(\omega_t g) + \cos^2(\omega_t g) = 1$. Second, within a family, the distance between two lifted symbols depends only on the difference between their coordinates and increases across the quantization grid. Adjacent levels are therefore nearest neighbors, and the levels of each axis follow their intended order along a curve. A linear lift would place intermediate levels inside the segment joining their neighbors, making them convex combinations of other symbols.

The placement in equation 5,

$$E_n = m + \kappa U \phi_n,$$

maps the lifted features into the F leading principal directions U of the centered pretrained embeddings. Here, m is the mean of the pretrained rows, R is their median distance from m, and $\kappa = R/\|\phi\|$. Because every lift has the same norm, every new row is placed at radius R. For Qwen3-4B, where $d = 2560$, we obtain $R = 1.0826$ and $\kappa = 0.3068$.

Leash regularization. The leash in equation 7 preserves the placed geometry during alignment. Its first term penalizes deviation between the coordinates currently encoded by a row,

$$\kappa^{-1} U^\top (E_n - m),$$

and its assigned lift ϕ_n. Its second term penalizes deviation of the row radius from R. We set

$$\beta = \frac{1}{\kappa^2 F} = 0.4087$$

to place the coordinate and radial penalties on comparable scales, and use $\lambda = 1$ to weight their sum against the next token loss. The coordinate term acts only within the F placement directions. The remaining $d - F$ directions remain available for adaptation, subject only to the constraint on the total radius.

Producibility. For a linear output head with logits $h^\top E_j$, token n can be the unique greedy output for some hidden state h exactly when its row is an exposed vertex of the convex hull of the output rows. We measure this property using the signed hull margin

$$\gamma_n = \max_{\|h\|_2 = 1} \left(h^\top E_n - \max_{j \neq n} h^\top E_j \right). \tag{10}$$

A positive margin means that some hidden state selects token n uniquely. A negative margin places the row strictly inside the convex hull of the remaining rows, and a zero margin indicates a boundary point without strict separation.

Rows initialized at the vocabulary mean lie inside the hull and cannot be selected uniquely until training moves them outward. Geometric placement puts the new rows on the shell of the pretrained vocabulary. All 54 new rows are vertices at initialization. The minimum margin is 0.020 and occurs on axis 0, whose 13 levels share one coordinate interval. All new rows remain vertices after training, with a minimum margin of 0.167.

Trainable parameters. The input embedding and output head are tied, so the same rows support both reading and generation. The pretrained rows remain frozen, and only the 54 new rows are updated during symbol alignment. We adapt the backbone with LoRA modules of rank 64 and scaling 128 on every attention and MLP projection. These modules contain 132.1 million parameters. Including the new embedding rows, the alignment stage trains 132.3 million parameters, or 3.3% of the model.

Geometry through training. Table 6 tracks the 50 shape symbol embeddings from initialization through the end of symbol alignment for LEXIS-4B and the two installation ablations in section 4.4. We report 6 quantities: the row radius relative to R; coordinate error relative to $\|\phi\|$; the fraction of squared norm outside the F placement directions; the effective rank of the centered symbol embeddings; the number of symbols appearing in their intended within axis order; and the minimum signed hull margin from equation 10. For mean initialization, coordinate error measures distance from the target geometry, although that geometry was not used to initialize the rows.

With the leash, the rows preserve their coordinates and radius to within 0.6%, while 14.7% of their squared norm moves into the free directions. Removing the leash produces almost the same total

displacement from initialization, 0.163 compared with 0.160 in embedding units. However, one third of that movement occurs within the placement directions, compared with 4% under the leash, and coordinate error rises to 5.5%. The leash therefore redirects adaptation away from the directions that encode symbol meaning without materially reducing the total movement of the embeddings.

Rows initialized at the vocabulary mean begin inside the output hull, exhibit little of the intended level order, and have effective rank 48.28. Training moves them outward and recovers the correct order along every axis, but 98.1% of their squared norm remains outside the placement subspace. Figure 3 presents the same embeddings through cosine similarity: unconstrained training recovers part of the within axis ladder, while geometric installation supplies the complete ordered structure at initialization.

Table 6: Geometry of the 50 shape symbol embeddings at initialization and after symbol alignment for LEXIS-4B and its two installation ablations. GI denotes geometric initialization; coordinate error is normalized by $\|\phi\|$; free share is the fraction of squared norm outside the placement subspace; and hull margin is defined by equation 10.

	radius / R	coord. error	free share	eff. rank	order kept	hull margin
GI, initialization	1.000	0.000	0.000	4.05	50 of 50	0.020
GI + leash, aligned	1.006	0.006	0.147	5.91	50 of 50	0.167
GI, no leash, aligned	1.009	0.055	0.138	6.62	50 of 50	0.173
Mean init, initialization	0.000	1.000	0.982	48.28	13 of 50	-0.085
Mean init, no leash, aligned	0.281	1.005	0.981	24.79	50 of 50	0.129

A.5 DECODING

Decoding a surface is arithmetic and loads no model: each patch is rebuilt by equation 1 from its envelope numerals, its symbols through equation 3, and its transients through equation 8. Figures 9 and 10 overlay decoded surfaces on series of the three benchmarks, with the three components drawn separately: the envelope traces the level and spread of each patch, the smooth reconstruction follows the curve within it, and the transient block adds back each excursion at its position. Figure 9 shows low error examples, where the temporal structure is faithfully recovered. Figure 10 shows high error examples, most of which are extremely noisy or oscillatory series, indicating a limitation of the smooth basis in recovering series with extremely high volatility; however, the rough shape and most of the temporal structure remain represented. Table 5 gives the error distribution over all series in the evaluated benchmarks.

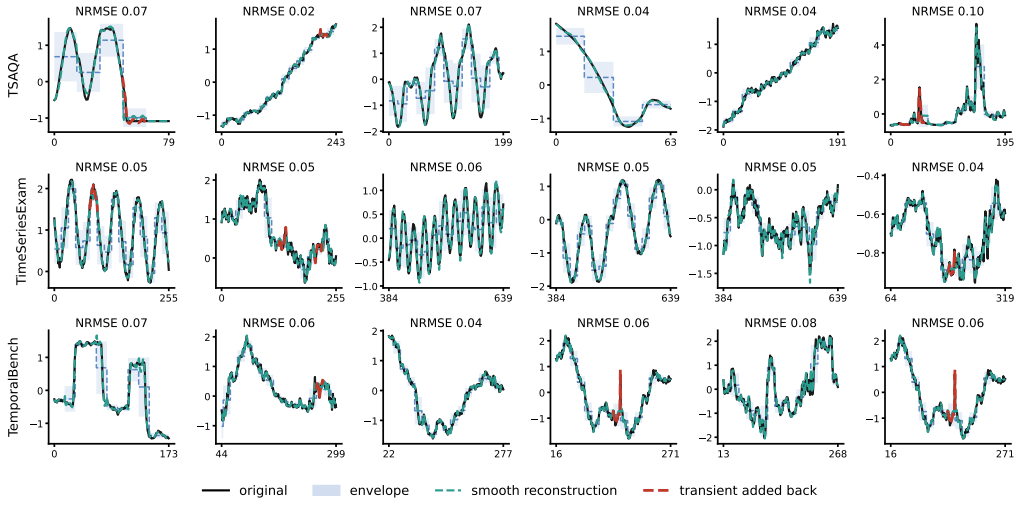


Figure 9: Low error examples of decoded surfaces overlaid on series of the evaluation benchmarks, 4 per benchmark.

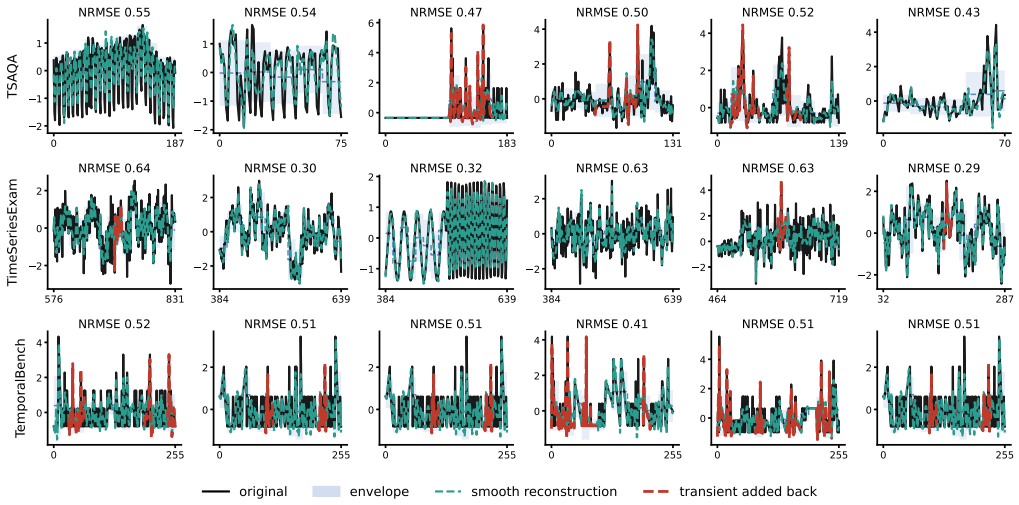


Figure 10: High error examples of decoded surfaces overlaid on series of the evaluation benchmarks, 6 per benchmark.

B Experimental details and additional results

B.1 Training corpus

We fine-tune LEXIS-4B on a Qwen3-4B backbone model in two stages. The symbol alignment stage, Stage 1, associates the shape symbols with the curves they encode and with corresponding language descriptions. The instruction fine tuning stage, Stage 2, adapts the model to time series understanding using the aligned symbols and the complete surface representation. The training corpus contains 1,336,145 examples drawn from our synthetic generator and publicly available datasets. It covers three task families: understanding (*e.g.*, trait classification, series comparison, event localization); description (*e.g.*, trajectory description, event description); and series writing (*e.g.*, event conditioned prediction, series decomposition). Table 7 summarizes the corpus.

Table 7: Training data of LEXIS-4B across the two-stage training pipeline.

Source	Series	Stage 1	Stage 2	Total
Understanding		370,000	355,480	725,480
Synthetic understanding	synthetic	370,000	197,592	567,592
TSAQA training split	real		142,888	142,888
Time-MQA understanding	real		15,000	15,000
Description		50,000	40,344	90,344
GIFT-Eval captions	real	50,000	40,344	90,344
Series writing		437,912	82,409	520,321
Synthetic series writing	synthetic	132,000	2,408	134,408
GIFT-Eval guided writing	real	245,912	40,001	285,913
CAF-7M	real	60,000	40,000	100,000
Total		857,912	478,233	1,336,145

Synthetic series. We construct a synthetic data generator to align the shape symbols with their semantics and teach the model to use them across time series tasks. The generator draws from 45 closed form processes organized into five groups: trend, periodic, stochastic, event, and composite. Because every series is generated from known parameters, task labels can be derived directly from the generating process.

Table 8: Synthetic tasks in the training data of LEXIS-4B.

Task	Stage 1	Stage 2	Total
Understanding	370,000	197,592	567,592
Trait detection	15,000	60,000	75,000
Trait classification	15,000	60,000	75,000
Component presence	50,000	16,000	66,000
Gated counterfactual	100,000	14,000	114,000
Localization	60,000	20,000	80,000
Similarity	30,000	16,000	46,000
Multivariate	30,000	11,592	41,592
Reasoning	9,994		9,994
Composition	10,006		10,006
Imputation	50,000		50,000
Series writing	132,000	2,408	134,408
Component removal	80,000		80,000
Forecasting	40,000		40,000
Multivariate forecasting	12,000	2,408	14,408
Total	502,000	200,000	702,000

We use an *aim then realize* strategy to balance the label distribution. Each example first samples a target label and then samples generation parameters that satisfy it. The generator subsequently renders those parameters into a series. Labels are therefore exact by construction, and classes are sampled with equal frequency. Table 8 reports the task distribution, and table 9 provides representative questions.

Table 9: Example questions of the synthetic tasks. `<ts>` marks the series; multiple choice questions end with their choices line and the answer instruction, omitted here.

Task	Example
Understanding	
Trait detection	Does this series loop the same pattern continuously? `<ts>`
Trait classification	Over time, do the series' fluctuations grow, shrink, or stay about the same? `<ts>`
Component presence	Is an abrupt, persistent shift to a new level present in this series? `<ts>`
Gated counterfactual	At step 393, is this series at or above 18.26? `<ts>`
Localization	What is the largest value across the span from step 196 to 239? `<ts>` Answer with a single number.
Similarity	Which later series shares the first's repeating pattern most closely? Time series 1: `<ts>` Time series 2: `<ts>` Time series 3: `<ts>` Time series 4: `<ts>`
Multivariate	Do the two series tend to move together, move in opposite directions, or move independently? Time series 1: `<ts>` Time series 2: `<ts>`
Reasoning	Which of the two events occurs first in this series? `<ts>`
Composition	Is this series several patterns overlaid on top of each other, a sequence of different patterns one after another, or a single pattern throughout? `<ts>`
Imputation	Some steps of this series were filled in by straight lines. Give the true value at each. `<ts>` Answer with one line per step, as step N: value.
Series writing	
Component removal	This series is a sum of simple parts. Give it again with the slow drift taken out, over the same 256 steps. `<ts>` Answer with the series only.
Forecasting	A rounded bump will occur around future step 208, reaching 4.92 and lasting about 13 steps. Forecast the next 64 steps. History: 192 steps. Horizon: 64 steps. `<ts>` Answer with the series only.
Multivariate forecasting	Series 1 to 2 are related series observed over the full period, including the future steps to be forecast. Series 3 is the target, observed only up to now. Forecast the target for the next 68 steps.

Real series. We draw real examples from four publicly available training resources: the GIFT-Eval pretraining corpus (Aksu et al., 2024), CAF-7M (Zheng et al., 2026), Time-MQA (Kong et al., 2025), and TSAQA (Jing et al., 2026). For GIFT-Eval, we use Qwen3.6-35B-A3B (Qwen Team, 2026), a vision language model, to generate structured descriptions of each series. We use these descriptions in both directions: captioning maps a series to language, while guided series writing maps a description to a series. This bidirectional supervision connects the surface representation with the language space of the pretrained backbone. The descriptions specify the progression of the curve, trend segments, extrema, change points, and periodic patterns, with references to their corresponding steps and values.

CAF-7M provides contextual forecasting examples in which a scenario description determines the continuation of a series. The model reads the observed history and its textual context, then generates the implied future. The Time-MQA anomaly detection subset and the TSAQA training split provide understanding questions grounded in real series.

B.2 Trained Representation Baselines

Decimal. This baseline uses the same training examples as LEXIS-4B, with each series represented by a scale frame containing `mean=` and `std=`, followed by its standardized values at three significant figures. It therefore retains the same global level and scale information as the surface. The model uses the same LoRA configuration, training stages, and optimization schedule as LEXIS-4B, with no vocabulary extension.

Soft token. This baseline implements the continuous projection interface described in section 1 using the same backbone. A frozen MOMENT-1-large encoder (Goswami et al., 2024) maps the series to one 1,024 dimensional vector per 8 samples. A two layer projector with dimensions 1,024, 2,560, and 2,560 and a GELU activation maps these vectors into the embedding space of the LLM. The projected vectors replace placeholder tokens between the series sentinels.

Training follows the same two stage structure as LEXIS-4B. The alignment stage initializes the projector randomly and trains it jointly with the LoRA adapters. The instruction fine tuning stage freezes the projector and continues training the adapters. The series encoder and pretrained token embeddings remain frozen throughout. The baseline uses the same adapter configuration and schedule as LEXIS-4B. Because projected vectors cannot be emitted by the output head, training excludes examples that require a series as output, leaving 420,000 examples in the alignment stage and 395,824 examples in the instruction fine tuning stage. To compensate for the excluded series writing examples, this variant trains to the same number of steps as LEXIS-4B in both stages (table 10).

Envelope-only. This baseline uses the same training examples, stages, and optimization recipe as LEXIS-4B, but represents each series using only its length, global scale frame, and the mean and standard deviation of each patch. It includes no shape symbols, transient block, or vocabulary extension.

B.3 Training Hyperparameters

Table 10 reports the optimization settings for both training stages. Each stage fine tunes Qwen3-4B for one epoch over packed sequences using LoRA adapters on every attention and MLP projection. Examples are packed into sequences of 4,800 tokens. Each optimizer step covers 16 packed sequences, one per GPU with 2 accumulation steps over 8 GPUs; every trained model in the paper uses this setting. The soft token variant reaches the same step counts over several epochs of its smaller corpus (section B.2). The symbol alignment stage additionally optimizes the 54 new embedding rows under the leash regularizer. The instruction fine tuning stage freezes these rows and updates only the LoRA parameters.

B.4 Evaluation protocol

Generation and matching. Every accuracy is generation based: the model decodes greedily for at most 768 new tokens and the answer is matched to an option by string match, with an unmatched row counted as wrong. The trained models are scored at their last checkpoint. The benchmarks hold

Table 10: Training hyperparameters of the two stages.

	Symbol alignment	Instruction fine-tuning
Trainable parameters	LoRA and the 54 symbol rows	LoRA
LoRA rank / scaling	64 / 128	64 / 128
Leash λ, β	1, 0.4087	off
Optimizer	AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$)	AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$)
Peak learning rate	10^{-4}	10^{-4}
Schedule	cosine, 3% warmup	cosine, 3% warmup
Weight decay	0	0
Gradient clipping	norm 1.0	norm 1.0
Sequence length	4,800 tokens, packed	4,800 tokens, packed
Batch	16 packed sequences per step	16 packed sequences per step
Epochs / steps	1 / 9,325	1 / 4,392
Precision	bf16, FlashAttention 2, gradient checkpointing	bf16, FlashAttention 2, gradient checkpointing
Seed	42	42
Hardware	8 A100 40GB	8 A100 40GB

1,100 (TSAQA test), 746 (TimeSeriesExam) and 2,775 (TemporalBench) questions, of which the 2,202 in tiers T1, T3 and T4 are reported. Paired comparisons use an exact McNemar test on the rows both models answer.

Reference models. The reference rows of table 1 are GPT-4o and GPT-4o-mini (OpenAI, 2024), Gemma-4-E4B-it (Gemma Team, 2026), Qwen2.5-7B (Qwen Team, 2024), Qwen3-1.7B (Yang et al., 2025), Mistral-7B (Jiang et al., 2023), and Llama-3.1-8B (Grattafiori et al., 2024) among general LLMs, and ChatTS (Xie et al., 2025) and TimeOmni-1 (Guan et al., 2026a) among TSLMs.

Prompt length. Table 11 measures the evaluation prompts under the extended tokenizer, splitting each into its series spans and the text around them (instruction, context and question). TemporalBench questions carry 5 to 13 times the text of TSAQA and TimeSeriesExam questions and refer to many series at once: a T3 prompt holds 7 series and 348 text tokens at the median, a T4 prompt 19 series and 763 text tokens. More question tokens demand a higher level of language and contextual understanding and reasoning that a 4B backbone may struggle with (section 5).

Table 11: Evaluation prompts of LEXIS-4B: median tokens of the text around the series, of the series spans and of the whole prompt, and the median number of series per prompt.

	series per prompt	text tokens	series tokens	prompt tokens
TSAQA	2	68	299	370
TimeSeriesExam	1	57	1,264	1,316
TemporalBench T1	2	116	846	963
TemporalBench T3	7	348	2,643	2,981
TemporalBench T4	19	763	4,440	4,990

Perturbations. A perturbation is applied to the exported evaluation rows as text surgery and scored through the same generation and matching path as the intact surface, on the rows it alters. *Masking* writes the neutral value of a component on every patch: for the envelope the corpus typical pair, for a symbol the level of the flat code on its axis; *resampling* replaces the component of every patch by that of a random patch of the same evaluation set; *shuffling* permutes the component across the patches of each series, never across series, with the two leading axes moved together so that a patch keeps its slope and curvature pair; the transient block is *emptied*. Resampling and shuffling are seeded per row. Series deletion removes the span and leaves the prompt otherwise unchanged.

Diagnostic set. The diagnostic set of section 4.3 asks one property of one series with a label computed from the values by a stated rule, on series of the evaluation benchmarks' own pool: 3,875

distinct series from 57 source datasets. Its one property questions with computed labels follow the feature understanding evaluation of Fons et al. (2024), on real series in place of synthetic ones. The set builds 11 families, direction, steepness, flatness, roughness, periodicity, level, spread, the presence and the sign of a transient, the direction of a stated range and the turning point of a window, 9,820 questions in all; the evaluation subsample takes 100 questions per family and class, 2,300 in total. Classes are balanced exactly, no source dataset contributes more than 15% of a family, and a blind probe that sees everything but the series scores at chance on every family.

Trait families. Figure 4 pools the diagnostic set, TSAQA and TimeSeriesExam with equal weight per set. Each question carries one or more trait families from the finest label its set provides: the diagnostic family, the TimeSeriesExam subcategory, the TSAQA task where the task is one property (anomaly detection, data transformation, temporal relationship), and keyword tags on the question and its gold option for the free text characterization and comparison tasks. Bold marks $p < 0.01$ under an exact McNemar test on the altered rows of the family.

B.5 Accuracy without the series

We evaluate the reference models from table 1 after removing the series from each prompt while leaving all other content unchanged. Table 12 reports the original accuracy with the series present, the conditional accuracy among questions answered after its removal, the proportion of questions declined, and the difference between the two accuracies. Because accuracy without the series excludes declined questions, a negative difference can reflect selective abstention and does not necessarily indicate improved performance. TemporalBench is aggregated over T1, T3, and T4, following the evaluation in the main text.

Table 12: Model performance on benchmarks with series removed from the question prompts. *series* denotes the accuracy of table 1, evaluated with the series shown to the model, *none* denotes the model accuracy with series removed, and *decl.* denotes the share of questions it declined to answer without series, which we regard as appropriate behavior and exclude from the accuracy calculation; Δ marks *series* $-$ *none*.

	TSAQA				TimeSeriesExam				TemporalBench			
	series	none	decl.	Δ	series	none	decl.	Δ	series	none	decl.	Δ
Large Language Models												
GPT-4o	60.82	54.11	42.5	+6.71	58.85	38.78	0.8	+20.07	40.64	20.89	0.0	+19.75
GPT-4o-mini	52.27	59.21	39.8	−6.94	50.54	39.54	0.0	+11.00	32.15	26.98	0.0	+5.17
Gemma-4-E4B-it	51.27	49.57	47.5	+1.70	49.33	38.32	2.4	+11.01	33.24	21.89	0.0	+11.35
Qwen2.5-7B	49.00	47.25	0.9	+1.75	47.45	39.41	0.0	+8.04	36.78	24.25	0.0	+12.53
Qwen3-1.7B	37.09	47.57	17.6	−10.48	42.76	38.74	0.0	+4.02	35.51	29.70	0.0	+5.81
Mistral-7B	45.55	52.93	33.4	−7.38	40.48	41.80	0.3	−1.32	28.61	28.47	0.0	+0.14
Llama-3.1-8B	46.27	48.87	43.6	−2.60	40.75	39.14	0.0	+1.61	26.84	31.34	0.0	−4.50
Time Series Language Models												
ChatTS-14B	36.82	41.53	3.9	−4.71	55.23	40.00	2.1	+15.23	41.01	41.33	0.1	−0.32
ChatTS-8B	35.91	37.85	38.3	−1.94	48.39	38.36	2.1	+10.03	28.88	27.34	5.5	+1.54
TimeOmni-1-7B	46.09	41.61	1.4	+4.48	49.06	41.79	1.2	+7.27	34.88	30.93	0.3	+3.95
Qwen3-4B base	46.00	47.83	10.1	−1.83	47.32	40.08	0.0	+7.24	33.61	33.15	0.0	+0.46

Substantial abstention occurs primarily on TSAQA, where several models decline between one third and one half of the questions after the series is removed. Decline rates are low on the other benchmarks, allowing a more direct comparison. The TSLMs retain substantial accuracy without the series, particularly on TemporalBench, where the absolute differences range from 0.32% to 3.95%. The decreases on TimeSeriesExam are larger, ranging from 7.27% to 15.23%, but considerable accuracy remains. These results indicate that the models answer a large portion of each benchmark without grounding their answers in the input series.

C Full Related Work

C.1 Multimodal Large Language Models

A multimodal language model depends on a representation that makes a new modality accessible to its language backbone. Most MLLMs project continuous features into the token space through a modality adapter (Liu et al., 2023b; Bai et al., 2025; Zhu et al., 2025; Tang et al., 2024; Goel et al., 2025) or attend to them through inserted cross attention layers (Alayrac et al., 2022). Both approaches preserve detailed modality information and underlie much of the progress in vision and audio language models. Other systems discretize the modality into new tokens, including acoustic codes (Défossez et al., 2024) and semantic identifiers (Rajput et al., 2023), or quantize it onto the lexical tokens of a frozen LLM (Yu et al., 2023; Liu et al., 2023a). Discrete tokens support unified input and generation, although unified training over an extended vocabulary can impair the language capabilities of smaller models (Wu et al., 2026). LEXIS uses a compact discrete alphabet to describe the shape of a time series.

C.2 Time Series Language Models

Most TSLMs for understanding encode time series patches as continuous vectors that substitute for text tokens (Jin et al., 2024a; Xie et al., 2025; Wang et al., 2025b; Yu et al., 2026b; Langer et al., 2026; Parker et al., 2026). Other approaches serialize observations as digits (Gruver et al., 2023; Kong et al., 2025; Lin et al., 2026; Guan et al., 2026a) or render the series as plots for vision language models (Liu et al., 2025a; Zhang et al., 2025; Guan et al., 2026b). Controlled analyses question how effectively these interfaces ground the model in the series. Removing the LLM from a reprogramming pipeline can leave predictive accuracy unchanged (Tan et al., 2024); projected representations show limited alignment with language embeddings (Zheng et al., 2025); digit based interfaces fail during numerical modeling of the input before reasoning (Liu et al., 2025a); series presented as text are understood less accurately than the same series rendered as images (Zhou & Yu, 2025); and general LLMs can answer no better with the series than without it (Merrill et al., 2024).

Discrete representations of time series predate LLMs. Symbolic methods such as SAX, 1d SAX, SFA, and SPARTAN encode segments or basis coefficients for indexing and classification (Lin et al., 2007; Malinowski et al., 2013; Schäfer & Högqvist, 2012; Yang & Paparrizos, 2025). Recent generative models tokenize scalar values (Ansari et al., 2024; Wang et al., 2025a; Li et al., 2026), learned codebook entries (Talukder et al., 2024; Parker et al., 2025; Shi et al., 2026; Tao et al., 2026; Zhang et al., 2026a), or symbolic segments spelled over an existing text vocabulary (Chen et al., 2026b), extending earlier symbolic approximation (Elsworth & Güttel, 2020). Factored quantizers either collapse the factors into a single index before the model receives them (Mentzer et al., 2024; Zhang et al., 2026c) or divide the index into sub tokens that encode groups of bits from a learned code (Yu et al., 2024; Zhao et al., 2025; Shi et al., 2026). A closed form frequency quantizer symbolizes each coefficient before byte pair encoding (Sennrich et al., 2016) merges several symbols into one token (Pertsch et al., 2025), while VQShape introduces typed structure outside an LLM (Wen et al., 2024).

Concurrent work induces a vocabulary of local shape units with a separate continuous scale head for a dedicated generator (Wei et al., 2026). Chat-TS further shows that discrete series tokens can outperform projected encoder features for understanding (Quinlan et al., 2026). Existing tokens generally denote complete values, segments, latent codes, or merged groups of coefficients. In LEXIS, each symbol denotes one fixed component of a curve, enabling the representation to be decoded and audited by component.

C.3 Time Series Decomposition and Numeracy

A time series often contains several superposed components, and separating them is a longstanding principle of time series modeling. Classical decomposition separates trend, seasonality, and residual variation (Cleveland et al., 1990); fixed interpretable bases make the coefficients of such components a forecaster's output (Oreshkin et al., 2020); and instance normalization removes and later restores the level and scale of each input window (Kim et al., 2022; Liu et al., 2022). Recent LLM interfaces apply related ideas by decomposing inputs before prompting (Cao et al., 2024), representing scale through separate tokens or prediction heads (Zhang et al., 2026c; Parker et al.,

2026), writing normalization constants or summary statistics into the prompt (Xie et al., 2025; Li et al., 2026), or treating transient events through thresholds, boundaries, captions, and parametric primitives (Masserano et al., 2025; Kimura et al., 2026; Xie et al., 2025; Kim et al., 2026).

These methods typically use decomposition as preprocessing without preserving it throughout the input representation. Scalar bin tokenizers quantize scale together with shape (Ansari et al., 2024; Wang et al., 2025a; Li et al., 2026), while other methods reduce scale to a few global statistics. The variation of scale across a series is rarely expressed as numerals, even though LLM numeracy is digit based and sensitive to tokenization (Wallace et al., 2019; Levy & Geva, 2025; Singh & Strouse, 2024). Alternative number encodings have been proposed for the same reason (Golkar et al., 2023; Zhou et al., 2026). Transient events likewise rarely receive an independent channel: programs that type events add no vocabulary (Kim et al., 2026), and tokenizers that align to events are not paired with a pretrained LLM (Masserano et al., 2025; Kimura et al., 2026). LEXIS preserves the decomposition in the interface itself, allowing each component to be represented and evaluated independently.

C.4 VOCABULARY EXTENSION IN PRETRAINED LANGUAGE MODELS

A discrete interface must determine how new tokens are placed in a pretrained embedding space and how their embeddings evolve during training. In language adaptation, new rows are initialized from related subwords with a calibrated norm (Joshi et al., 2026), from aligned embeddings of a source vocabulary (Minixhofer et al., 2022), or by distillation from the model's own representations (Dobler et al., 2026). Mean initialization can collapse new tokens into a degenerate subspace from which subsequent fine tuning struggles to recover (Chen et al., 2026a). Time series models commonly sample new embeddings from a moment matched distribution, warm them before full training, or leave their initialization unspecified (Tao et al., 2026; Parker et al., 2025; Wang et al., 2025a). Analyses also find that learned series representations can occupy regions of the embedding space separated from the original vocabulary (Zhang et al., 2026b).

COM provides a notable alternative by arranging new embeddings along a curve fitted to the text embedding distribution and regularizing their order during training (Li et al., 2026). Its sensitivity to shuffled token order shows that both the geometry and the assignment of tokens along it influence performance. However, the preservation of this geometry during fine tuning and the ability of the output head to produce every new symbol remain underexamined. LEXIS constructs its embedding geometry directly from the structure of its symbol alphabet, preserves that geometry during alignment, and verifies that every new symbol remains accessible to the output head.