ZEST: ZERO-SHOT END-TO-END TIME SERIES STATE DETECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Time series state detection (TSSD) identifies state boundaries and recurring states in multivariate sequences. Deep self-supervised TSSD methods typically combine target-specific representation learning with clustering, hindering direct transfer to unseen sequences. Our analyses identify persistent conflicts in temporal-proxy supervision and limited transfer gains from enlarging the pretraining corpus under the tested fixed-budget settings. We introduce ZEST, a zero-shot, end-to-end framework that recasts TSSD as hierarchical structured generation, adapting a pretrained vision-language model to predict state structure from global sequence images. To build scalable supervision without manual annotation, we use forecast consistency to guide state-prototype construction from large-scale unlabeled real-world time series. A controllable scheduler then recombines these prototypes into multivariate sequences with exact state labels. To capture state relations across granularities, we construct hierarchical supervision for joint generation of leaf segmentations and state-merge trees. The adapted model provides a default leaf partition and multiple coarser candidates on unseen sequences without target labels, parameter updates, hyperparameter tuning, or clustering. Experiments on six benchmarks demonstrate strong zero-shot leaf performance, while a separate target-label-assisted study shows the utility of selecting among the predicted granularities.

1 INTRODUCTION

Multivariate time series (MTS) capture the evolving dynamics of complex systems, where observations are often associated with a small set of latent states. Time Series State Detection (TSSD) compactly describes these dynamics by assigning a state label to each time point (Wang et al., 2023), thereby tracking both state transitions and state identities over time (Wang et al., 2024). TSSD is therefore directly useful for applications such as activity recognition (Lai et al., 2024), while also providing useful priors for downstream tasks such as forecasting (Yang et al., 2026; Ge et al., 2025).

Existing unsupervised TSSD methods can be broadly categorized into statistical modeling (Hallac et al., 2017), compression-based approaches (Matsubara et al., 2014), and self-supervised learning (Lai et al., 2024; Ermshaus et al., 2025). Deep self-supervised methods, represented by Time2State (Wang et al., 2023) and E2USD (Lai et al., 2024), have established a two-stage TSSD framework (Figure 1(a)). In the *representation learning* stage, fixed-length subsequences are extracted using a sliding window, and their temporal proximity is exploited to construct self-supervised signals for contrastive learning (Chen et al., 2020). In the *state inference* stage, the resulting representations are clustered (Blei & Jordan, 2006), followed by post-processing strategies such as voting to aggregate subsequence-level assignments into point-wise state labels.

Despite their strong performance, these two-stage methods remain difficult to extend to zero-shot, end-to-end state detection on unseen sequences. We examine both the supervision used for transfer and the inference procedure. First, the representation stage builds on temporal proximity, which need not match state identity: nearby windows may straddle a transition, whereas distant ones may share a recurring state. Under a fixed data distribution and sampling rule, more samples do not remove these conflicts. Our experiments find substantial conflict rates and limited transfer gains from corpus-scale pretraining, even with ground-truth state supervision (Section 4). Second, the inference stage keeps the target-fitted clustering, so inference cannot be *end-to-end* even if representations could transfer.

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

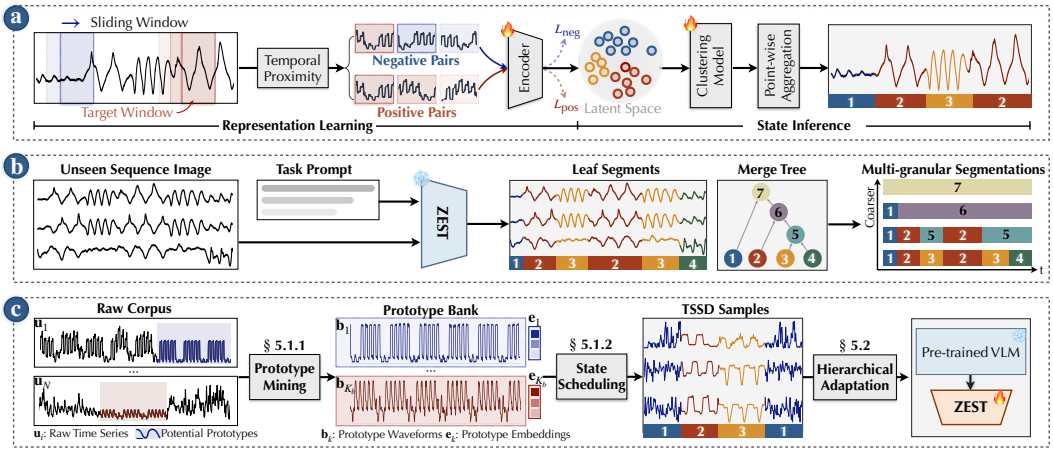


Figure 1: **From two-stage TSSD to zero-shot, end-to-end detection with ZEST.** (a) Temporal-proxy representation learning precedes target-specific clustering and point-wise aggregation. (b) From an unseen sequence image and a task prompt, ZEST predicts leaf segments and a merge tree for multigranular detection without target labels, parameter updates, tuning, or clustering. (c) Prototypes mined from unlabeled real series yield labeled samples for hierarchical adaptation, enabling (b).

Moreover, the appropriate state granularity depends on an application's tolerance to distributional change; a single partition does not expose these alternative interpretations.

We therefore study direct detection on unseen sequences using a model trained once on a source corpus (Figure 1(b)). Realizing this setting poses two challenges. First, scalable supervision: we need precise state labels for diverse temporal patterns. Dense annotation of real time series is difficult to scale, while purely synthetic data may offer limited coverage of real-world dynamics. Second, transferable state modeling: a detector must recognize both local state changes and recurring states across distant intervals. Temporal proximity alone does not reliably encode these relations, since nearby observations may straddle a boundary and distant observations may belong to the same state.

To address these challenges, we introduce **Z**ero-shot **E**nd-to-end **S**tate detection for **T**ime series (ZEST). It combines real-data-grounded supervision with a pretrained vision-language model (VLM) for direct state detection (Figure 1(c)). To build scalable supervision, we develop uncertainty-guided mining and extension to construct reusable state prototypes from unlabeled real series, then recombine them through multivariate state scheduling. This incorporates real waveform patterns while providing state identities and boundaries defined exactly by the scheduler, without manual annotation. To model state structure across the sequence, we adapt the VLM to jointly generate temporal segments and recurring state identities from global time series images. We further construct merge-tree supervision from the generated state definitions and train the model to predict these trees alongside the segmentation. The resulting output provides a default leaf partition and nested coarser candidates without target-specific training or clustering. Experiments on six benchmarks demonstrate strong zero-shot leaf performance. Ablations examine data sources, prototype-construction recipes, and initialization configurations, while a target-label-assisted study evaluates partition selection.

2 Related Work

Time series state detection. Existing unsupervised TSSD methods mainly follow statistical modeling, compression-based approaches, and self-supervised learning. Statistical methods such as HVGH and TICC model state dynamics or observation distributions (Nagano et al., 2019; Hallac et al., 2017), while HDP-Flow also explores cross-population transfer (Tonekaboni et al., 2025). Compression-based methods such as AutoPlait jointly determine state patterns and segmentations using minimum description length (Matsubara et al., 2014). Among self-supervised methods, Time2State and E2USD learn window representations and then cluster them, fitting both stages to each target sequence (Wang et al., 2023; Lai et al., 2024). E2USD introduces False Negative Cancellation Contrastive Learning (FNCC) to mitigate false negatives. CLaP instead uses classification confusion to determine whether

segments share a state (Ermshaus et al., 2025). PromptTSS studies multigranularity segmentation with point-wise training labels and target prompts in its transfer experiments (Chang et al., 2025). This is a target-assisted setting; our comparison evaluates the default detector without target labels.

In contrast, ZEST learns transferable state partitioning and recurrence recognition, enabling detection on unseen sequences without target annotations or target-specific fitting.

Time series foundation models and cross-modal transfer. Pretrained models support time series analysis through two complementary routes: learning from time series corpora and transferring language or vision priors. Along the first route, MOMENT learns general-purpose representations (Goswami et al., 2024), while Chronos-2 focuses on zero-shot forecasting (Ansari et al., 2025). Cross-modal approaches differ in how they connect numerical sequences to pretrained models: they serialize values as text (Gruver et al., 2023), align time series and language representations (Jin et al., 2024; Xie et al., 2024), or convert time series into visual inputs (Chen et al., 2025; He et al., 2026).

ZEST extends this direction to TSSD through real-data-grounded state supervision and joint generation of segments and state merges from a global sequence image.

3 Problem Setup

Time series state detection. Let $\boldsymbol{X} = [\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{L-1}] \in \mathbb{R}^{M \times L}$ be an M-variate time series. TSSD assigns each time point a state ID, yielding a piecewise-constant sequence $\boldsymbol{y} = (y_0, \ldots, y_{L-1})$. State IDs are arbitrary within-sequence labels, not predefined semantic classes; recurring states share an ID across non-adjacent intervals. We represent $\boldsymbol{y}$ by its ordered maximal constant-state segments

$$\mathcal{S} = \big[(a_r, b_r, c_r)\big]_{r=1}^{R}, \tag{1}$$

where c_r labels $[a_r, b_r)$. These intervals partition $[0, L)$ with distinct adjacent IDs. The C latent states are canonically numbered $1, \ldots, C$ by first appearance, with C typically unknown a priori.

Zero-shot end-to-end setting. We consider a detector pretrained on a source corpus that maps an unseen target sequence to its state segmentation:

$$F_\theta : \boldsymbol{X} \longmapsto \widehat{\mathcal{S}}. \tag{2}$$

We call detection *zero-shot* when the detector is applied without target annotations, parameter updates, or target-specific hyperparameter tuning, and *end-to-end* when it directly predicts $\widehat{\mathcal{S}}$ without fitting a separate clustering model or performing other target-specific optimization.

Multigranular extension. Standard TSSD produces a flat segmentation. We extend the output to $(\widehat{\mathcal{S}}, \widehat{\mathcal{T}})$, where $\widehat{\mathcal{T}}$ is a hierarchy whose leaves represent states in $\widehat{\mathcal{S}}$. Successive state merges yield nested, increasingly coarse segmentations, allowing application-specific granularity selection without rerunning the detector. The leaf is the fixed default when no task feedback is available; selecting a coarser level using target labels is a separate assisted setting. Examples appear in Appendix F.2.

4 Temporal-Proxy Conflicts and Transferability

A natural way to extend two-stage TSSD methods is to pretrain their encoders at scale using contrastive objectives. We analyze temporal-proxy conflicts and evaluate transfer as the available corpus grows under a fixed update budget. The tested methods retain target-fitted clustering, so encoder pretraining alone does not yield end-to-end inference.

4.1 Non-vanishing conflicts in temporal supervision

Temporal contrastive learning forms positive pairs from nearby windows and negative pairs from distant ones (Wang et al., 2023). These pairs can conflict with state identity: positives may cross state boundaries, while negatives may share the same state. To quantify these conflicts, let $(S_t)_{t \geq 0}$ be a stationary, irreducible, aperiodic K-state Markov chain with transition matrix P and stationary distribution $\boldsymbol{\pi}$. Suppose $\delta \geq 1$ and $\ell \geq 1$ denote the time lags of positive and negative pairs.

Proposition 1 (Temporal-proxy conflicts)**.** *Under the Markov model above, the conflict probabilities for positive and negative pairs are*

$$\eta_+(\delta) := \Pr(S_t \neq S_{t+\delta}) = 1 - \sum_{k=1}^{K} \pi_k (P^\delta)_{kk}, \quad \eta_-(\ell) := \Pr(S_t = S_{t+\ell}) = \sum_{k=1}^{K} \pi_k (P^\ell)_{kk}. \tag{3}$$

Moreover,

$$\lim_{\ell \to \infty} \eta_-(\ell) = \sum_{k=1}^{K} \pi_k^2 \geq \frac{1}{K}. \tag{4}$$

Proofs and details are provided in Appendix B.1. Increasing negative-pair separation cannot eliminate false negatives: their asymptotic rate is at least $1/K$ and can be substantial when the number of states is small. Under a fixed process and sampling rule, the underlying conflict probability is independent of corpus size. Thus, scaling may improve representations through broader pattern coverage, but it cannot by itself correct the mismatch between temporal supervision and state identity. We next measure conflicts and transfer empirically, without imposing Markov assumptions.

4.2 Corpus size and transfer under a fixed update budget

We pretrain Time2State and E2USD encoders on progressively larger subsets of the corpus in Section 5.1, retaining their architectures, contrastive objectives, and samplers. Each setting uses 5,000 updates, sampling one sequence per update. Larger subsets therefore expand the sampling pool without increasing training exposure.

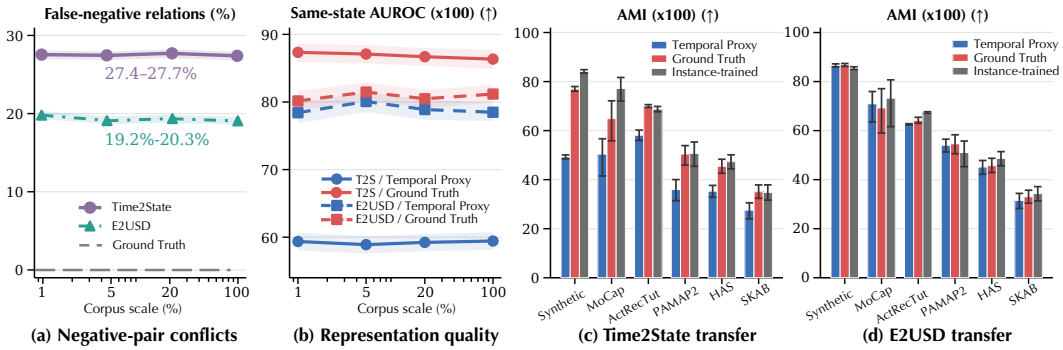


Figure 2: **Fixed-update analysis of Time2State and E2USD.** (a) False-negative rates for Time2State and E2USD (the latter with FNCC), using ground-truth state relations. (b) Same-state AUROC of frozen encoders across pretraining scales. (c–d) Downstream AMI compared with target-specific training. Shading and error bars denote 95% sequence-bootstrap intervals for fixed models.

Figure 2(a) shows substantial false-negative rates for both samplers across corpus sizes, providing empirical evidence of persistent conflicts beyond the Markov model.

For same-state discrimination on held-out sequences, we compare temporal proxies with ground-truth state-relation supervision under otherwise fixed settings. The latter yields higher AUROC, and expanding the sampling pool does not substantially close this gap (Figure 2(b)).

For downstream detection, we replace each encoder with its frozen pretrained counterpart. Temporal pretraining generally underperforms target-specific training in AMI; even state-relation supervision does not consistently outperform this baseline (Figure 2(c–d)). Thus, improving pairwise supervision does not consistently improve downstream transfer in these configurations. This motivates evaluating direct state-structure prediction as an alternative; the experiments do not test scaling with proportionally increased training or establish that a VLM is necessary.

5 Method

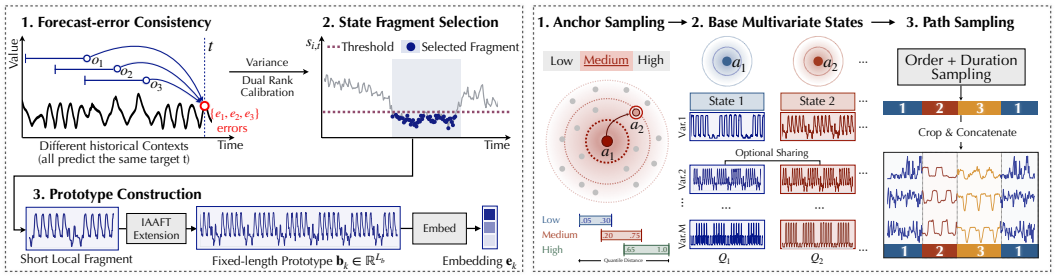
ZEST learns state detection through task adaptation of a pretrained VLM. A global image presents distant occurrences in a shared visual context, while structured generation represents boundaries

and within-sequence state identities in one output. The framework combines two components: a real-data-grounded generator producing sequences with exact state annotations, and VLM adaptation via joint segmentation and hierarchy prediction.

The generator supplies state-labeled sequences without manual annotation (Section 5.1). State definitions also provide merge-tree targets for adaptation (Section 5.2); successive merges in one output yield coarser partitions without target fitting.

5.1 Real-data-grounded state-detection corpus

Our generator maps a raw univariate collection $\mathcal{D}_{\text{raw}}$ to labeled multivariate TSSD samples $(\boldsymbol{X}, \boldsymbol{y})$ (Figure 3). Real series need not exhibit clear state structure, and arbitrary fragments may lack stable dynamics suitable for use as prototypes. Using forecast consistency as a stability cue, we construct reusable state prototypes through uncertainty-guided mining and extension. Then, a state scheduler uses these prototypes and their morphological relations to control state differences, durations, and recurrence, producing multivariate sequences with known state identities and boundaries.



(a) State Prototype Mining (b) Multivariate State Scheduling

Figure 3: **Real-data-grounded corpus construction.** (a) Cross-context forecast consistency guides fragment selection, followed by prototype extension and embedding. (b) Contrast-controlled prototype sampling defines multivariate states; path sampling schedules them to generate labeled sequences.

5.1.1 State prototype mining and extension

We introduce *Uncertainty-Guided Regime Mining and Extension* (URME) to convert unlabeled univariate series $\mathcal{D}_{\text{raw}} = \{\boldsymbol{u}_i\}_{i=1}^{N}$[1] into a library of K_b waveform prototypes $\mathcal{B} = \{\boldsymbol{b}_k\}_{k=1}^{K_b}$ of length L_b and their D-dimensional embeddings $\mathcal{E} = \{\boldsymbol{e}_k\}_{k=1}^{K_b}$ (Figure 3(a)). We first apply per-series z-score normalization to obtain $\widetilde{\boldsymbol{u}}_i$, removing global level and scale without smoothing or detrending.

Forecast consistency and uncertainty scoring. We use forecast consistency across historical contexts as a cue for selecting reusable waveform fragments. For predictions targeting the same time point, dispersion in their errors reflects disagreement across contexts. Under a Gaussian surrogate (Chen et al., 2024; Fu et al., 2025) with variance σ^2, differential entropy is

$$\mathcal{H}_{\text{G}}(\sigma^2) = \frac{1}{2}\log\left(2\pi \mathrm{e}\,\sigma^2\right), \qquad \sigma^2 > 0. \tag{5}$$

Its monotonic dependence on variance motivates using error variance directly to rank this disagreement. This provides an entropy-based interpretation of the score without requiring the actual errors to be Gaussian. We compute the score using Chronos-2 (Ansari et al., 2025) forecasts from overlapping historical contexts. For normalized series $\widetilde{\boldsymbol{u}}_i$, forecast origin o, and horizon $h \in \{1, \ldots, H\}$, let $\widehat{u}_{i,o,h}$ denote the prediction for $t = o + h$. We pool signed errors targeting the same time and compute their variance:

$$e_{i,o,h} = \widetilde{u}_{i,t} - \widehat{u}_{i,o,h}, \qquad \mathcal{R}_{i,t} = \{e_{i,o,h} : o + h = t\}, \qquad v_{i,t} = \mathrm{Var}(\mathcal{R}_{i,t}). \tag{6}$$

For fixed t, the observation is constant, so error variance equals forecast variance across contexts. We use low disagreement as a prototype-mining heuristic, rather than a guarantee of forecast accuracy or state purity.

[1] We use the BLAST corpus (Shao et al., 2025).

Cross-source calibration and fragment mining. Disagreement scores can vary across heterogeneous sources, making a single raw threshold difficult to apply. We therefore assess $v_{i,t}$ through its empirical percentile ranks $g_{i,t}$ across the corpus and $r_{i,t}$ within its source series, combining them with thresholds τ_g, τ_r:

$$s_{i,t} = \max\left(\frac{g_{i,t}}{\tau_g}, \frac{r_{i,t}}{\tau_r}\right). \tag{7}$$

The corpus rank provides a common reference, while the within-series rank favors relatively consistent regions of each source. Before smoothing, $s_{i,t} \leq 1$ requires both rank thresholds to be satisfied. After median filtering, points with scores at most 1 seed fragments, which grow through neighboring points under a relaxed threshold. Examples appear in Appendix F.3.

Prototype extension and morphology embeddings. For reuse in state scheduling, we extend variable-length fragments to L_b using length-adapted iterative amplitude-adjusted Fourier transform (IAAFT) (Schreiber & Schmitz, 1996), approximately preserving their marginal distributions and spectra. We obtain $\boldsymbol{e}_k$ by averaging MOMENT-1-large (Goswami et al., 2024) embeddings over each prototype's overlapping windows, followed by unit normalization, for morphology-aware selection. Details appear in Appendix C.1.

5.1.2 Multivariate state scheduling

The scheduler combines the mined waveform prototypes into states, arranges their occurrences and durations, and generates a complete labeled sequence $(\boldsymbol{X}, \boldsymbol{y})$ (Figure 3(b)). We additionally introduce semantic augmentation to cover state variations beyond basic prototype recombination.

Base state construction. Each base state c is defined by M prototypes, $\boldsymbol{Q}_c = (\boldsymbol{b}_{k_{c,1}}, \ldots, \boldsymbol{b}_{k_{c,M}})$, one per variable. We first select C anchor prototypes, one for each state. To cover both subtle and pronounced state differences, we use three contrast levels—low, medium, and high—with different distance ranges. Let $d(k, k') = \|\boldsymbol{e}_k - \boldsymbol{e}_{k'}\|_2$ denote prototype distance, and let $[d_{\min}, d_{\max}]$ be the range for the selected contrast. Let a_i denote the library index of the ith anchor prototype. We draw a_1 uniformly from the library and each subsequent anchor from

$$\mathcal{A}_i = \{k \notin \{a_1, \ldots, a_{i-1}\} : d_{\min} \leq d(k, a_j) \leq d_{\max},\ \forall j < i\}, \qquad a_i \sim \mathrm{Unif}(\mathcal{A}_i), \tag{8}$$

where k indexes the prototype library. This tests each candidate against every previously selected anchor. We then construct $\boldsymbol{Q}_c$ by sampling prototypes from its anchor's embedding neighborhood, and allow some variables to reuse prototypes from earlier states. Sampling settings, combination checks, and fallback rules appear in Appendix C.2.1 and C.3.2.

State path sampling. With the states defined, we specify when they appear, how long they last, and how they recur. We assign each state one occurrence, randomly allocate the remaining occurrences, and order them with no adjacent repeats. Bounded random durations yield a path of R segments,

$$\boldsymbol{\omega} = \big[(c_1, \ell_1), \ldots, (c_R, \ell_R)\big], \qquad \sum_{r=1}^{R} \ell_r = L, \tag{9}$$

where c_r and ℓ_r specify segment r's state and length. A state can recur at non-adjacent positions with different durations. Cropping and concatenating segments from the corresponding state streams produces $\boldsymbol{X} \in \mathbb{R}^{M \times L}$; cumulative durations and state identities determine boundaries and exact labels $\boldsymbol{y} \in \{1, \ldots, C\}^L$ (Appendix C.2.2).

Semantic augmentation. Predefined sampling distributions can introduce generator-specific statistical patterns that need not hold in real data. We use semantic augmentation to vary state differences, recurrence, and within-state observations in the generated sequences.

At the *state level*, shared prototypes receive distinct mean, amplitude, noise, and drift parameters, extending state differences beyond prototype changes. For state c and variable m, we collect these parameters into a statistical descriptor $\boldsymbol{\phi}_{c,m}$. Stochastic processes supplement mean- and amplitude-based transitions. At the *temporal level*, long-gap recurrence, interleaved background states, and contrasting durations vary when and for how long the same state appears. At the *observation level*, sparse-variable differences and mild label-preserving perturbations cover boundaries visible in only a few variables and variation within a state.

5.2 VLM ADAPTATION AS HIERARCHICAL STRUCTURED GENERATION

We render generated sequences as global images and construct joint prediction targets: leaf segmentations from state labels and merge trees from the scheduler's state definitions.

Global visual input. We plot variables in vertically stacked panels sharing a time axis, forming a global image $\boldsymbol{I}$ of the complete sequence. This presents cross-variable changes and recurrence across non-adjacent intervals within a single input. The image contains no label cues. Rendering and visual preprocessing set the temporal detail available to the model (Appendix E.2).

Constructing hierarchical supervision. Leaf labels define the finest partition but do not specify which states should merge at coarser granularities. We derive state similarity from the scheduler's prototype assignments and persistent statistical properties. For state c and variable m, let $\boldsymbol{\phi}_{c,m} \in \mathbb{R}^4$ encode log-amplitude, mean shift, noise scale, and drift. Combining this descriptor with $\boldsymbol{e}_{k_{c,m}}$ gives

$$\boldsymbol{r}_c = \frac{1}{\sqrt{M}} \operatorname{concat}_{m=1}^{M} \left[\boldsymbol{e}_{k_{c,m}}, \boldsymbol{\phi}_{c,m} \right]. \tag{10}$$

Using Euclidean distances between these descriptors, we apply average linkage offline to construct a merge tree $\mathcal{T}$ with $C-1$ merges. This defines a consistent training target for nested coarsenings under the chosen state descriptors. Each leaf represents a state rather than a temporal segment; all occurrences of the same state share a single leaf. Occurrence durations and crop positions are excluded from the descriptor, as are label-preserving perturbations. Thus different appearances of a recurring state retain one representation, while persistent differences in waveform morphology and statistical properties determine the training-time merge order.

Joint adaptation and inference. We pair each global image with a leaf segmentation $\mathcal{S}$ and its merge tree $\mathcal{T}$, jointly serialized as a structured target. Segments are encoded as start–end–state triples: a path $1 \to 2 \to 1$ contains three temporal segments but only two state identities. Reusing ID 1 connects the separated occurrences in the output. Each subsequent merge names two active state IDs and a new parent ID. Applying it relabels every occurrence of both child states, so granularity changes consistently across the whole sequence. The prompt supplies the sequence length and variable count to relate these coordinates to the image. We adapt Qwen3.5-2B (Qwen Team, 2026) with LoRA (Hu et al., 2022), using an autoregressive loss to learn both structures.

For unseen sequences, the model directly predicts the segmentation and merge tree without target-specific retraining or clustering (Figure 1(b)). Successive merges along the predicted tree yield segmentations at different granularities from the same output. Input and output specifications, hierarchy construction, training settings, and structural validation are detailed in Appendix D.

6 EXPERIMENTS

6.1 EXPERIMENTAL SETUP

Datasets and baselines. We evaluate on six public benchmarks used in prior work (Lai et al., 2024; Ermshaus et al., 2025): Synthetic, MoCap, ActRecTut, PAMAP2, HAS, and SKAB. All six benchmark sources are excluded from ZEST's training corpus construction. We compare against eleven baselines: seven specialized TSSD methods (HVGH (Nagano et al., 2019), TICC (Hallac et al., 2017), AutoPlait (Matsubara et al., 2014), ClaSP-KMeans (Ermshaus et al., 2023), Time2State (Wang et al., 2023), E2USD (Lai et al., 2024), and CLaP (Ermshaus et al., 2025)) and four VLM/LLM baselines (Qwen3.5-2B (Qwen Team, 2026), ChatTS-8B (Xie et al., 2024), TimeOmni-1-4B (Guan et al., 2026), and TimeMaster-SFT-3B (Zhang et al., 2025)). The latter are evaluated without target-specific training. This comparison evaluates the resulting detectors under their respective training regimes; it does not match source-task supervision or training cost. Dataset statistics and baseline protocols are provided in Appendix E.

Implementation and evaluation. We train for one epoch on the training split of a corpus containing one million generated sequences. We report adjusted Rand index (ARI) and normalized mutual information (NMI), averaged over sequences within each dataset. ZEST-leaf evaluates the finest

Table 1: **State detection on six benchmarks** ($\times 100$)**.** Bold marks the best non-oracle score. Oracle uses test labels to select a level and is diagnostic.

	Method	Synthetic		MoCap		ActRecTut		PAMAP2		HAS		SKAB	
		ARI↑	NMI↑	ARI↑	NMI↑	ARI↑	NMI↑	ARI↑	NMI↑	ARI↑	NMI↑	ARI↑	NMI↑
Specialized TSSD baselines	HVGH	8.09	16.06	5.00	15.23	8.81	20.88	0.32	3.74	6.72	16.97	10.92	13.92
	TICC^K	62.42	74.89	72.18	75.24	78.39	74.66	30.08	59.55	27.52	29.62	0.09	3.41
	AutoPlait	7.13	13.07	80.57	**82.89**	5.86	14.18	21.20	21.21	46.82	53.27	37.84	39.71
	ClaSP-KMeans^K	29.50	44.80	54.50	67.63	28.25	23.09	17.00	58.30	32.73	39.12	10.19	15.57
	Time2State	81.76	84.07	75.29	75.84	76.70	74.07	31.35	59.05	32.64	45.36	13.90	21.15
	E2USD	88.43	80.25	78.96	78.12	79.09	74.73	33.45	61.43	34.79	46.68	4.57	4.47
	CLaP	61.34	78.05	58.03	67.98	41.40	53.46	22.54	51.63	29.46	35.44	29.46	35.49
LLM/VLM baselines	Qwen3.5	19.27	36.92	33.01	53.53	29.00	46.32	21.75	51.13	39.08	49.94	27.84	36.02
	ChatTS	1.93	4.41	0.00	0.00	0.00	0.00	0.00	0.00	12.21	13.35	0.11	0.41
	TimeOmni-1	12.71	35.67	14.63	34.89	20.32	27.64	11.93	34.71	23.60	33.57	0.81	3.10
	TimeMaster	12.15	20.65	32.09	51.84	20.36	26.94	21.34	48.57	37.57	48.11	24.29	36.55
Ours	**ZEST-leaf**	**96.84**	**94.22**	**81.33**	81.08	**80.46**	**81.32**	**40.49**	**64.52**	**47.29**	**56.75**	**39.96**	**46.14**
	Δ vs. best baseline	+8.41	+10.15	+0.76	-1.81	+1.37	+6.59	+7.04	+3.09	+0.47	+3.48	+2.12	+6.43
	ZEST-*oracle*	96.87	94.27	82.81	82.52	87.11	82.49	41.74	64.56	72.87	75.15	55.83	56.59

K: true state count supplied. ChatTS zeros on MoCap, ActRecTut, and PAMAP2 reflect single-state predictions.

predicted partition without target labels or a given state count. For diagnosis, ZEST-oracle selects the hierarchy level maximizing test-label ARI and reports both metrics at that level. It provides an ARI upper bound within the predicted hierarchy, not a deployable selection rule. Training configurations and evaluation protocols are detailed in Appendix E.

6.2 Main results

Overall performance. ZEST-leaf ranks first in eleven of the twelve comparisons in Table 1. Its advantage is particularly pronounced on Synthetic, where controlled state structure allows clear state partitions to be effectively recovered. On the five real-world datasets, it attains the highest ARI in each case and the highest NMI in four. These are observed single-run rankings; the margins vary across datasets and do not establish training-seed stability.

The role of task adaptation. ZEST-leaf outperforms the unadapted Qwen3.5-2B and all three time-series LLM/VLM baselines on both metrics across all datasets. The same-backbone comparison demonstrates the benefit of the complete task-adaptation recipe; it does not separate corpus supervision from the output design or identify a specific pretrained capability. For ChatTS, approximately 84% of inputs fall back to single-state predictions, so its scores also reflect format-following limitations.

Candidate partitions and oracle diagnosis. The oracle–leaf gap varies substantially across datasets: the finest partition need not best match the reference labels, which reflect task-specific distinctions. The gap measures headroom within this candidate set, rather than an additional zero-shot gain or a comparison against alternative trees. We next examine selection with target-label feedback, keeping the predictions fixed.

6.3 Target-label-assisted granularity selection

Different tasks can assign different state distinctions to the same observations. We use labeled timestamps as a proxy for these preferences and evaluate selection within the fixed predicted hierarchy. This study measures adaptation of an existing output, separately from the zero-shot leaf comparison. For each sequence, we select the predicted hierarchy level that maximizes adjusted mutual information (AMI) on the labeled timestamps. Label budgets range from 0 to 2%, with the leaf partition used when no labels are available. The candidate partitions remain fixed across budgets; feedback changes only which level is selected. We score all budgets on a common set of timestamps that excludes the largest annotation set, keeping selection labels separate from evaluation (Appendix E.4).

Most of the macro-average improvement is attained at a 0.5% label budget (Figure 4), with gains concentrated on HAS, SKAB, and MoCap. Other datasets change little, and PAMAP2 declines

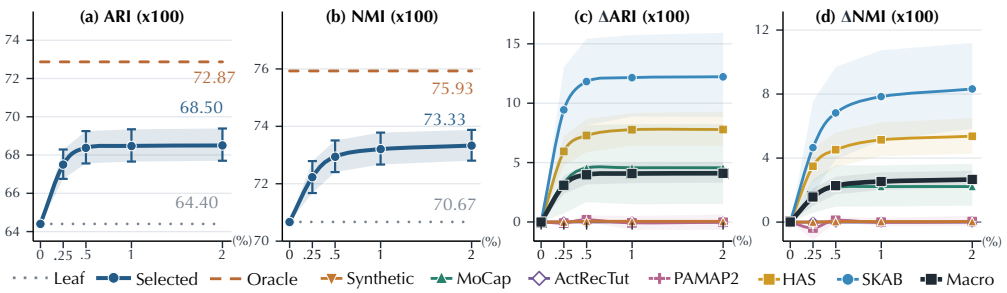


Figure 4: **Target-label-assisted granularity selection.** (a–b) Dataset-macro ARI and NMI. (c–d) Gains over ZEST-leaf for each dataset and their equal-weight mean. Bands and error bars show 95% paired-bootstrap gain intervals. All budgets share evaluation timestamps.

slightly at the lowest nonzero budget. Thus feedback can improve partition selection, but its benefit depends on the task and the observed labels.

6.4 Ablation Study

Experimental design. We examine data source, prototype construction, and pretrained initialization. We cross two sources—real BLAST observations (B) and synthetic KernelSynth sequences (KS) (Ansari et al., 2024)—with URME (U) and RandomCrop (RC), which directly samples raw fragments. B+U is the complete ZEST pipeline. We also train the same architecture from random initialization on B+U, updating all parameters. Figure 5 reports equally weighted dataset averages; settings and per-dataset results appear in Appendix E.5.

Data source and prototype-construction recipe. B+U performs best, while KernelSynth outperforms BLAST with RandomCrop. The source advantage therefore depends on the construction recipe. Here URME combines forecast-guided selection with IAAFT extension, whereas RandomCrop uses random fragments and interpolation when needed. The improvement on BLAST supports this combined recipe; it does not attribute the gain to forecast-guided selection alone.

Initialization and adaptation configuration. The pretrained model with LoRA outperforms the randomly initialized architecture trained end to end under the evaluated recipes. This supports using the pretrained configuration in this system; initialization and parameter-update strategy change together, so the comparison does not isolate visual or world knowledge.

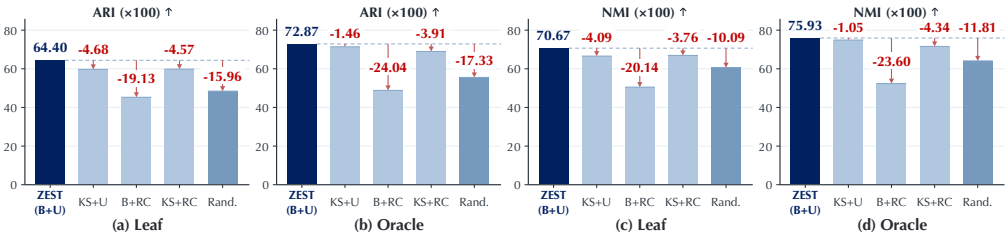


Figure 5: **Ablation study.** (a–b) Leaf and oracle ARI; (c–d) leaf and oracle NMI, averaged equally across six benchmarks. Rand. uses random initialization and full-parameter training.

7 Conclusion

We introduced ZEST, a zero-shot, end-to-end framework for detecting state boundaries and recurring states in unseen time series. Our analyses reveal persistent temporal-proxy conflicts and limited transfer gains under the tested training settings. Motivated by these findings, ZEST combines uncertainty-guided prototype mining with state scheduling to build scalable supervision from unlabeled real data. We use this supervision to adapt a pretrained VLM to jointly generate state segmentations and merge trees. Across six benchmarks, ZEST achieves strong leaf performance without target-specific training or clustering. A separate label-assisted study shows how task feedback helps select suitable granularities from the same output.

AI USE STATEMENT

We used generative AI tools to assist with manuscript drafting and revision, figure design, code organization, and discussions of theoretical assumptions. The authors retain responsibility for the experimental design, verification of results, and all claims and code in the final submission.

ETHICS STATEMENT

Our experiments use existing time series datasets, including human-activity and industrial data. Human-activity data may reveal personal behavior, so practical applications should consider data privacy and avoid overinterpreting the predicted states.

REPRODUCIBILITY STATEMENT

The proof and its scope are given in Appendix B.1. Data construction and model implementation are described in Appendices C and D. Experimental settings and evaluation procedures are provided in Appendix E.

REFERENCES

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. *arXiv preprint arXiv:2403.07815*, 2024. URL https://arxiv.org/abs/2403.07815.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting. *arXiv preprint arXiv:2510.15821*, 2025. URL https://arxiv.org/abs/2510.15821.

David M Blei and Michael I Jordan. Variational inference for dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–143, 2006.

Ching Chang, Ming-Chih Lo, Wen-Chih Peng, and Tien-Fu Chen. PromptTSS: A prompting-based approach for interactive multi-granularity time series segmentation. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, 2025. doi: 10.1145/3746252.3761142.

Mouxiang Chen, Lefei Shen, Han Fu, Zhuo Li, Jianling Sun, and Chenghao Liu. Calibration of time-series forecasting: Detecting and adapting context-driven distribution shift. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 341–352, 2024.

Mouxiang Chen, Lefei Shen, Zhuo Li, Xiaoyun Joy Wang, Jianling Sun, and Chenghao Liu. VisionTS: Visual masked autoencoders are free-lunch zero-shot time series forecasters. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 8979–9007, 2025. URL https://proceedings.mlr.press/v267/chen25be.html.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PmLR, 2020.

Arik Ermshaus, Patrick Schäfer, and Ulf Leser. ClaSP: Parameter-free time series segmentation. *Data Mining and Knowledge Discovery*, 37:1262–1300, 2023. doi: 10.1007/s10618-023-00923-x. URL https://link.springer.com/article/10.1007/s10618-023-00923-x.

Arik Ermshaus, Patrick Schäfer, and Ulf Leser. CLaP – state detection from time series. *arXiv preprint arXiv:2504.01783*, 2025. URL https://arxiv.org/abs/2504.01783.

Yisong Fu, Zezhi Shao, Chengqing Yu, Yujie Li, Zhulin An, Qi (Cheems) Wang, Yongjun Xu, and Fei Wang. Selective learning for deep time series forecasting. In *Advances in Neural Information Processing Systems*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/8cf54ff53f44835b9bdab2c546a1ca6d-Abstract-Conference.html.

Yunfeng Ge, Ming Jin, Yiji Zhao, Hongyan Li, Bo Du, Chang Xu, and Shirui Pan. Eventtsf: Event-aware non-stationary time series forecasting. *arXiv preprint arXiv:2508.13434*, 2025.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A family of open time-series foundation models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 16115–16152, 2024. URL https://proceedings.mlr.press/v235/goswami24a.html.

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, 2023. URL https://arxiv.org/abs/2310.07820.

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*, 2026. URL https://arxiv.org/abs/2509.24803.

David Hallac, Sagar Vare, Stephen Boyd, and Jure Leskovec. Toeplitz inverse covariance-based clustering of multivariate time series data. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 215–223, 2017. doi: 10.1145/3097983.3098060.

Zelin He, Sarah Alnegheimish, and Matthew Reimherr. Harnessing vision-language models for time series anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 21690–21698, 2026. doi: 10.1609/aaai.v40i26.39319.

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.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985. doi: 10.1007/BF01908075.

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*, 2024. URL https://openreview.net/forum?id=Unb5CVPtae.

Zhichen Lai, Huan Li, Dalin Zhang, Yan Zhao, Weizhu Qian, and Christian S. Jensen. E2USD: Efficient-yet-effective unsupervised state detection for multivariate time series. In *Proceedings of the ACM Web Conference 2024*, pp. 3010–3021, 2024. doi: 10.1145/3589334.3645593.

Yasuko Matsubara, Yasushi Sakurai, and Christos Faloutsos. AutoPlait: Automatic mining of co-evolving time sequences. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pp. 193–204, 2014. doi: 10.1145/2588555.2588556.

Masatoshi Nagano, Tomoaki Nakamura, Takayuki Nagai, Daichi Mochihashi, Ichiro Kobayashi, and Wataru Takano. Hvgh: unsupervised segmentation for high-dimensional time series using deep neural compression and statistical generative model. *Frontiers in Robotics and AI*, 6:115, 2019.

Qwen Team. Qwen3.5: Towards native multimodal agents. Official Qwen technical blog, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Thomas Schreiber and Andreas Schmitz. Improved surrogate data for nonlinearity tests. *Physical Review Letters*, 77(4):635–638, 1996. doi: 10.1103/PhysRevLett.77.635.

Zezhi Shao, Yujie Li, Fei Wang, Chengqing Yu, Yisong Fu, Tangwen Qian, Bin Xu, Boyu Diao, Yongjun Xu, and Xueqi Cheng. BLAST: Balanced sampling time series corpus for universal forecasting models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2502–2513, 2025. doi: 10.1145/3711896.3736860.

Sana Tonekaboni, Tina Behrouzi, Addison Weatherhead, Emily Fox, David Blei, and Anna Goldenberg. HDP-Flow: Generalizable bayesian nonparametric model for time series state discovery. In *Proceedings of the Forty-First Conference on Uncertainty in Artificial Intelligence*, volume 286 of *Proceedings of Machine Learning Research*, 2025. URL https://proceedings.mlr.press/v286/tonekaboni25a.html.

Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11:2837–2854, 2010. URL https://jmlr.org/papers/v11/vinh10a.html.

Chengyu Wang, Kui Wu, Tongqing Zhou, and Zhiping Cai. Time2State: An unsupervised framework for inferring the latent states in time series data. *Proceedings of the ACM on Management of Data*, 1(1):1–18, 2023. doi: 10.1145/3588697.

Chengyu Wang, Xionglve Li, Tongqing Zhou, and Zhiping Cai. Unsupervised time series segmentation: A survey on recent advances. *Computers, Materials & Continua*, 80(2):2657–2673, 2024. doi: 10.32604/cmc.2024.054061.

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. *arXiv preprint arXiv:2412.03104*, 2024. URL https://arxiv.org/abs/2412.03104.

Jie Yang, Yifan Hu, Yuante Li, Kexin Zhang, Kaize Ding, and Philip S. Yu. From observations to states: Latent time series forecasting. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=s49fw3BVU0.

Junru Zhang, Lang Feng, Xu Guo, Yuhan Wu, Yabo Dong, and Duanqing Xu. TimeMaster: Training time-series multimodal LLMs to reason via reinforcement learning. *arXiv preprint arXiv:2506.13705*, 2025. URL https://arxiv.org/abs/2506.13705.

A NOTATION

Table 2 summarizes the principal symbols used in the paper.

Table 2: Principal notation. State and tree-node IDs are one-based; temporal intervals remain zero-based and half-open.

Symbol	Meaning
$\boldsymbol{X} \in \mathbb{R}^{M \times L}$	Input with M variables and length L
$\boldsymbol{y}, C$	Point-wise labels $y_t \in \{1, \dots, C\}$ and number of states
$\mathcal{S}$	Segmentation represented by segment boundaries and state IDs
$\mathcal{T}$	State-merge tree
$(S_t), P, \boldsymbol{\pi}$	Markov state process, transition matrix, and stationary distribution
$\delta, \ell, \eta_+, \eta_-$	Positive/negative lags and their population conflict rates
$\mathcal{D}_{\text{raw}} = \{\boldsymbol{u}_i\}_{i=1}^{N}$	Raw univariate time-series corpus
$\mathcal{B} = \{\boldsymbol{b}_k\}_{k=1}^{K_b}$	Length-L_b waveform prototype library
$\mathcal{E} = \{\boldsymbol{e}_k\}_{k=1}^{K_b}$	Aligned D-dimensional prototype embeddings
$\boldsymbol{Q}_c$	Multivariate prototype tuple of leaf state c
$a_i, \mathcal{A}_i$	Anchor prototype index and eligible set at sampling step i
$d(k, k')$	Distance between prototype embeddings
$\boldsymbol{\omega} = \big[(c_r, \ell_r)\big]_{r=1}^{R}$	State path and segment durations
$\boldsymbol{\phi}_{c,m} \in \mathbb{R}^4$	Persistent statistical descriptor: log-amplitude, mean shift, noise scale, and drift (Equation 21)
$\boldsymbol{r}_c$	State representation for hierarchical supervision
$\boldsymbol{I}$	Global time-series image

B TEMPORAL-PROXY CONFLICTS AND SCALING EXPERIMENTS

We provide the proof behind Section 4.1, then describe how conflicts and transfer performance are measured in Section 4.2.

B.1 PROOF AND SCOPE OF TEMPORAL-PROXY CONFLICTS

Proof of Proposition 1. By stationarity and the Markov property, for any state k and lag d,

$$\Pr(S_t = k, S_{t+d} = k) = \Pr(S_t = k) \Pr(S_{t+d} = k \mid S_t = k) = \pi_k (P^d)_{kk}. \tag{11}$$

Summing over states gives $\Pr(S_t = S_{t+d}) = \sum_k \pi_k (P^d)_{kk}$. Taking the complement at $d = \delta$ gives the positive-pair conflict probability; setting $d = \ell$ gives the negative-pair conflict probability.

For the large-lag limit, we assume the finite chain is irreducible and aperiodic: all states communicate, and returns are not restricted to a fixed cycle. Its transition probabilities then converge to the stationary distribution, $P^\ell \to \mathbf{1}\boldsymbol{\pi}^\top$, giving

$$\lim_{\ell \to \infty} \eta_-(\ell) = \sum_{k=1}^{K} \pi_k^2 \geq \frac{1}{K}. \tag{12}$$

The inequality follows from $(\sum_k \pi_k)^2 \leq K \sum_k \pi_k^2$ and $\sum_k \pi_k = 1$. This lower bound applies to the limit, not to every finite lag.

Scope. The model explains a source of conflicting supervision; it is not assumed to describe every TSSD sequence. The same limit holds for a stationary finite-state process whenever $\Pr(S_t = i, S_{t+\ell} = j) \to \pi_i \pi_j$. With a fixed data distribution and pair-selection distribution, increasing the corpus size does not change the population conflict probability. Similarity-based filtering can change which pairs are selected, so the bound should not be applied directly to filtered negatives; their rates are measured below. The proposition establishes persistent supervision conflicts, not a ceiling on representation or detection performance.

B.2 EMPIRICAL CONFLICT-RATE ESTIMATION

We use the corpus labels to check whether sampled pairs agree with state identity. Each window contains 256 time points. We count the points belonging to each state and assign the window to the most frequent state. A positive pair conflicts with the labels if its windows belong to different states; a negative pair conflicts if they belong to the same state. For window pairs (u_r^+, v_r^+) and (u_r^-, v_r^-), the rates are

$$\begin{aligned} \widehat{\eta}_+ &= \frac{1}{n_+} \sum_{r=1}^{n_+} \mathbb{I}[s(u_r^+) \neq s(v_r^+)], \\ \widehat{\eta}_- &= \frac{1}{n_-} \sum_{r=1}^{n_-} \mathbb{I}[s(u_r^-) = s(v_r^-)]. \end{aligned} \tag{13}$$

where $s(u)$ is the assigned window state and n_+, n_- are the numbers of evaluated pairs. Figure 2(a) reports $\widehat{\eta}_-$, the fraction of negative pairs that share a state.

Time2State samples groups of neighboring windows and treats different groups as negatives. We measure conflicts across the constituent window pairs, since its group-average embeddings have no single reference state. E2USD applies FNCC to the same candidates, retaining the half with lower trend-and-seasonal similarity. Its conflicts are measured after filtering, using the final checkpoint at each corpus scale. The state-relation comparison uses known labels to form same-state positives and different-state negatives, giving zero conflicts by construction.

At each scale, we randomly sample 5,000 sequences. Both comparisons use the same subset supporting at least two window states, leaving 4,486–4,519 sequences. We divide the total number of conflicting pairs by the total number of evaluated pairs.

B.3 SCALING AND TRANSFER EVALUATION

Data and supervision controls. We shuffle the one-million-sequence corpus and reserve 1,000 sequences for evaluation. Training subsets contain 1, 5, 20, or 100% of the remaining data: 9,990, 49,950, 199,800, and 999,000 sequences. Each smaller subset is included in the larger ones. We standardize each variable separately and add zero-valued channels to obtain 10 input channels. Windows contain 256 time points; evaluation windows have stride 50.

Each setting uses one seed-123 run with 5,000 Adam updates and learning rate 3×10^{-3}. Each update samples one sequence and 10 groups of four windows. Thus, larger corpora increase the available data while the update budget stays fixed. Within each method, both supervision settings use the same architecture and optimizer; state labels replace temporal proximity when forming groups and negative pairs. Figure 2 reports these 5,000-update runs. Each run can therefore visit at most 5,000 distinct sequences; increasing the available corpus changes sampling diversity, not the update count. The experiment does not assess training to convergence at every scale.

Same-state discrimination. We test whether windows from the same state have more similar embeddings than windows from different states, using cosine similarity as the score and area under the receiver operating characteristic curve (AUROC). The encoder remains frozen. From the first 500 held-out sequences, we retain windows with at least 75% of their time points in one state. For each sequence, we sample 128 same-state and 128 different-state pairs. Sequences unable to provide both types are excluded, leaving 418 sequences and 107,008 pairs in every setting. We compute AUROC over all these pairs, rather than averaging separate sequence scores.

Downstream state detection. We compare frozen encoders pretrained on the full corpus with encoders trained for 20 updates on each target sequence. Both use the same window size, stride, and subsequent processing. A Bayesian Gaussian mixture clusters the window embeddings separately for each target, using at most 20 components, concentration 1,000, and at most 1,000 iterations (seed 123). Each time point receives the most common cluster label among windows covering it. Neither setting uses target labels or the true state count. AMI is evaluated on the original time points and averaged within each benchmark. The six benchmarks contain 404 sequences after excluding one processed ActRecTut duplicate.

Confidence intervals. Shading and error bars in Figure 2 denote 95% bootstrap confidence intervals obtained by resampling sequences, not variation across independent training runs.

C CORPUS CONSTRUCTION AND GENERATION CONTROLS

We describe prototype mining, state/path construction, semantic augmentation, and checks for observable state differences. Numerical settings describe the current recipe; the reported training runs are specified in Appendix E.2.

C.1 UNCERTAINTY-GUIDED REGIME MINING AND EXTENSION

URME mines reusable waveform prototypes from unlabeled series. Forecast-error variance guides fragment selection, extension supplies a common length, and morphology embeddings organize similar and contrasting prototypes.

Normalization and forecast errors. Each BLAST series is normalized before forecasting:

$$\widetilde{u}_{i,t} = (u_{i,t} - \mu_i)/\sigma_i. \tag{14}$$

Statistics exclude missing observations, whose positions remain missing. This removes global level and scale without smoothing or detrending. Chronos-2 produces point forecasts from overlapping historical contexts. Equation 6 computes signed prediction errors, which are pooled by target time as $\mathcal{R}_{i,t}$, giving

$$v_{i,t} = \frac{1}{|\mathcal{R}_{i,t}|} \sum_{e \in \mathcal{R}_{i,t}} (e - \bar{e}_{i,t})^2. \tag{15}$$

Since $\widetilde{u}_{i,t}$ is fixed across these forecasts, $v_{i,t} = \mathrm{Var}(\{\widehat{u}_{i,o,h} : o + h = t\})$ measures cross-context forecast disagreement. Selection uses this variance directly and requires no parametric error-density model. Consistent forecasts may share a bias, so this score is a selection heuristic rather than a test of state purity.

Dual ranks and seed growth. We compute the corpus percentile directly from raw variances using an approximate empirical CDF, and within-series ordinal ranks normalized by the scored-row length. Ties follow temporal order; missing positions receive percentile 1. Their maximum after threshold scaling, Equation 7, combines corpus-wide and within-series selection criteria; it does not make error distributions identical across sources. A width-seven median suppresses isolated score fluctuations. Positions with a filtered score at most 1 seed a fragment, which grows through adjacent positions scoring at most 1.2. Only connected regions containing a seed are retained; gaps of at most 16 points are joined. Extraction removes fragments shorter than 128 points, and prototype construction retains those of at least 512 points.

Table 3: Mining and prototype-construction settings for BLAST.

Setting	Value
Raw sequence / forecast context / horizon	4096 / 1024 / 128 points
Forecast-origin stride / forecast type	8 / point prediction
Scored region	After 1152 points; effective length ≥ 128, missing fraction $\leq 1\%$ excluding terminal padding
Corpus / within-series thresholds	$\tau_g = 0.50$ / $\tau_r = 0.30$
Corpus CDF approximation	Up to 2×10^6 sampled values; 131,072 lookup knots
Extension length / iteration limit	$L_b = 8192$ / 500
Period search / minimum autocorrelation	2 to $\min(\lfloor \ell/2 \rfloor, 512)$ / 0.1
Embedding window / stride	1024 / 256 points
Mining / extension random seed	42 / 42

Length-adapted IAAFT. Internal missing values are linearly interpolated. Each fragment is centered by its median and scaled by $\mathrm{IQR}/1.349$, using standard deviation when the IQR vanishes. If a salient autocorrelation peak is found, phase-wise medians form a periodic template, smoothed by

a circular width-three median for periods of at least five points. We extend its residual; otherwise we extend the entire normalized fragment.

To extend a source of length ℓ, we interpolate its nonzero-frequency Fourier magnitudes onto the target grid:

$$A_{\text{target}}(f) = \text{Interp}(A_{\text{source}})(f)\sqrt{L_b/\ell}. \tag{16}$$

We also sample L_b source values with replacement to define the target marginal distribution. Starting from their random permutation, IAAFT replaces Fourier magnitudes while retaining phases, then assigns the sampled values in amplitude order. It stops at the iteration limit, a spectral-error change below 10^{-7}, or an unchanged rank mapping. A periodic template is repeated with a random phase and added back to the extended residual. The output is restored to the source scale, clipped to its 1st–99th percentiles, and z-score normalized; valid prototypes have finite values and positive variance. This approximately retains marginal and spectral structure while supplying a common reusable length.

Morphology embeddings. MOMENT-1-large encodes overlapping prototype windows. Patch outputs, when present, are averaged within a window, then window embeddings are averaged within a prototype. These vectors are normalized to unit length before scheduler distances and hierarchical supervision are computed. Waveforms and embeddings refer to the same prototypes throughout the pipeline.

C.2 Multivariate state scheduler

The scheduler defines states, samples their paths, and realizes waveforms, with semantic augmentation broadening each stage. The current recipe uses $L = 4096$, $M \in [1, 10]$, $C \in [2, 15]$, and $C \leq R \leq 45$, subject to duration feasibility; single-state samples have $C = R = 1$. At most 8192 prototypes are selected uniformly without replacement using seed 123; their unit-normalized embeddings define the active library.

C.2.1 Base state construction

Anchors and contrast. Let F_d be the empirical distribution of pairwise prototype distances. The low/medium/high ranges in Table 4 span subtle to pronounced state differences. Equation 8 draws the first anchor uniformly, then draws uniformly from unused prototypes satisfying the distance interval with every earlier anchor.

Table 4: Prototype-distance quantiles for the three contrast levels.

Contrast	Probability	Anchor interval	State-combination interval
Low	0.125	$[0.05, 0.30]$	$[0.05, 0.50]$
Medium	0.500	$[0.20, 0.75]$	$[0.05, 0.95]$
High	0.375	$[0.65, 1.00]$	$[0.20, 1.00]$

Multivariate combinations. For each sequence, sample a neighborhood quantile $q_N \sim U[0.05, 0.15]$ and sharing probability $p_S \sim U[0.15, 0.35]$. Each anchor's neighborhood contains prototypes within $F_d^{-1}(q_N)$, with at least eight nearest candidates when the library permits. The first state's variables draw independently from its neighborhood. Later variables copy the corresponding prototype of a uniformly chosen earlier state with probability p_S, or draw from the new neighborhood. This permits partial sharing across states.

After forming $\boldsymbol{Q}_c$, we check distinctness and all pairwise distances:

$$D(c, c') = \frac{1}{M}\sum_{m=1}^{M} d(k_{c,m}, k_{c',m}). \tag{17}$$

The bounds use the same F_d as the anchor intervals. This second check is needed because variable sharing changes the distance between complete states. It applies to base prototype combinations; the statistical and sparse constructions below define their distinctions separately.

C.2.2 STATE PATHS AND SEQUENCE GENERATION

Counts, order, and duration. Every state receives one occurrence; the remaining $R - C$ occurrences follow an equal-probability multinomial. Counts satisfy $\max_c n_c \leq \lceil R/2 \rceil$, allowing an order without adjacent repeats. At each step we choose only states that leave a feasible continuation. When both introduced and unseen states are eligible, the introduced group is selected with probability 0.80, then a state is drawn uniformly within the chosen group.

Ordinary paths uniformly select a feasible segment count and duration style. Starting from each segment's lower bound, log-normal weights allocate the remaining length, respecting upper bounds and integer rounding until the durations sum to L.

Table 5: Ordinary duration styles; σ is the normal-distribution standard deviation underlying the log-normal allocation weights.

Style	Short	Medium	Long	Broad
Length range	$[32, 256]$	$[128, 768]$	$[512, 2048]$	$[32, 2048]$
σ	0.30	0.40	0.50	0.55

Waveforms and exact labels. Each prototype-driven state supplies a transformed stream of length $\ell_{\max} + 2p$, with margin $p = \lceil 0.10\ell_{\max} \rceil$ around its longest occurrence. An occurrence of length ℓ_r is cropped near position p, with a uniform integer offset bounded by $\mathrm{round}(\rho\ell_r)$ and clipped to valid crop positions. Ordinary prototype combinations use $\rho = 0.10$; shared-prototype settings appear below; sparse and duration-only constructions use zero jitter. Cropping retains the native sampling rate. Crops are concatenated along the path; cumulative durations define boundaries. Labels use state IDs $1, \ldots, C$, assigned by first appearance and reused on recurrence.

C.2.3 SEMANTIC AUGMENTATION

Persistent state-level parameters define identities. Temporal augmentation varies their recurrence and durations; mild observation perturbations preserve the assigned identity.

State level: statistical distinctions. For source waveform b, median v, standard deviation s, and linear coordinate $\zeta_t \in [-0.5, 0.5]$, the state stream is

$$g_{c,m,t} = v + a_{c,m}(b_t - v) + \mu_{c,m}s + d_{c,m}s\zeta_t + \sigma^{\mathrm{noise}}_{c,m}s\xi_t, \quad \xi_t \sim \mathcal{N}(0, 1). \tag{18}$$

Here a, μ, σ^{noise}, and d control amplitude, mean, noise, and drift. They remain fixed across occurrences; source-stream noise is drawn once before cropping. Shared-prototype states reuse the same multivariate prototype combination and differ through these persistent parameters.

Table 6: Per-variable state transforms. Activation is the initial sampling probability; inactive variables use $(a, \mu, \sigma^{\mathrm{noise}}, d) = (1, 0, 0, 0)$.

Construction	Activation	a	μ	σ^{noise}	d
Prototype combination	0.35	$[0.80, 1.25]$	$[-0.20, 0.20]$	$[0, 0.05]$	0
Shared prototypes	0.70	$[0.55, 1.80]$	$[-1.25, 1.25]$	$[0, 0.12]$	$[-0.20, 0.20]$

Shared-prototype profiles control how clearly statistics separate states (Table 7). Candidate bases are ranked by consistency across 16 windows:

$$h(b) = \frac{\mathrm{Std}_w(m_w)}{\mathrm{Std}(b)} + \mathrm{Std}_w(\log s_w) + 0.5\,\mathrm{Std}_w(\log q_w), \tag{19}$$

where m_w, s_w, and q_w are window mean, standard deviation, and root-mean-square first difference. Lower scores favor more stable bases. For every state pair, at least $\lceil \rho_s M \rceil$ variables satisfy

$$|\mu_{c,m} - \mu_{c',m}| \geq \Delta_\mu \quad \text{or} \quad \max(a_{c,m}/a_{c',m}, a_{c',m}/a_{c,m}) \geq r_a. \tag{20}$$

Separated parameter grids enforce these differences. For clear and medium profiles, grid spacings are multiplied by $\mathrm{clip}(\sqrt{96/\ell_{\min}}, 1, 1.5)$, where $\ell_{\min}$ is the shortest occurrence.

State level: stochastic processes. Supplementary states use fixed means, amplitudes, and smoothing widths. Each occurrence draws fresh Gaussian noise, smooths and standardizes it, then applies its state's parameters. Recurrence therefore depends on statistical identity even when individual waveforms differ. An explicitly labeled fluctuation F in $A \to F \to A$ is a new state; the weaker within-state perturbations below retain the label.

Temporal level: recurrence and duration. Balanced recurrence gives each state 2–4 occurrences with counts differing by at most one. Long-horizon and background paths vary recurrence context through long gaps and interleaved states (Table 9). Long-horizon and duration-contrast paths replace ordinary same-state duration ceilings. Background paths use the hard shared profile; prototype-driven backgrounds may reuse an activity's prototypes with amplitude $[0.03, 1.80]$, noise $[0, 0.05]$, and drift $[-0.03, 0.03]$. Duration-allocation parameters are 0.70 for duration contrast, 0.55 for long-horizon/activity segments, and 0.50 for background segments.

Observation level: sparse evidence and within-state variation. Sparse samples use 1, 2, or $\lceil M/2 \rceil$ informative variables with probabilities 0.50/0.30/0.20, capped at M. Each informative

Table 7: Shared-prototype profiles, distinct from embedding contrast. Probabilities are renormalized over feasible choices. A dash adds no bound.

Setting	Clear	Medium	Hard
Probability	0.65	0.30	0.05
Maximum stability quantile	0.60	0.80	1.00
Minimum length / crop jitter ρ	$96/0$	$64/0.03$	— / 0.10
Maximum same-state duration ratio	2.5	3.5	—
Informative fraction ρ_s	0.50	0.34	0.01
Mean gap Δ_μ / amplitude ratio r_a	$0.65/1.40$	$0.40/1.25$	$0.15/1.08$

Table 8: Stochastic-state settings. Amplitudes receive a per-variable factor in $[0.75, 1.25]$, except for explicitly labeled fluctuations.

Setting	Value
Smoothing width	$\{1, 3, 5, 9\}$ points
Low / high amplitude	$[0.03, 0.12]$ / $[0.80, 1.50]$
Low-state mean shifts	Reference mean zero; other states shift 1–$\lceil M/2 \rceil$ variables by magnitude $[0.45, 1.50]$
High-state mean / envelope	Probability 0.50 each: one-variable mean shift up to 0.45; sinusoidal envelope with occurrence-specific strength $[0.15, 0.45]$ and phase
Explicit fluctuation	Probability 0.15 when feasible; length 16–48, amplitude $[1.50, 2.50]$, smoothing width 1 or 3

Table 9: Temporal augmentation settings. Recurrence subtype probabilities are conditional on feasibility.

Construction	Structure
Balanced (0.90)	$C \in [4, 10]$; prefer $R \in [3C, 4C]$ with probability 0.70; otherwise $R = 2C + E$, $E \in [3, \min(4, C-1)]$. Introduced-state preference: 0.50.
Long horizon (0.10)	$C \in [4, 7]$, $R \in [24, 43]$; minimum length 64; two frequent states target 55–72% of segments and 4–6× duration contrast. Rare-state gap target: 55–95% of segment positions, with return in the last 30%; introduced-state preference: 0.70.
Recurring background	$q \to a_1 \to q \to a_2 \to \cdots \to q$; $C \in [3, 13]$, 0–4 extra activity occurrences. Background occupies 15–38% of L; background/activity lengths $[16, 394]/[32, 2048]$.
Duration contrast	One recurring state has 4–10× duration contrast; no crop jitter, occurrence warps, or boundary adjustment.
Long hold	Probability 0.15 for ordinary base prototype combinations; extend one segment to 2–5 times the original median, capped at 2048, when feasible.

variable uses high-contrast anchors. Others share continuous streams across the sequence, so independent crops cannot create spurious boundaries. The combined construction adds duration contrast to this sparse evidence.

Mild perturbations apply to recurring, clear/medium shared-prototype states on ordinary or recurrence-focused paths, with sample probabilities 0.35/0.50. The first occurrence receives $0.40\times$ strength without a global gain or offset; the second is perturbed when long enough, and later occurrences with probability 0.85. Smooth envelopes vanish at endpoints, and time warps share a mapping across variables. These changes preserve state descriptors. Boundary adjustment reduces splice artifacts

Table 10: Label-preserving observation perturbations.

Perturbation	Settings
Mild shared-prototype variation	Length ≥ 64; variable activation 0.75, at least $\lceil M/2 \rceil$ selected
Gain / offset	$[0.85, 1.18]$ / $[-0.12, 0.12]$ local robust scale
Amplitude / mean envelopes	Strength $[0.15, 0.40]$ / $[0.10, 0.25]$ local robust scale; 3–5 control points; total gain clipped to $[0.55, 1.60]$
Local time warp	Probability 0.65; interior rate $[0.75, 1.30]$
Stochastic noise window	Probability 0.65; 8–32 points, at most one quarter of a segment; variable activation 0.50; strength 8–25% of state amplitude
Boundary adjustment	4–14-point median comparison; offset factor $[0.15, 0.55]$, clipped to 1.5 local robust scales; cosine, linear, or exponential decay

without moving labels. Its probability is 0.65 for ordinary prototype combinations, 0/0.35/0.65 for clear/medium/hard shared profiles, and 0.15 for pure sparse samples. Medium and sparse adjustments cover at most 12.5% and 8% of a new segment. Background boundaries, duration-contrast, combined, and stochastic samples receive no boundary adjustment.

C.3 Generation controls and reproducibility

We use 128 workers and seed 123, with a 2% sampling probability for single-state, boundary-free examples. For multi-state samples, Table 11 combines the augmentation levels conditionally. Feasibility checks can change realized frequencies. Base and duration-contrast samples select prototype combinations or shared prototypes with probabilities 0.60/0.40. Base paths use ordinary/recurrence/background probabilities 0.55/0.30/0.15; pure sparse samples exclude background paths and renormalize the other two weights.

Table 11: Configured multi-state generation mixture.

Construction	Probability	Conditional choices
Base	0.60	Prototype, contrast, and path mixtures; mild variation when eligible
Duration contrast	0.13	Balanced recurrence, high contrast, varied occurrence lengths
Sparse boundaries	0.13	High contrast; ordinary or recurrence-focused paths
Combined	0.04	Sparse evidence with duration contrast and balanced recurrence
Stochastic	0.10	Ordinary paths; fresh occurrences under persistent statistical definitions

C.3.1 Observable separation and recurrence checks

Exact labels alone do not ensure visible differences. We therefore check whether different states remain separated, adjacent segments show a change, and repeated occurrences remain closer to their own state than to others. For shared-prototype and duration checks, a segment descriptor contains each variable's median, log robust scale, and tail-minus-head median. Location and trend are normalized by the full-sequence 1st–99th percentile range, scale is expressed relative to that range, and distances are divided by $\sqrt{3M}$. Robust scale uses 1.4826 MAD, with standard deviation for degenerate cases; head/tail windows use approximately one eighth of the segment.

Let d_{state} be the smallest distance between state centroids, d_{adj} the smallest adjacent-segment distance, and s_{within} the median distance of recurring segments to their own centroid. Separation is $R_{\text{sep}} = d_{\text{state}} / \max(s_{\text{within}}, 0.03)$. The recurrence ratio R_{reid} is the minimum, over recurring segments, of the nearest other-state distance divided by the own-state distance floored at 0.03. Higher ratios indicate clearer identity evidence. Mild variation also checks internal changes: divide segments into four windows and compare the 90th-percentile adjacent-window distance with $\max(d_{\text{adj}}, 0.03)$, giving R_{internal}. Sparse checks use per-variable median, scale, and first-difference roughness. They take the strongest informative-variable distance to avoid diluting sparse evidence: adjacent and state-centroid distances must reach 0.07, and the recurrence ratio must reach 0.85 (pure sparse) or 0.45 (combined). Base prototype combinations, hard shared profiles, and stochastic samples have no corresponding waveform rejection thresholds.

Table 12: Waveform acceptance thresholds. A dash disables a check. Mild variation also retains its shared profile's lower bounds.

Construction	$R_{\text{sep}} \geq$	$d_{\text{adj}} \geq$	$R_{\text{reid}} \geq$	$s_{\text{within}} \leq$	$R_{\text{internal}} \leq$
Clear shared	2.25	0.12	1.50	—	—
Medium shared	1.40	0.07	1.10	—	—
Mild variation	1.15	Profile	1.10	0.16	4.00
Duration contrast	0.90	—	0.65	0.36	—
Combined	0.75	—	0.55	0.40	—

C.3.2 Resampling and fallback

Failed distance or quality checks trigger bounded redraws. Distance fallback selects candidates with smaller violations while preserving distinct anchors and distinct base-state combinations. Structural requirements, including complete coverage, legal recurrence, and feasible durations, remain enforced. Statistical fallback stays within the specified parameter ranges. Quality resampling holds the construction family and profile fixed. If mild variation repeatedly fails, it is disabled and candidates are redrawn within that family. With quality relaxation, the candidate closest to the required thresholds may be retained with failed checks recorded; otherwise generation fails. Each sample's random stream depends on its index and the global seed, making generation independent of worker assignment. Saved settings and realized choices support reproduction and measurement of fallback frequencies.

D Hierarchical structured generation

We construct segmentation and hierarchy targets offline, then train the VLM to predict both from a global image without target-specific clustering.

D.1 Visual input and structured output

Variable panels share a time axis without annotation cues. The prompt supplies sequence length L and variable count M, and describes persistent changes and non-adjacent recurrence. For an illustrative $L = 1000$ sequence with path $1 \to 2 \to 1$, the target is

```
{"segments":[[0,200,1],[200,500,2],[500,1000,1]],
 "merges":[[1,2,3]]}
```

The integer triples encode half-open intervals with state IDs, and merges with two children and a parent. Here both occurrences of state 1 share one leaf. Answers contain only these two arrays. Image resizing does not change output coordinates $[0, L)$. Rendering details appear in Appendix E.

D.2 From state definitions to a merge tree

Persistent statistics complement morphology in Equation 10:

$$\boldsymbol{\phi}_{c,m} = [\log a_{c,m}, \mu_{c,m}, \sigma_{c,m}^{\text{noise}}, d_{c,m}]. \tag{21}$$

The entries encode log-amplitude, mean shift, noise scale, and drift. Stochastic samples use statistical vectors only. Occurrence durations, crop positions, and label-preserving perturbations are excluded, so recurring occurrences retain one state representation.

Let $D^{\text{hier}}_{c,c'} = \|\boldsymbol{r}_c - \boldsymbol{r}_{c'}\|_2$. For active groups of leaf states A and B, average linkage is

$$D^{\text{link}}(A, B) = \frac{1}{|A||B|} \sum_{c \in A} \sum_{c' \in B} D^{\text{hier}}_{c,c'}. \tag{22}$$

Each step merges the closest groups, resolving numerical ties by child IDs. Merge $j \in \{1, \ldots, C-1\}$ creates parent $C + j$; after $C - 1$ merges the root is $2C - 1$. A single-state sample has no merges. At inference, the model generates merges directly.

Applying each predicted merge relabels all occurrences of its children and coalesces newly equal adjacent segments. The resulting C nested partitions provide progressively coarser granularities (Appendix F.2). These candidates reflect the chosen state descriptors, their relative scales, and the linkage rule. They define one nested family, which need not represent every task-dependent grouping. Merging cannot recover a missing leaf boundary or split states incorrectly assigned the same leaf ID.

D.3 Adaptation, decoding, and validity

Only assistant answer tokens contribute to the autoregressive loss; image, prompt, and padding tokens are masked. LoRA settings and checkpoint selection appear in Appendix E.2.

After coalescing adjacent equal-state segments, validation requires exactly the keys `segments` and `merges`, positive-length integer intervals partitioning $[0, L)$, and leaf IDs $1, \ldots, C$ assigned by first appearance. The $C - 1$ merge triples must use two distinct active children in increasing ID order, create parent $C + j$ at step j, and leave one root. An invalid response receives one corrective regeneration using the original image, previous answer, and structural error, without reference labels. All evaluated sequences in the main comparison and ablations yield valid outputs; none are excluded because of generation failure. No deterministic boundary or merge completion is used in these evaluations.

E Experimental protocols and additional results

This appendix supplements Section 6 with benchmark statistics, training settings, scoring rules, ablation controls, and timing details.

E.1 Benchmarks and comparison conditions

The 404 sequences exclude one processed ActRecTut duplicate. Synthetic, MoCap, ActRecTut, and PAMAP2 follow E2USD (Lai et al., 2024); HAS and SKAB follow CLaP (Ermshaus et al., 2025). They cover synthetic recurring states, human activities, and industrial operations; HAS includes 31 single-state sequences.

Table 13: Benchmark statistics. Ranges are minima and maxima across sequences; lengths count original observations.

Dataset	Sequences	Variables	States	Segments	Raw length
Synthetic	100	4	5	13–20	19,999
MoCap	9	4	3–9	6–11	4,579–10,617
ActRecTut	2	5	6	43	31,392–32,577
PAMAP2	9	9	2–13	3–28	8,477–447,000
HAS	250	6–9	1–12	1–15	340–41,465
SKAB	34	8	2	2–3	745–1,327
Total	404				

All six sources are excluded before prototype mining; their observations and labels are not used for corpus construction or checkpoint selection. We also exclude UCR-derived benchmarks because

UCR observations occur in BLAST and cannot be reliably separated by instance. These exclusions apply to our constructed corpus.

Specialized baselines. The seven specialized baselines are fitted independently to each sequence using their released preprocessing and settings. TICC and ClaSP-KMeans receive the true state count, marked by K in the table. ClaSP-KMeans uses up to 40 candidate change points. Time2State and E2USD train window encoders and cluster with DPGMM; CLaP combines segmentation with label-profile-based grouping.

Foundation-model baselines. The four foundation baselines use no labels, state counts, or target-specific training. Qwen3.5-2B and TimeMaster-SFT-3B receive global images; ChatTS-8B receives at most 1024 uniformly sampled values per channel, and TimeOmni-1-4B receives 256 values per channel as text. Qwen, ChatTS, and TimeMaster use a shared compact-output prompt and greedy decoding. TimeMaster uses the CTU SFT checkpoint. TimeOmni uses its native reasoning/segment protocol with temperature 0.1, top-p 0.001, top-k 20, and repetition penalty 1.05. These conditions preserve each model's input modality but differ in temporal resolution.

A deterministic parser normalizes recoverable output formats and state names without labels. Unrecoverable outputs become single-state predictions; approximately 84% of ChatTS outputs use this fallback.

E.2 Training and inference

The generation recipe produces one million length-4096 sequences with 1–10 variables, at most 15 states, and at most 45 segments, using seed 123. The rendering/split seed is 42 and the validation fraction is 0.001 (Appendix C.3). Main and ablation results are single runs. Validation runs every 500 optimizer steps; checkpoints are selected by the lowest loss on held-out URME-generated data. Table 14 specifies the latest main-run LoRA recipe.

Table 14: Latest main-run LoRA training recipe.

Setting	Value
Backbone	Qwen3.5-2B
Trainable modules	Language LoRA; vision encoder and projector frozen
LoRA rank / scaling / dropout	16 / 32 / 0.05
Precision / quantization	BF16 / no 4-bit quantization
Optimizer / learning rate	Fused AdamW / 4×10^{-4}
Schedule / warmup / clipping	Cosine / 3% / 0.3
Epochs / GPUs	1 / seven RTX 4090 GPUs
Validation interval / selection	500 optimizer steps / lowest validation loss
Per-GPU batch / accumulation steps	1 / 16 (effective batch 112)
Memory and attention	Gradient checkpointing; FlashAttention-2
Loss	Cross-entropy on assistant output tokens only
Training pixel budget	131,072–786,432

Evaluation images plot raw observations in aligned variable panels, without time-domain resampling, state colors, or boundary annotations. Rendering uses 128 DPI, a 12-inch width, and 0.8-inch panels; the axis spans $[0, L)$. The vision processor may resize images within 131,072–786,432 pixels; the timing experiment uses a larger cap (Appendix E.6). Although every observation is plotted, rasterization and resizing compress temporal detail; retaining original output coordinates does not imply sample-level visual resolution.

Inference uses BF16, no 4-bit quantization, FlashAttention-2, greedy decoding, and at most 1024 new tokens. Structural validation and corrective regeneration follow Appendix D. One valid response provides all hierarchy levels.

E.3 Scoring and oracle definitions

ARI (Hubert & Arabie, 1985), NMI, and AMI (Vinh et al., 2010) compare pointwise states regardless of ID numbering. Scores are averaged within datasets, with equal dataset weights for macro results.

Main-text scores are scaled by 100; appendix accuracy tables are unscaled. These metrics assess overall pointwise partition agreement and do not separately quantify boundary localization, short-state recovery, or non-adjacent recurrence recognition.

For the main comparison and ablations, the oracle maximizes test-label ARI over the hierarchy, breaking ties by NMI; both metrics use that level. It bounds ARI within those candidates, while NMI can fall below leaf NMI. The leaf needs neither labels nor a state count. Sparse-label selection uses AMI on annotated timestamps, and its diagnostic oracle maximizes AMI on held-out timestamps (Appendix E.4); that oracle does not bound ARI or NMI.

E.4 Sparse-label granularity adaptation

The 404 hierarchies stay fixed; only level selection changes with label budgets. For annotation fraction $p > 0$, we request $B_p = \max\{2, \mathrm{round}(pL)\}$ timestamps. A deterministic van der Corput ordering spreads timestamps across the sequence without inspecting labels. Its prefixes form nested sets $A_{0.25\%} \subset A_{0.5\%} \subset A_{1\%} \subset A_{2\%}$. All budgets share the evaluation set

$$Q = \{0, \ldots, L-1\} \setminus A_{2\%}.$$

If the labels in A_p contain k_p states, eligible levels must predict at least k_p states on A_p. Among them, we select

$$l_p^* = \arg\max_l \mathrm{AMI}(\mathbf{y}_{A_p}, \widehat{\mathbf{y}}_{A_p}^{(l)}),$$

breaking ties within 10^{-12} toward the finest level. We use the leaf with no labels, fewer than two observed states, or no eligible level. The count constraint need not satisfy every observed state relation. The oracle maximizes AMI on Q over all levels with the same tie rule.

Gain intervals use 2000 paired-bootstrap resamples with seed 123: resample sequences within datasets, compute selected-minus-leaf gains, and average the six dataset means. The 2.5th and 97.5th percentiles give 95% intervals conditional on fixed predictions, excluding training-run variation. Figure 4 gives ARI/NMI gains. The 2% budget recovers about 50% of the AMI oracle–leaf gap. Percentage budgets measure timestamp counts, not annotation effort; the absolute count grows with sequence length. These results compare selection within ZEST's fixed candidates, rather than competing target-label-assisted methods or alternative merge trees.

E.5 Ablation controls and full results

The four corpus variants cross BLAST and KernelSynth-1M with URME and RandomCrop. KernelSynth-1M is the kernel-generated Chronos subset (Ansari et al., 2024). For its 1024-point sequences, URME uses a 256-point forecast context rather than the 1024-point context used for BLAST. The intended controls are shared MOMENT representations, scheduler settings, hierarchical targets, and the pretrained Qwen3.5-2B backbone.

URME selects low-uncertainty fragments and extends them using IAAFT (Section 5.1.1). RandomCrop draws raw fragments uniformly: 512–2048 points from 4096-point BLAST sequences, or 128–1024 points from 1024-point KernelSynth sequences. It skips uncertainty filtering and IAAFT; crops are interpolated when longer durations are needed. Thus, the source comparison also differs in fragment lengths, and the URME comparison jointly tests mining and extension.

Random initialization trains the same architecture on BLAST+URME for one epoch, randomly initializing and updating all vision, projector, and language parameters. It retains the tokenizer/processor and selects checkpoints by held-out loss. This tests pretraining under the available budget, but also changes the update scheme from LoRA to full-parameter training. Tables 15 and 16 give all dataset-level results underlying Figure 5.

E.6 Efficiency evaluation

ZEST has mean latency comparable to Time2State and CLaP. The comparison is implementation-specific, not hardware-normalized, and does not measure total raw-sequence-to-output latency.

All 250 HAS sequences are processed individually in a run separate from the main accuracy comparison. Time2State, E2USD, and ZEST each use one RTX 4090; CLaP uses a dual AMD EPYC

Table 15: Per-dataset ARI for the ablation study. Each configuration is evaluated at the leaf level and at the level selected by test-label ARI. Macro gives equal weight to the six datasets.

Configuration	Level	Synthetic	MoCap	ActRecTut	PAMAP2	HAS	SKAB	Macro
BLAST + URME (ZEST)	Leaf	0.9684	0.8133	0.8046	0.4049	0.4729	0.3996	0.6440
	Oracle	0.9687	0.8281	0.8711	0.4174	0.7287	0.5583	0.7287
KernelSynth + URME	Leaf	0.9596	0.8115	0.7099	0.3777	0.4596	0.2649	0.5972
	Oracle	0.9596	0.8318	0.8653	0.3782	0.7369	0.5127	0.7141
BLAST + RandomCrop	Leaf	0.9536	0.1511	0.7019	0.3844	0.5106	0.0144	0.4527
	Oracle	0.9537	0.1511	0.8698	0.3990	0.5420	0.0144	0.4883
KernelSynth + RandomCrop	Leaf	0.9648	0.7562	0.7726	0.3478	0.5294	0.2187	0.5983
	Oracle	0.9664	0.7917	0.8730	0.3533	0.7001	0.4529	0.6896
Random initialization	Leaf	0.8522	0.3634	0.7296	0.3931	0.3708	0.1970	0.4844
	Oracle	0.8533	0.3812	0.7307	0.4184	0.5516	0.3970	0.5554

Table 16: Per-dataset NMI for the ablation study. Each configuration is evaluated at the leaf level and at the level selected by test-label ARI. Macro gives equal weight to the six datasets.

Configuration	Level	Synthetic	MoCap	ActRecTut	PAMAP2	HAS	SKAB	Macro
BLAST + URME (ZEST)	Leaf	0.9422	0.8108	0.8132	0.6452	0.5675	0.4614	0.7067
	Oracle	0.9427	0.8252	0.8249	0.6456	0.7515	0.5659	0.7593
KernelSynth + URME	Leaf	0.9473	0.8268	0.7325	0.5163	0.5560	0.4160	0.6658
	Oracle	0.9473	0.8314	0.8188	0.6152	0.7561	0.5238	0.7488
BLAST + RandomCrop	Leaf	0.9409	0.1671	0.7393	0.6245	0.5449	0.0148	0.5053
	Oracle	0.9411	0.1671	0.8308	0.6232	0.5626	0.0148	0.5233
KernelSynth + RandomCrop	Leaf	0.9407	0.7843	0.7689	0.5816	0.6004	0.3384	0.6691
	Oracle	0.9416	0.7921	0.8261	0.5745	0.7166	0.4443	0.7159
Random initialization	Leaf	0.8700	0.5755	0.7162	0.6384	0.4756	0.3589	0.6058
	Oracle	0.8705	0.5788	0.7198	0.6594	0.6103	0.4086	0.6412

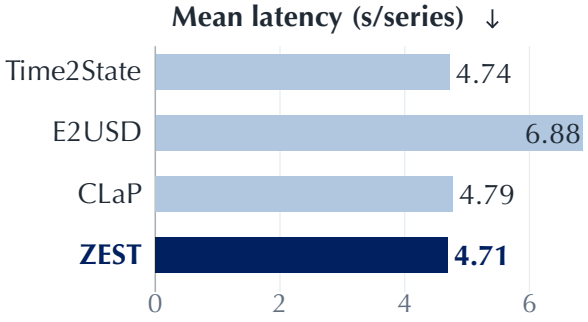


Figure 6: **Inference latency on HAS.** Mean over 250 sequences at batch size 1 under the stated configurations; image rendering is excluded.

9354 32-core server with 32 workers. Time2State/E2USD use seed 42, and CLaP/ZEST use seed 20260909. ZEST uses the inference settings above, batch size one, and a 131,072–1,048,576 pixel budget.

- **Time2State and E2USD:** include target-specific encoder training, window embeddings, DPGMM fitting, and pointwise-label reconstruction; exclude input reading, metrics, and output writing.
- **CLaP:** include input reading/cleaning, ADF-based distance selection, BinaryClaSP segmentation, ROCKET/CLaP fitting, state grouping, and metrics; exclude output writing.
- **ZEST:** include image loading, visual preprocessing, prompt construction, generation, validation, retries, hierarchy parsing, and metrics; exclude model loading, offline rendering, output writing, corpus construction, and training.

Metric computation costs 0.0069 seconds for CLaP and 0.0389 seconds for ZEST per sequence, each below 1% of the measured mean. Three ZEST predictions require a format retry: all 250 succeed after 253 generation attempts.

Table 17: HAS latency in seconds. The common subset removes the sequences corresponding to the first CLaP and ZEST calls from all methods.

	All 250 sequences			Common 248 sequences		
Method	Mean	Median	P90	Mean	Median	P90
Time2State	4.744	4.712	4.887	4.744	4.712	4.887
E2USD	6.883	6.808	7.010	6.884	6.808	7.010
CLaP	4.794	1.992	10.732	4.647	1.992	10.625
ZEST	4.713	3.941	8.705	4.679	3.941	8.523

The full-set means match Figure 6. First-call initialization takes 45.70 seconds for CLaP and 15.37 seconds for ZEST. ZEST peaks at 4.52 GiB allocated and 4.72 GiB reserved CUDA memory. These measurements describe the specified HAS implementations and hardware.

F Qualitative examples

The following two figures use the same saved ZEST prediction for MoCap `amc_86_07`, first showing its leaf partition and then its hierarchy. This sequence has the lowest leaf ARI among the nine MoCap sequences. It illustrates how merging can improve an overly fine partition.

F.1 Observed signals and leaf predictions

Figure 7 places the recorded signals, reference annotations, and leaf predictions on the same original time axis. All four channels and all samples are shown, without smoothing. Channels are independently scaled and vertically offset for display.

For readability, predicted colors are matched one-to-one to reference colors by temporal overlap; unmatched states use different colors. Color matching changes neither partitions nor scores. Saved state and tree-node IDs are shifted by one for display. State numbers are local to each prediction, not semantic classes shared across sequences.

F.2 Multiple granularities from one prediction

Figure 8 shows the saved tree and all eight levels for `amc_86_07`. Each merge groups every occurrence of its child states, including nonadjacent intervals. Tree height denotes merge order, not distance or confidence. A merged node takes the color of its longest-duration descendant leaf.

The first two merges combine predicted states 1, 7, and 6, recovering the recurring reference state 1. ARI increases from 0.685 at the leaf level to 0.894 after these merges, then decreases with further merging. Thus, coarser is not always better. The marked best level is selected using the reference labels and is therefore an oracle, not the sparse-label selector evaluated in Section 6.3.

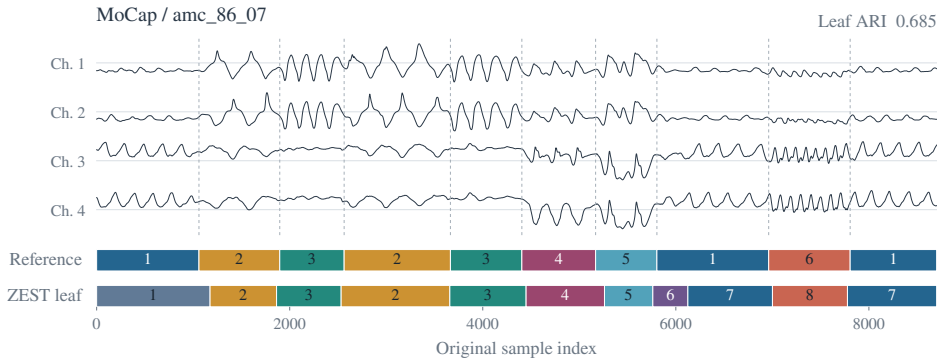


Figure 7: **MoCap signals, reference labels, and leaf predictions.** MoCap `amc_86_07`: six reference states and eight predicted states; leaf ARI 0.685. Dashed lines mark reference boundaries. Figure 8 continues with the hierarchy from this same prediction.

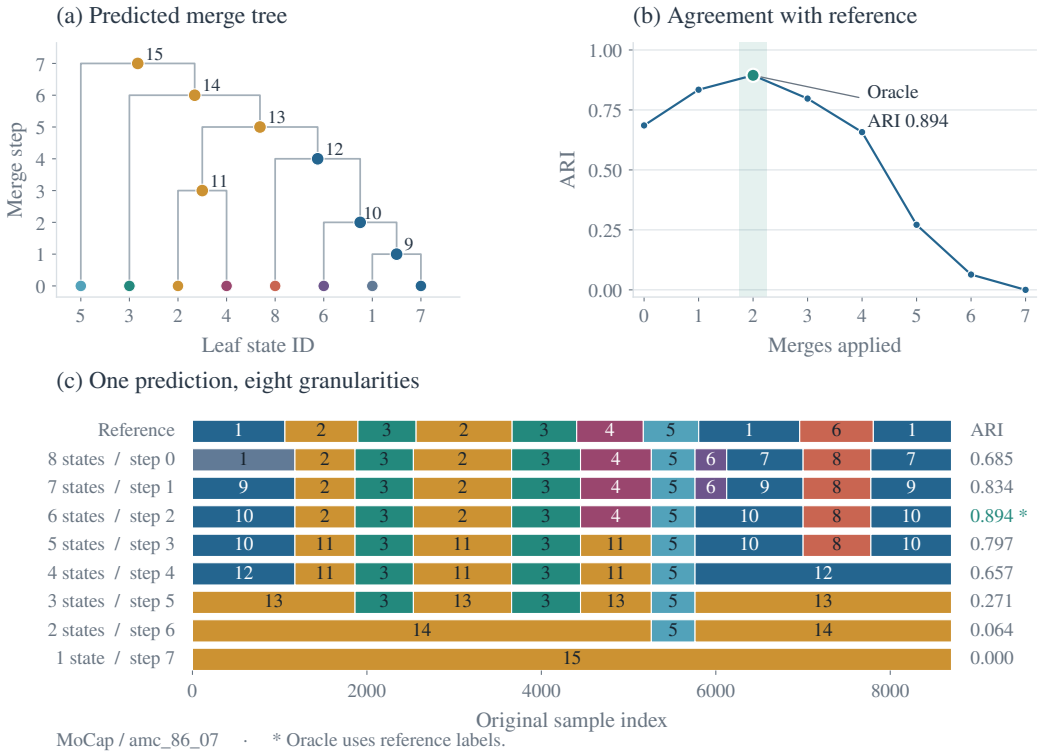


Figure 8: **Predicted hierarchy for MoCap `amc_86_07`.** (a) Predicted merge tree. (b) ARI at each level. (c) Reference and all predicted partitions, with one-based node IDs. All levels come from the same output; no new model inference or clustering is performed. The oracle uses reference labels.

F.3 PROTOTYPE MINING

Figure 9 follows one BLAST sequence from fragment selection to prototype extension. The shaded interval is the saved extracted fragment. This selected periodic example illustrates selection and extension of a non-flat pattern; it does not estimate aggregate mining quality. Extension constructs a longer prototype, not a forecast of the source's future. The extraction and extension procedures are described in Appendix C.1.

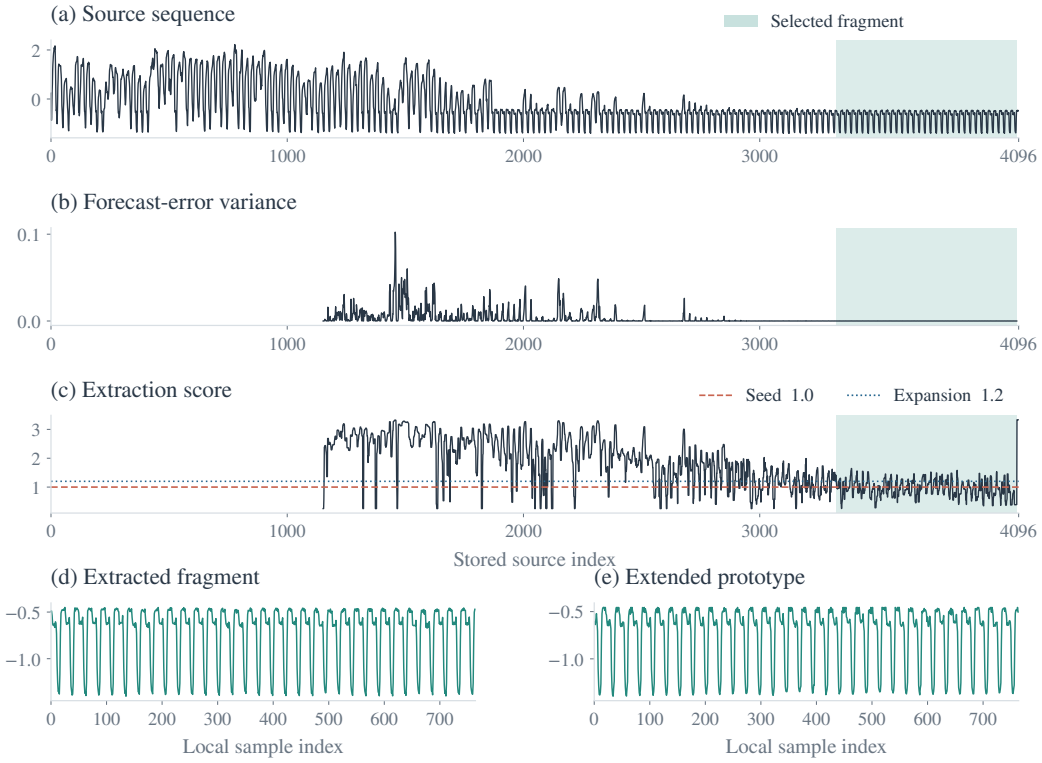


Figure 9: **From a real sequence to a reusable prototype.** (a) Source sequence. (b) Saved forecast-error variance. (c) Extraction score, seed threshold (1.0), and expansion threshold (1.2). Shading marks the final selected interval after expansion and short-gap joining, using the archived extraction indices. (d) Extracted fragment. (e) An equal-length prefix of its saved 8192-point extension, making the recurring pattern easier to compare.

F.4 STATE CONSTRUCTION AND AUGMENTATION

Figure 10 illustrates how selected prototypes define multivariate states. Each state assigns a prototype to each variable. Transformed crops are then placed along a sampled state path; repeated occurrences retain the same state ID. Thus, boundaries and labels follow directly from construction, without labeling the unannotated source sequences.

Figure 11 complements this example with three prototype-contrast settings, states that share prototypes but differ in statistics, changes confined to selected variables, and recurrence after a long gap. Contrast refers to distance between prototype embeddings, not guaranteed visual difficulty. These are separate examples, not a controlled comparison of contrast levels.

These length-4096 examples are selected from eight candidates per construction type using 1024 randomly sampled library prototypes. They illustrate the procedures in Appendix C.2, rather than their corpus frequencies or model performance. Channels are independently scaled for display, without smoothing; state IDs and colors are local to each example.

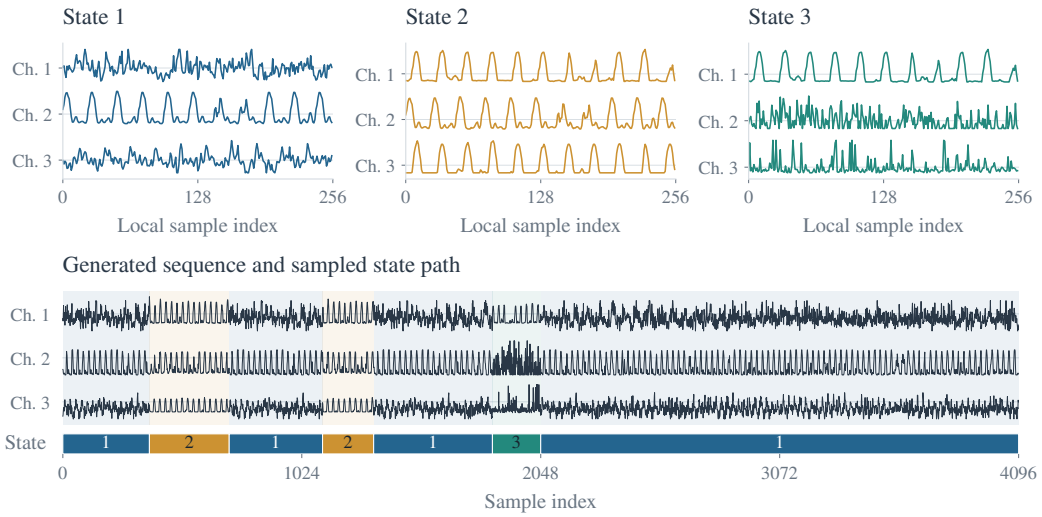


Figure 10: **From prototypes to a labeled multivariate sequence.** Top: the first 256 points of each selected prototype, grouped by state. Bottom: the generated three-variable sequence and its state path. Colors identify states within this example.

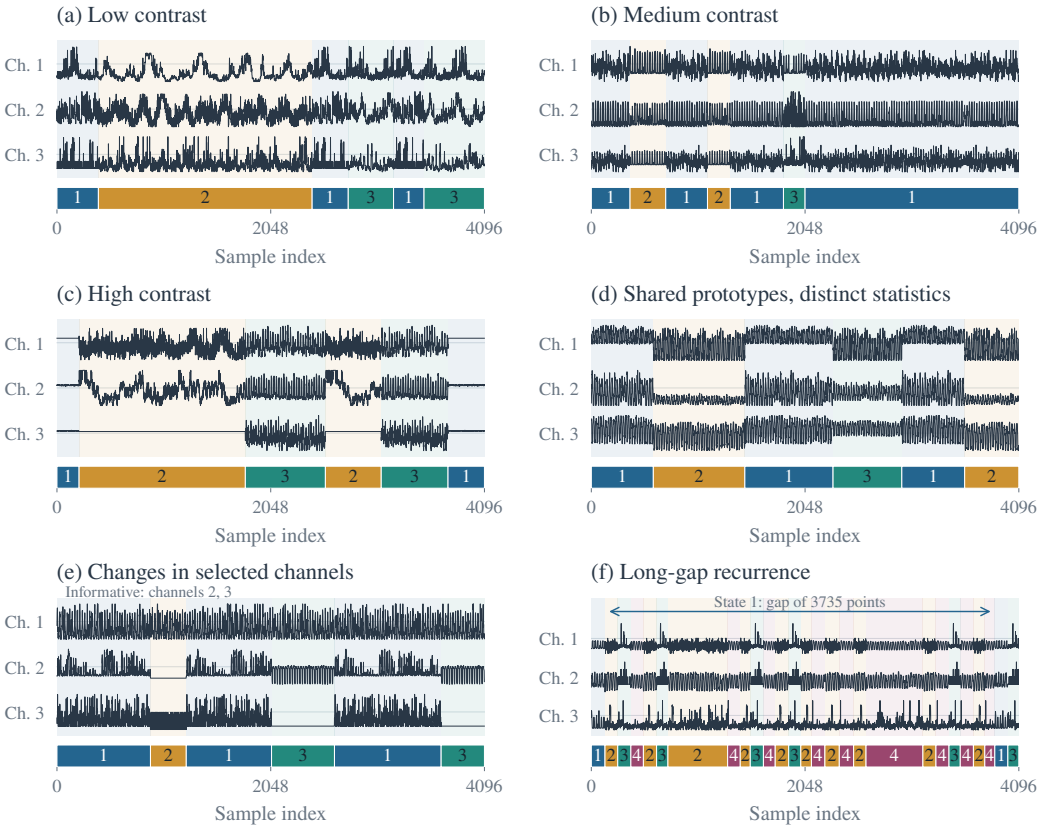


Figure 11: **Examples of state diversity and recurrence.** (a–c) Low, medium, and high prototype contrast. (d) Shared prototypes with state-specific statistics. (e) State changes carried by selected channels. (f) A state returning after a long absence. Colored strips show the exact generated labels.