PairAlign: A Framework for Sequence Tokenization via Self-Alignment with Applications to Audio Tokenization

Anonymous authors
Paper under double-blind review

Abstract

Modern learning systems represent perceptual signals with continuous vectors, but many operations over sensory data—comparison, retrieval, memory, alignment, and reasoning—are naturally expressed over discrete symbolic sequences. In language, this interface is given a priori through tokens; for continuous sensory signals such as speech and audio, it must be learned. Existing audio tokenizers typically induce symbols through frame- or short-window-level quantization, clustering, or reconstruction, leaving sequence-level properties such as cross-realization consistency, learned length, termination, compactness, and edit geometry only indirectly controlled. We introduce **PairAlign**, a framework for learning compact audio token strings through sequence-level self-alignment. PairAlign formulates tokenization as conditional sequence generation: an encoder maps speech to a continuous condition, and an autoregressive decoder emits a symbolic sequence from BOS to EOS, learning token identity, order, length, and termination. Given two content-preserving views, each view's token string is trained to be likely under the other's representation, while unrelated in-batch examples provide competing sequences. This gives a scalable surrogate for edit-distance preservation while discouraging many-to-one collapse. Starting from a VQ-style geometric tokenizer, PairAlign extends a frame-synchronous geometric prior into an autoregressive sequence tokenizer. It first uses deterministic VQ-derived targets as a bridge, then transitions to adaptive EMA-teacher sequence tokenization using cross-paired teacher forcing, anti-bypass regularization, likelihood contrast, length-constrained decoding, and post-hoc timing recovery. On continuous 3 s speech, PairAlign learns compact, broad-vocabulary, non-degenerate token strings with strong cross-view consistency. In retrieval, it operates at 12.71 tokens/s and reduces archive token count by approximately 55% relative to the VQ-style geometric baseline while preserving meaningful edit-distance search. The results expose a compactness–locality trade-off: PairAlign does not aim to dominate high-rate geometric or pretrained SSL tokenizers on every local retrieval metric. Instead, it provides a lower-rate symbolic interface for comparison, retrieval, and structural analysis. More broadly, PairAlign is a sequence-symbolic analogue of JEPA-style predictive learning, where the predicted target is a learned variable-length symbolic sequence rather than a fixed-dimensional continuous latent.

1 Introduction

Toward symbolic interfaces for audio. Discrete token sequences are useful interfaces for comparison, retrieval, alignment, memory, editing, generation, and structured reasoning. In language, this interface is given by text; in audio, it must be learned from continuous signals. Audio tokenization is therefore central to neural codecs, speech and music generation, text-conditioned audio synthesis, multimodal modeling, and retrieval-oriented speech processing (Mousavi et al., 2025). Existing tokenizers are effective, but most induce symbols through local frame- or short-window-level assignment: codec tokenizers optimize reconstruction-oriented codes, semantic tokenizers discretize self-supervised speech representations, and hierarchical or factorized tokenizers distribute information across codebooks (Zeghidour et al., 2021; Défossez et al., 2022; Kumar et al., 2023; Baevski et al., 2019; Hsu et al., 2021; Chung et al., 2021; Zhang et al., 2023; Ju

et al., 2024). PairAlign addresses a complementary question: how to learn the emitted token string itself, including ordering, compactness, stability under content-preserving perturbations, edit-distance geometry, learned length, and termination.

Why framewise assignment is not enough. Geometric discretization is a strong inductive bias: assigning each encoder frame to a nearby centroid or quantizer entry preserves timing and local detail, and works well for compression, reconstruction, and frame-synchronous modeling. However, the resulting token stream is not usually optimized as a string. This matters for retrieval, matching, indexing, alignment, and edit-based comparison, where related views (x, x^+) should have closer token strings than unrelated examples x^-. Edit distance is natural for this requirement because it jointly accounts for identity, order, substitutions, insertions, deletions, and length. Retrieval-oriented systems such as wav2tok and BEST-STD motivate this symbolic view (Banerjee & Arora, 2022; Singh et al., 2025), but edit distance is combinatorial and non-differentiable, making it difficult to optimize directly inside neural tokenizers.

Learning the string, not only the frame labels. We introduce **PairAlign**, a framework for learning compact discrete audio token sequences through sequence-level self-alignment. Conceptually, PairAlign follows the predictive-abstraction principle behind JEPA-style learning: rather than reconstructing low-level observations, the model predicts an abstract target associated with another view. The difference is that the target is not a fixed-dimensional continuous representation, but a learned variable-length symbolic sequence. Given a speech segment x, an encoder produces a continuous condition

$$Z = Enc(x), \tag{1}$$

and an autoregressive decoder defines

$$p(\mathcal{T} \mid Z) = \prod_{l=1}^{|\mathcal{T}|} p(\tau_l \mid \tau_{<l}, Z). \tag{2}$$

At inference time, decoding starts from BOS and terminates by emitting EOS. Token identity, order, length, and termination are therefore learned sequence-level decisions rather than direct consequences of encoder frame rate, nearest-centroid assignment, or de-duplication.

The learning signal is paired-view self-alignment. Given two content-preserving realizations (x, x^+), with representations Z and Z^+, the token sequence induced from one view should receive high conditional likelihood under the other:

$$\log p(\mathcal{T}^+ \mid Z) \qquad \text{and} \qquad \log p(\mathcal{T} \mid Z^+). \tag{3}$$

Unrelated in-batch examples provide competing symbolic sequences that should score lower under the same condition. PairAlign thus uses cross-paired conditional likelihood as a scalable surrogate for edit-geometry preservation: related realizations should induce mutually predictable strings, while unrelated inputs should remain separated.

From stable geometry to adaptive symbols. PairAlign is trained with a conservative three-stage pipeline. Stage I learns a contextual speech encoder and nearest-centroid VQ tokenizer using self-supervised contrastive learning and a commitment objective; this geometric tokenizer also serves as the main controlled baseline. Stage II freezes the Stage I encoder and quantizer, and trains an autoregressive decoder on deterministic token targets produced by the frozen tokenizer under cross-paired conditioning. Stage III replaces the fixed teacher with an EMA teacher over the full encoder–decoder tokenizer: the teacher generates adaptive free-running token sequences, while the student aligns paired views and separates hard in-batch negatives. PairAlign therefore uses geometric tokenization as a stable initial interface, then refines it through adaptive sequence-level self-alignment.

Making autoregressive tokenization stay grounded. Autoregressive tokenization is expressive, but it introduces failure modes that do not arise in the same form for framewise tokenizers. The decoder may emit repetitive strings, map unrelated inputs to similar outputs, produce unstable lengths, or behave like

an unconditional language model over token strings. The central issue is *decoder bypass*: under teacher forcing, target-side prefixes and token-continuation statistics can explain the objective while the acoustic condition Z is underused. PairAlign treats this as a design constraint and combines cross-paired likelihood, prefix corruption, encoder-summary conditioning, structured self-attention dropout, hardest-K likelihood contrast, entropy regularization, repetition-aware teacher generation, explicit length constraints, and differential encoder–decoder learning rates. The goal is not merely to add a decoder, but to make autoregressive tokenization remain input-grounded while the symbolic interface itself is being learned.

The price of compactness. Because the decoder emits EOS, output length is no longer tied to encoder frame rate. This enables shorter symbolic sequences for storage, retrieval, edit-distance comparison, and downstream sequence processing. The trade-off is that compact autoregressive tokens are not natively timestamped. When temporal grounding is needed, we recover approximate timestamps post hoc: decoder cross-attention is treated as a soft token-to-frame association, a weak monotonicity bias is applied, and monotone Viterbi decoding extracts an approximate segmentation of encoder frames into decoded token positions. This procedure is inference-only; it does not modify training, impose hard monotonicity, or affect ordinary token-space retrieval and comparison.

What must a learned symbolic interface preserve? We evaluate PairAlign as a learned symbolic interface, not merely as a source of discrete codes. The main experiments test compactness, paired-view stability, edit-distance retrieval, vocabulary non-collapse, local sequence structure under controlled context shifts, and optional post-hoc timing recovery from cross-attention. We further include two stress tests of the learned similarity geometry: lexically disjoint Tamil transfer, and comparison with large-data-exposure pretrained SSL tokenizers such as HuBERT+VQ and WavLM+VQ. These comparisons clarify the operating point of PairAlign. Dense geometric and pretrained SSL tokenizers provide strong acoustic-symbolic references with more local evidence for retrieval, while PairAlign shows that a frame-synchronous geometric prior can be extended into a compact autoregressive sequence tokenizer through paired self-alignment. Overall, PairAlign learns a substantially lower-rate, non-collapsed, input-grounded symbolic sequence that preserves meaningful edit-distance structure, while exposing a compactness–locality trade-off relative to denser geometric and pretrained SSL tokenizers.

Contributions. This paper makes four contributions. **First**, it formulates audio tokenization as input-conditioned sequence generation rather than only frame-level geometric assignment, so token identity, order, length, and EOS termination are learned directly. **Second**, it introduces paired self-alignment as a scalable surrogate for edit-distance preservation: related acoustic realizations are encouraged to induce mutually predictable token strings, while unrelated in-batch examples are separated through likelihood contrast. **Third**, it provides a controlled recipe for extending a frame-synchronous geometric prior into an autoregressive sequence tokenizer, moving from a VQ tokenizer to a deterministic-teacher autoregressive bridge and then to EMA-teacher self-alignment, while addressing decoder bypass, repetition, unstable length, and collapse. **Fourth**, it evaluates the resulting token sequences across consistency, compactness, inventory usage, collapse, edit-operation structure, retrieval, continuous-sweep local evolution, lexically disjoint transfer, pretrained SSL comparisons, and timing recovery. The results show that PairAlign does not dominate dense geometric or pretrained SSL tokenizers on every local retrieval metric, but learns a lower-rate, non-collapsed, input-grounded symbolic sequence that reduces token count and comparison cost while preserving useful edit-distance structure.

Beyond speech tokenization. Although this paper studies continuous-speech tokenization, the framework is broader. PairAlign applies whenever continuous or weakly structured inputs must be mapped into compact symbolic sequences that remain stable under nuisance variation while preserving discriminative content. From a JEPA-style predictive learning perspective, PairAlign replaces continuous latent prediction with symbolic sequence prediction: the learned target is a variable-length string with its own vocabulary, order, length, and termination behavior. More generally, the results suggest that self-alignment can be used not only to evaluate symbolic representations, but also to induce and organize them. This points toward predictive systems that learn compact discrete event interfaces across views, time, tasks, and modalities.

2 Related Work

Audio tokenizers as frame-derived symbolic interfaces. Discrete audio tokenization is now central to neural audio modeling. Neural codec tokenizers such as SoundStream, EnCodec, and related high-fidelity codecs learn discrete codes optimized for reconstruction, compression, and generation (Zeghidour et al., 2021; Défossez et al., 2022; Kumar et al., 2023). Semantic speech tokenizers discretize representations from self-supervised models such as wav2vec, HuBERT, and WavLM (Baevski et al., 2019; Hsu et al., 2021; Chung et al., 2021; Chen et al., 2022), while recent hierarchical and factorized tokenizers distribute acoustic, semantic, speaker, prosodic, or residual information across multiple streams (Zhang et al., 2023; Ju et al., 2024). These systems provide strong frame-derived symbolic interfaces for synthesis, compression, audio-language modeling, and generation. PairAlign addresses a complementary question: how to learn the emitted token string itself as a sequence-level symbolic object with learned order, compactness, stability, length, termination, and edit geometry.

From geometric assignment to learned sequence induction. Most audio tokenizers first encode audio into continuous frame or window representations and then assign each local representation to a code by nearest-centroid assignment, VQ, clustering, residual quantization, or related local rules. This preserves timing and local detail, and it gives a stable interface for downstream language modeling over audio tokens. However, the token sequence structure is largely fixed before the language model is trained: encoder stride, feature head, quantizer, codec rate, windowing, and post-processing determine the temporal grid, ordering, length, and termination behavior. PairAlign instead asks whether the emitted token string can itself be the object of tokenization. It keeps geometric discretization as a stable initialization, but then induces token identity, order, length, termination, and cross-view stability through paired-view conditional likelihood.

Retrieval-oriented tokenizers and edit-distance comparison. Retrieval-oriented tokenizers such as wav2tok and BEST-STD motivate discrete speech tokens as symbolic interfaces for query-by-example spoken term detection and related matching tasks (Banerjee & Arora, 2022; Singh et al., 2025). Edit distance is natural in this setting because it compares token identity, order, substitutions, insertions, deletions, and length, but it is combinatorial and non-differentiable. wav2tok is the closest precursor to PairAlign: it uses a CTC-style pairwise alignment constraint to encourage paired speech realizations to produce compatible frame-indexed token sequences. PairAlign preserves this relational motivation, but moves beyond retrieval-oriented frame tokenization by learning the emitted sequence itself through input-conditioned autoregressive generation. Retrieval is therefore used here as a probe of edit-distance structure, not as the sole definition of tokenizer quality.

Sequence transduction aligns known labels; PairAlign induces them. CTC and RNN-T provide sequence-level likelihoods when the target label sequence is known but its alignment to input frames is latent (Graves et al., 2006; Graves, 2012). CTC marginalizes monotone alignments with frame-indexed posteriors and a collapse map, while RNN-T also conditions on previously emitted labels. PairAlign is related in its use of sequence-level likelihoods, but differs in the object being learned: the target is not an external transcript or label string. It is a learned audio-token sequence whose identity, order, length, and EOS behavior are induced during training.

Predicting symbolic abstractions instead of continuous latents. Self-supervised predictive representation learning often predicts abstract targets associated with another view, region, or time step rather than reconstructing raw observations. JEPA-style methods formalize this principle by predicting representations instead of pixels, samples, or other low-level signals (LeCun et al., 2022; Assran et al., 2023; Bardes et al., 2024; Fei et al., 2023). PairAlign follows this predictive-abstraction view, but changes the target space from a fixed-dimensional continuous latent to a learned variable-length symbolic sequence induced from a paired content-preserving view. It can therefore be viewed as a sequence-symbolic analogue of predictive self-supervision, where the representation and the discrete sequence interface are learned jointly.

Alignment before the tokenizer is fixed. PairAlign uses the term *alignment* differently from alignment in LLM post-training. RLHF, RLAIF, Constitutional AI, DPO, IPO, KTO, RLVR, and related methods

reshape behavior over an already fixed tokenizer and vocabulary (Stiennon et al., 2020; Ouyang et al., 2022; Bai et al., 2022a;b; Rafailov et al., 2023; Azar et al., 2024; Ethayarajh et al., 2024; Shao et al., 2024; Guo et al., 2025). PairAlign instead learns the symbolic interface itself: token identity, order, length, and termination evolve during training, and the alignment signal comes from paired-view data geometry rather than external preference labels. In this work, alignment is therefore a representation-learning principle for inducing and stabilizing a token space.

A decoder, but not a prompt-continuation model. PairAlign uses conditional autoregressive likelihoods, but it is not a standard language-modeling, prompt-continuation, or text sequence-to-sequence objective. GPT-style models learn continuation, BERT predicts masked symbols, XLNet changes the factorization, UniLM unifies attention masks, and MASS/BART perform denoising sequence-to-sequence reconstruction (Radford et al., 2018; 2019; Devlin et al., 2019; Yang et al., 2019; Dong et al., 2019; Song et al., 2019; Lewis et al., 2020). PairAlign differs in both the condition and the generated object: the input is continuous speech, the output symbols are learned audio tokens, and inference starts from BOS without a text prompt, semantic prompt, acoustic-token prompt, or target-side prefix. The acoustic condition must determine the token trajectory, identities, length, and EOS placement.

3 Methodology

3.1 Problem Formulation: Sequence Tokenization as Conditional Language Modeling

Let $\mathcal{D} = \{x_i\}_{i=1}^{N}$ be a dataset of variable-length signals. An encoder maps each input to a sequence of continuous latent states,

$$Z = Enc(x), \qquad Z \in \mathbb{R}^{d \times T}, \tag{4}$$

where d is the latent dimension and T is the downsampled temporal length. We define *sequence tokenization* as mapping Z to a compact discrete string

$$\mathcal{T} = [\tau_1, \ldots, \tau_L], \qquad \tau_l \in \mathcal{A}, \tag{5}$$

where $\mathcal{A}$ is a finite alphabet and typically $L \ll T$. The goal is not only to assign local frame labels, but to learn a symbolic sequence whose identity, order, length, and edit geometry support comparison, retrieval, alignment, memory, and downstream sequence modeling.

PairAlign formulates tokenization as conditional language modeling. Given encoder states Z, an autoregressive decoder defines

$$p(\mathcal{T} \mid Z; \theta_{\text{AR}}) = \prod_{l=1}^{L} p(\tau_l \mid \tau_{<l}, Z; \theta_{\text{AR}}). \tag{6}$$

Thus, sequence length and termination are not inherited from encoder stride or framewise quantization; the tokenizer learns which symbols to emit, in what order, and when to stop.

Target property: preserving sequence-level relational structure. The desired token space should preserve similarity relations between inputs. For a content-preserving pair (x, x^+) and an unrelated example x^-, with tokenizations $\mathcal{T}$, $\mathcal{T}^+$, and $\mathcal{T}^-$, we would like

$$ED(\mathcal{T}, \mathcal{T}^+) < \min\{ED(\mathcal{T}, \mathcal{T}^-), ED(\mathcal{T}^+, \mathcal{T}^-)\}, \tag{7}$$

where $ED(\cdot, \cdot)$ denotes edit distance. This captures the central relational requirement: related views should induce compatible symbolic strings, while unrelated inputs should remain separated.

Edit distance as the sequence geometry. Edit distance is appropriate because it compares token identity, order, length, substitutions, insertions, and deletions. Unlike unordered overlap, it respects the fact that the representation is a string, which is important for audio where content and timing are expressed through ordered symbolic structure. The constraint in Eq. 7 also excludes trivial stability: mapping all inputs to the same generic string would make positive and negative pairs equally close. The token space must therefore be stable across content-preserving views while remaining discriminative across unrelated inputs.

Why edit distance is not optimized directly. Although Eq. 7 defines the desired geometry, direct edit-distance optimization is not practical for neural tokenizer training. Edit distance depends on discrete identities and hard alignment choices, is piecewise constant with respect to model logits, and provides little useful gradient. A direct ranking objective would also require decoding variable-length strings and recomputing dynamic-programming alignments against many negatives during training. PairAlign therefore uses edit distance as the evaluation geometry, but optimizes a differentiable sequence-likelihood surrogate.

Cross-view likelihood as a differentiable surrogate. For a paired example (x_i, x_i^+), let

$$Z_i = Enc(x_i), \qquad Z_i^+ = Enc(x_i^+),$$

with token sequences $\mathcal{T}_i$ and $\mathcal{T}_i^+$. Instead of directly minimizing $ED(\mathcal{T}_i, \mathcal{T}_i^+)$, PairAlign makes one view's token sequence likely under the other view's representation:

$$\log p(\mathcal{T}_i^+ \mid Z_i), \qquad \log p(\mathcal{T}_i \mid Z_i^+). \tag{8}$$

These autoregressive likelihoods provide differentiable token-level credit assignment while remaining sensitive to order, length, and EOS placement. The surrogate is not equivalent to edit-distance minimization; it is an optimization-friendly proxy for the same relational objective.

From pairwise predictability to a discriminative symbolic space. Positive-pair predictability alone is insufficient, since a collapsed tokenizer could make many inputs mutually predictable by assigning generic strings. PairAlign therefore trains on minibatches of disjoint positive pairs, $\{(x_i, x_i^+)\}_{i=1}^{B}$, and treats token sequences from other batch elements as negatives. The paired sequence should receive higher conditional likelihood than mismatched sequences, converting edit-geometry preservation into a discriminative sequence-likelihood objective.

Similarity in the audio setting. In this paper, two audio segments are treated as similar when they preserve the same spoken content while differing in nuisance acoustics. We construct such pairs through mild augmentations, $x^+ = \mathrm{Aug}(x)$, including gain changes, additive noise, filtering, reverberation, and short masking. The learning signal is therefore content-preserving cross-view predictability, not transcript supervision.

Training path: from geometric tokens to self-aligned sequences. PairAlign is trained in three stages. Stage I learns a contextual encoder and nearest-centroid VQ tokenizer, providing a controlled VQ-style geometric baseline. Stage II freezes this tokenizer and trains an autoregressive decoder to predict one view's deterministic token sequence from the paired view's representation. Stage III replaces the fixed teacher with an EMA teacher over the full encoder–decoder model, allowing the encoder, decoder, output length, and generated token targets to co-evolve under cross-view self-alignment, hard-negative likelihood contrast, and anti-collapse regularization. In JEPA-style terms, PairAlign predicts an abstract target associated with another view; unlike standard continuous latent prediction, the target is a learned variable-length symbolic sequence with its own vocabulary, order, and termination behavior.

3.2 Stage I: Learning a Base Encoder and VQ Tokenizer

Stage I provides the geometric starting point for PairAlign. It learns contextual frame-level representations and converts them into discrete symbols through nearest-centroid vector quantization. This stage is both a valid tokenizer in its own right and the controlled baseline against which the later sequence-level stages are compared.

Contextual sequence encoder. We use a unidirectional Mamba encoder $Enc(.)$ based on selective state-space models (Dao & Gu, 2024). Structured state-space models provide efficient sequence modeling through recurrent state dynamics (Gu et al., 2021), and Mamba makes the state-space parameters input-dependent, allowing the encoder to model long-range temporal dependencies efficiently. PairAlign is not tied to this specific encoder; the key contribution lies in the sequence-level tokenization objective built on top of the encoder states.

Self-supervised frame-level representation learning. The encoder is trained with the frame-level contrastive objective used in BEST-STD (Singh et al., 2025). Paired utterances are aligned with dynamic time warping, aligned frames are treated as positives, and unrelated frames in the batch provide negatives. A commitment term pulls encoder outputs toward their assigned VQ centroids. The Stage I objective is

$$\mathcal{L}_{\text{StageI}} = \mathcal{L}_{\text{contrast}} + \lambda_{\text{commit}} \mathcal{L}_{\text{commit}}. \tag{9}$$

This yields discriminative contextual embeddings suitable for geometric discretization.

Nearest-centroid vector quantization. Given encoder embeddings $Z_i = [z_{i,t}]_{t=1}^{T}$, a VQ codebook $C = \{c_a : a \in \mathcal{A}\}$ assigns each frame to its nearest centroid:

$$\tau_{i,t} = \arg\min_{a \in \mathcal{A}} \|z_{i,t} - c_a\|_2^2. \tag{10}$$

This produces a raw frame-synchronous token sequence. The codebook centroids are updated from the current encoder assignments using an exponential moving average over assigned frame counts and assigned encoder states, following the standard VQ update used to keep centroid estimates stable during training. For sequence-level comparison, we apply run-length deduplication $\phi(\cdot)$ to remove consecutive repetitions:

$$\mathcal{T}_i = \phi([\tau_{i,1}, \ldots, \tau_{i,T}]).$$

The resulting deduplicated VQ string is the Stage I geometric tokenizer and serves as the deterministic target source for Stage II. Full implementation details for Stage I, including the contrastive objective, commitment loss, nearest-centroid assignment, EMA codebook update, and run-length deduplication, are provided in Appendix A.

Relation to VQ-VAE-style tokenization. Stage I deliberately matches the dominant geometric-tokenization paradigm: continuous encoder latents are discretized by nearest-neighbor assignment to a learned codebook, as in VQ-VAE-style tokenizers and related audio-tokenization methods (Van Den Oord et al., 2017; Baevski et al., 2019; Hsu et al., 2021; Chung et al., 2021). This makes Stage I a meaningful bridge to neural codecs, semantic speech units, and retrieval-oriented VQ tokenizers. It also isolates the contribution of PairAlign: later improvements are not obtained by replacing a weak tokenizer, but by adding sequence-level conditional generation and self-alignment on top of a strong VQ-style baseline.

Optional sequence-level strengthening of the VQ tokenizer. We also consider a Stage I+ baseline that augments the geometric tokenizer with a wav2tok-style pairwise no-blank CTC objective (Banerjee & Arora, 2022). Stage I first learns a stable encoder and VQ codebook using the frame-level contrastive and commitment losses. Stage I+ then adds a soft sequence-level constraint: for paired views (x_i, x_i^+), the deduplicated VQ sequence from one view is used as the target sequence for the other view.

Concretely, the conditioning view's encoder latents are converted into framewise token probabilities over the VQ alphabet,

$$p_{i,t}(a) = p(\tau = a \mid z_{i,t}), \qquad a \in \mathcal{A},$$

using the same geometric token space as nearest-centroid assignment. The target sequence is then scored with a no-blank CTC likelihood, which marginalizes over monotonic frame-level paths that collapse to the target:

$$p_{\text{CTC}}(\mathcal{T} \mid Z) = \sum_{\pi:\, \phi(\pi) = \mathcal{T}} \prod_{t=1}^{T} p(\pi_t \mid z_t).$$

Thus, Stage I+ encourages $\mathcal{T}_i^+$ to be recoverable from Z_i, and $\mathcal{T}_i$ to be recoverable from Z_i^+, under a monotonic alignment model. This injects order-sensitive sequence pressure into the VQ tokenizer while keeping token identity frame-synchronous and nearest-centroid based.

Because the CTC sequence likelihood operates on a different numerical scale from the frame-level contrastive objective, we use an adaptive weight for its contribution. The weight is set from the ratio of minibatch-averaged contrastive and sequence losses, with a small stabilizer, so that the sequence-level term remains proportional to the representation-learning objective rather than dominating the second-phase optimization.

Stage I+ is therefore an intermediate comparison between pure geometric tokenization and PairAlign. It strengthens the VQ tokenizer through monotonic sequence-likelihood consistency, whereas PairAlign proper learns compact token sequences through conditional autoregressive generation with learned length and EOS placement. The no-blank CTC likelihood, symmetric pairwise sequence-consistency loss, adaptive weighting, and complete Stage I+ objective are provided in Appendix B.

3.3 Stage II: Learning Cross-Paired Conditional Sequence Prediction

Stage II introduces the autoregressive decoder on top of the frozen Stage I encoder and VQ codebook. The goal is not yet to adapt the symbolic targets, but to learn an input-grounded sequence model over a stable token space. The frozen Stage I tokenizer provides deterministic VQ-derived strings, and the decoder is trained to predict one view's string from the paired view's encoder representation. This makes pairwise self-alignment a stationary conditional sequence-prediction problem before Stage III introduces adaptive EMA-generated targets.

For a candidate token sequence $\mathcal{T}'$, we define the length-normalized conditional score

$$\bar{s}(\mathcal{T}' \mid Z; \theta_{\mathrm{AR}}) = \frac{1}{|\mathcal{T}'|} \sum_{l=1}^{|\mathcal{T}'|} \log p_{\theta_{\mathrm{AR}}}(\tau'_l \mid \tau'_{<l}, Z). \tag{11}$$

Length normalization makes candidate comparisons less dominated by sequence length, while preserving sensitivity to the full autoregressive string.

Sequence-likelihood surrogate. For a paired example (x_i, x_i^+), with representations Z_i and Z_i^+, Stage II maximizes the cross-paired scores

$$\bar{s}(\mathcal{T}_i^+ \mid Z_i) \quad \text{and} \quad \bar{s}(\mathcal{T}_i \mid Z_i^+).$$

Thus, each view predicts the token sequence induced from the other content-preserving view. This provides a differentiable surrogate for Eq. 7: paired views should be mutually predictable as ordered strings, without directly optimizing discrete edit distance.

Why begin with a deterministic teacher. Stage I provides learned discrete units, but its sequence construction remains locally defined by nearest-centroid assignment and run-length deduplication. Stage II uses this frozen rule as a deterministic teacher, giving the decoder a stable initial task: predict the paired view's VQ-derived string from the current view's acoustic representation.

This separation is important because Stage III jointly updates the encoder, decoder, conditioning representation, generated strings, and target distribution through EMA self-alignment. Starting directly in that regime would entangle acoustic grounding with target drift, representation drift, and possible collapse to generic strings. Stage II therefore provides a grounded autoregressive prior before the fixed-target constraint is relaxed.

3.3.1 Deterministic Teacher from Frozen Encoder and Nearest-Centroid VQ

In Stage II, the Stage I encoder and VQ codebook are kept fixed. For each paired example, the frozen tokenizer produces deduplicated nearest-centroid strings

$$\mathcal{T}_i = \phi(h(Enc(x_i))), \qquad \mathcal{T}_i^+ = \phi(h(Enc(x_i^+))).$$

These strings serve as deterministic targets for the autoregressive decoder; no encoder or codebook parameters are updated in this stage.

Frozen VQ targets as a stable bridge. The frozen Stage I tokenizer supplies a learned symbolic alphabet and a fixed frame-to-token assignment rule. Its outputs are meaningful VQ-induced units, but their sequence structure is still produced by local nearest-centroid assignment and run-length deduplication rather than by conditional sequence generation. Stage II therefore uses these strings as a stable bridge: the decoder first learns ordered token generation from acoustic conditioning before the targets are allowed to adapt in Stage III.

Cross-paired teacher forcing. We implement Dec_{AR} as a Whisper-style Transformer decoder (Radford et al., 2023), with causal self-attention over previous target tokens and cross-attention to encoder states. For a paired example (x_i, x_i^+), Stage II trains the decoder across views:

$$\bar{s}(\mathcal{T}_i^+ \mid Z_i; \theta_{\mathrm{AR}}) \qquad \text{and} \qquad \bar{s}(\mathcal{T}_i \mid Z_i^+; \theta_{\mathrm{AR}}). \tag{12}$$

In each direction, the acoustic condition is computed from one view and the teacher-forced target string from the other content-preserving view.

This construction lifts the invariance requirement from local frame assignments to ordered symbolic sequences. The decoder is not trained merely to reproduce the tokenization of the same waveform; it must assign high likelihood to a string induced from a perturbed realization of the same segment. Thus, paired views are encouraged to remain mutually predictable as token strings even when nuisance variation perturbs local VQ assignments.

Cross-pairing alone, however, does not guarantee acoustic grounding. Under teacher forcing, the decoder also observes the target-side prefix, which may explain next-token prediction through local continuation statistics while using Z weakly. The next section therefore introduces Stage II regularization so that high likelihood reflects input-conditioned sequence prediction rather than primarily prefix-based continuation.

3.3.2 Preventing Decoder Bypass in Cross-Paired Teacher Forcing

Cross-paired teacher forcing promotes sequence-level invariance, but the likelihood is still computed while the decoder observes a valid target prefix. Because fixed VQ-derived targets have regular local structure, high teacher-forced likelihood does not by itself imply strong use of the acoustic condition Z. This matters because the decoder is the tokenizer: at inference, decoding starts from BOS and the full string must be generated from Z and the model's own previous outputs. Stage II therefore regularizes teacher forcing so that likelihood reflects input-conditioned sequence prediction rather than prefix continuation.

Decoder bypass as the main failure mode. The main failure mode is that the decoder becomes insensitive to Z. Under teacher forcing, next-token prediction can be explained by token-transition statistics; if the BOS-induced state and prefix history dominate, the decoder behaves like an unconditional language model over token strings, emitting common prefixes and continuing them through self-attention with weak dependence on the encoder states.

This produces misleading invariance: acoustically different inputs may map to the same or nearly same generic string, causing Jaccard similarity, edit similarity, and exact-match rate to rise without preserving content-dependent distinctions. We call this *decoder bypass*: the target-token pathway explains the objective while the acoustic pathway that should define the tokenizer is bypassed. Structured self-attention dropout weakens this continuation path and increases prediction pressure on decoder–encoder cross-attention to Z.

Three mechanisms for input grounding. We use three complementary controls. Prefix corruption masks selected teacher-forced prefix tokens while keeping the prediction target clean, reducing the reliability of local token history. Encoder-summary conditioning adds a projected global summary $c(Z)$ to every decoder input, providing a direct input-dependent pathway in addition to cross-attention. Structured self-attention dropout stochastically removes the decoder self-attention residual branch during training, weakening token-side continuation. Together, these controls preserve causal autoregressive training while making high likelihood harder to obtain from the target prefix alone.

Scheduled relaxation of bypass regularization. Decoder-side bypass controls are applied as an annealed curriculum. Prefix corruption and structured self-attention dropout are strongest early and are linearly relaxed over training. Early training therefore makes the target prefix and self-attention continuation path unreliable, encouraging acoustic grounding; later relaxation reduces train–test mismatch as the decoder approaches its full-capacity inference configuration. Stage-specific start and end values are reported in Appendix F.5.

Masked teacher-forcing score. Let $\mathcal{Y} = [y_1, \ldots, y_M, \text{EOS}]$ be the clean target sequence and $\mathcal{U} = [\text{BOS}, y_1, \ldots, y_M]$ the shifted teacher-forcing prefix. We sample a mask set $\mathcal{M}$ over valid prediction positions and construct a corrupted prefix $\widetilde{\mathcal{U}}$ by replacing selected non-special prefix tokens with MASK. Targets remain clean, so the decoder must predict the same next tokens from a partially degraded history, encoder states Z, and summary bias $c(Z)$.

Masking is biased toward early positions because early tokens establish the symbolic trajectory from BOS and can become self-reinforcing during autoregressive decoding. We score masked and unmasked positions separately, obtaining $\bar{s}_m(\mathcal{Y} \mid Z; \theta_{\text{AR}})$ and $\bar{s}_u(\mathcal{Y} \mid Z; \theta_{\text{AR}})$, each normalized over its own valid positions. The final score is

$$\bar{s}_{\text{MTF}}(\mathcal{Y} \mid Z; \theta_{\text{AR}}) = \alpha\, \bar{s}_m(\mathcal{Y} \mid Z; \theta_{\text{AR}}) + (1-\alpha)\, \bar{s}_u(\mathcal{Y} \mid Z; \theta_{\text{AR}}), \qquad \alpha \in [0,1]. \tag{13}$$

Masked positions test recovery when prefix information is unreliable, while unmasked positions retain ordinary teacher-forcing stability. Unless otherwise stated, $\alpha = 0.5$.

Encoder-summary conditioning. Prefix corruption weakens the target-side shortcut, but does not by itself ensure use of the acoustic condition. We therefore add a global conditioning pathway in parallel with cross-attention. Cross-attention provides time-resolved access to Z, while the summary pathway provides an input-dependent acoustic bias at every decoder position, including early steps where the prefix is short.

Given encoder states $Z = [z_1, \ldots, z_T]$, we compute $\bar{z} = \frac{1}{T}\sum_{t=1}^{T} z_t$, or the mask-normalized average when padding is present, and project it to the decoder dimension:

$$c(Z) = W_c\, \text{LayerNorm}(\bar{z}) + b_c, \qquad c(Z) \in \mathbb{R}^{d_{\text{dec}}}.$$

For the Whisper-style decoder, the input at position l is

$$\widetilde{e}_l = E(\widetilde{u}_{l-1}) + p_l + c(Z),$$

where $E(\widetilde{u}_{l-1})$ is the corrupted-prefix token embedding and p_l is the learned positional embedding. The summary is not a prompt token and is not prepended; it is an additive input-dependent bias applied to each decoder input, with scale controlled by layer normalization and projection.

This conditioning is not tied to absolute positional embeddings. Since $c(Z)$ is added to the token stream, the same idea applies to decoders using rotary, relative, ALiBi-style, or other position-aware attention mechanisms (Shaw et al., 2018; Su et al., 2024; Press et al., 2021); positional information is then supplied by the decoder's native positional mechanism.

Structured self-attention dropout. To weaken the autoregressive shortcut directly, we apply structured dropout to the decoder self-attention residual branch during training:

$$h^{(r+1)} = h^{(r)} + g^{(r)} \odot \text{SA}^{(r)}(h^{(r)}) + \text{CA}^{(r)}(h^{(r)}, Z) + \cdots,$$

where $g^{(r)} \sim \text{Bernoulli}(1 - p_{\text{sa}})$. The gate acts on the self-attention residual branch at decoder layer r, while cross-attention to Z remains active. Dropping this branch weakens token-side continuation and routes more predictive information through cross-attention and encoder-summary conditioning.

Regularized Stage II objective. The final Stage II objective is the symmetric cross-paired masked teacher-forcing loss:

$$\mathcal{L}_{\text{StageII}} = -\frac{1}{B}\sum_{i=1}^{B}\Big[\bar{s}_{\text{MTF}}(\mathcal{T}_i^{+} \mid Z_i; \theta_{\text{AR}}) + \bar{s}_{\text{MTF}}(\mathcal{T}_i \mid Z_i^{+}; \theta_{\text{AR}})\Big]. \tag{14}$$

This stage learns an input-conditioned decoder over fixed VQ targets while discouraging teacher-forced likelihood from becoming a prefix-continuation shortcut.

3.4 Stage III: Full-Model EMA Self-Alignment for Adaptive Tokenization

Stage II learns an input-grounded decoder over fixed VQ-derived targets, but the target strings remain determined by the frozen geometric tokenizer. Stage III removes this constraint by replacing the deterministic VQ teacher with an EMA teacher over the full encoder–decoder tokenizer. The encoder and decoder are then optimized jointly, allowing the conditioning space, generated string, output length, EOS placement, and target distribution to co-evolve.

The difficulty is that Stage III is self-referential: the model family both generates the symbolic targets and learns from them. Without additional structure, target generation, target fitting, and negative comparison can drift together, leading to unstable targets, decoder bypass, or many-to-one collapse. We therefore separate these roles. The EMA teacher generates free-running targets at full decoding capacity, the student fits paired targets using the bypass-resistant $\bar{s}_{\mathrm{MTF}}$ score from Stage II, and hard-negative comparison uses clean-prefix likelihoods so that the likelihood matrix measures symbolic confusability rather than robustness to a sampled mask pattern.

Thus, Stage III is not ordinary self-training on decoded strings. It is a controlled refinement step that decouples adaptive target generation, grounded positive fitting, and discriminative negative comparison. This lets PairAlign move beyond frame-derived VQ targets while preserving stable target generation, input grounding, and sequence-level separability.

3.4.1 Full-Model EMA Teacher and Joint Encoder–Decoder Optimization

EMA teacher over the complete tokenizer. Let $\theta = (\theta_{\mathrm{enc}}, \theta_{\mathrm{AR}})$ denote the student parameters. Stage III maintains an EMA teacher $\tilde{\theta}$ updated as

$$\tilde{\theta} \leftarrow \alpha\tilde{\theta} + (1-\alpha)\theta.$$

Unlike the Stage I deterministic teacher, this teacher contains both the encoder and autoregressive decoder. It provides adaptive symbolic targets whose evolution is slower than the student updates, allowing the target distribution to change without training directly on rapidly moving student-generated strings.

Full-capacity EMA target generation. Given a teacher representation $\widetilde{Z}$, the EMA decoder generates a target sequence by free-running autoregressive decoding from BOS. Generation uses the full decoder computation: prefix corruption is disabled, self-attention dropout is disabled, and cross-attention and encoder-summary conditioning remain active. The teacher therefore defines the current adaptive tokenizer, rather than a regularized training-time approximation to it.

To avoid repetition loops, generic starts, and uncontrolled continuation, teacher decoding uses top-p sampling (Holtzman et al., 2019), repetition control, early-step stochasticity, and a sample-specific length cap. For each paired example, the teacher produces

$$\widehat{\mathcal{T}}_i = \mathrm{Sample}(\widetilde{Enc}(x_i); \tilde{\theta}), \qquad \widehat{\mathcal{T}}_i^{+} = \mathrm{Sample}(\widetilde{Enc}(x_i^{+}); \tilde{\theta}).$$

These strings replace the fixed Stage I VQ targets as adaptive self-alignment targets. Sampling details are given in Section 3.4.3.

Bypass-resistant student fitting. The EMA teacher generates targets with the full decoder, while the student fits positive pairs with $\bar{s}_{\mathrm{MTF}}$. Prefix corruption and structured self-attention dropout remain active for this fitting step, as in Stage II, so positive alignment cannot be solved only through teacher-forced prefix continuation. Encoder-summary conditioning remains part of the decoder throughout Stages II–III, keeping the acoustic condition directly available at every prediction step. Thus, even with adaptive targets, fitting remains an input-grounded cross-paired sequence-prediction problem.

Stable joint refinement. Stage III updates both encoder and decoder, but the encoder also defines the conditioning space used for teacher generation and student scoring. To reduce simultaneous drift in representations, targets, and likelihood scores, we update the encoder more conservatively than the decoder:

$\eta_{\text{enc}} = 0.1\,\eta_{\text{dec}}$. This allows the full tokenizer to move beyond the fixed Stage I geometry while limiting instability in the moving target distribution.

Cross-paired positive alignment. Using student representations Z_i and Z_i^+, Stage III transfers teacher-generated tokenizations across paired views:

$$\mathcal{P}_{ii} = \bar{s}_{\text{MTF}}(\widehat{\mathcal{T}}_i^+ \mid Z_i; \theta), \qquad \mathcal{P}_{ii}^+ = \bar{s}_{\text{MTF}}(\widehat{\mathcal{T}}_i \mid Z_i^+; \theta). \tag{15}$$

The first term asks whether x_i supports the teacher tokenization of x_i^+; the second applies the same criterion in reverse. These terms retain the Stage II cross-view prediction principle, but the targets are now adaptive EMA-generated strings rather than fixed VQ-derived strings.

Uncorrupted likelihoods for negative comparison. Positive fitting uses $\bar{s}_{\text{MTF}}$ because paired-target alignment should remain bypass-resistant. Negative comparison has a different role: it should measure which teacher-generated strings are genuinely confusable under a given acoustic condition. We therefore compute the hard-negative likelihood matrix with clean teacher-forced prefixes, using the length-normalized score $\bar{s}(\cdot \mid Z)$.

Here, "uncorrupted" refers only to the teacher-forced prefix. Candidate strings are scored with clean prefixes so that confusability does not depend on a sampled mask pattern. The decoder architecture and conditioning pathways are otherwise unchanged, and structured self-attention dropout follows the training schedule. The resulting matrix tests whether each acoustic condition prefers its paired EMA-generated string over the most confusable mismatched strings in the minibatch.

3.4.2 Preventing Tokenization Collapse with In-Batch Likelihood Contrast

Positive self-alignment alone does not ensure a discriminative token space. A collapsed tokenizer can make paired views mutually predictable while also making unrelated examples highly likely. Stage III therefore adds sequence-level discriminative pressure: under each conditioning representation, the paired teacher tokenization should outrank the most confusable teacher tokenizations from other examples in the minibatch.

Likelihood matrix for symbolic confusability. For negative comparison, we construct likelihood matrices using the clean-prefix score $\bar{s}(\cdot \mid Z)$ from Eq. 11. The matrix measures relative symbolic preference: given an acoustic condition, does the model assign higher likelihood to the paired teacher string than to mismatched teacher strings?

For the forward direction,

$$M_{ij}^{\rightarrow} = \bar{s}(\widehat{\mathcal{T}}_j^+ \mid Z_i; \theta), \qquad i, j \in \{1, \ldots, B\}. \tag{16}$$

Row i fixes the condition Z_i, and column j provides a candidate positive-view teacher string. The diagonal is the paired target; off-diagonal entries are mismatched alternatives. The reverse matrix $M^{\leftarrow}$ is defined analogously by exchanging anchor and positive views. In both directions, the desired behavior is row-wise separation: the diagonal should dominate the off-diagonal entries.

Hardest-K confusable negatives. For each row, we retain the K largest off-diagonal entries:

$$\mathcal{H}_K^{\rightarrow}(i) = \text{TopK}\left(\{M_{ij}^{\rightarrow} : j \neq i\}, K\right), \qquad \mathcal{H}_K^{\leftarrow}(i) = \text{TopK}\left(\{M_{ij}^{\leftarrow} : j \neq i\}, K\right). \tag{17}$$

These are the mismatched teacher strings that currently score highest under the wrong acoustic condition. They are the relevant negatives for collapse prevention: high likelihood for these strings indicates loss of input-dependent discriminability.

Row-wise contrast over sequence likelihoods. We apply a row-wise contrastive loss between the paired target and its hardest mismatched competitors. For the forward direction,

$$\mathcal{L}_{\text{NCE}}^{\rightarrow} = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{\exp(M_{ii}^{\rightarrow}/\tau_\ell)}{\exp(M_{ii}^{\rightarrow}/\tau_\ell) + \sum_{j \in \mathcal{H}_K^{\rightarrow}(i)} \exp(M_{ij}^{\rightarrow}/\tau_\ell)}, \tag{18}$$

where $\tau_\ell > 0$ is a temperature over likelihood scores. The reverse loss $\mathcal{L}^{\leftarrow}_{\text{NCE}}$ applies the same row-wise contrast to $M^{\leftarrow}$ and $\mathcal{H}^{\leftarrow}_{K}(i)$, giving

$$\mathcal{L}_{\text{NCE}} = \mathcal{L}^{\rightarrow}_{\text{NCE}} + \mathcal{L}^{\leftarrow}_{\text{NCE}}.$$

This term complements positive alignment: paired views must be mutually predictable, but mismatched teacher strings must not become equally likely under the same condition.

Entropy regularization. We also add a negative-entropy penalty on the student next-token distribution, averaged over batch elements and decoder steps. Minimizing this term discourages early over-confidence and concentration on generic high-prior tokens, helping preserve broad token usage during EMA self-alignment.

Final Stage III objective. The positive self-alignment loss is

$$\mathcal{L}_{\text{pos}} = -\frac{1}{B}\sum_{i=1}^{B}\left(\mathcal{P}_{ii} + \mathcal{P}^{+}_{ii}\right), \tag{19}$$

where $\mathcal{P}_{ii}$ and $\mathcal{P}^{+}_{ii}$ are the cross-paired positive scores computed with $\bar{s}_{\text{MTF}}$. The full objective is

$$\mathcal{L}_{\text{StageIII}} = \mathcal{L}_{\text{pos}} + \lambda_{\text{NCE}}\left(\mathcal{L}^{\rightarrow}_{\text{NCE}} + \mathcal{L}^{\leftarrow}_{\text{NCE}}\right) + \lambda_{\text{entropy}}\mathcal{L}_{\text{entropy}}. \tag{20}$$

The terms have distinct roles: positive alignment transfers adaptive teacher tokenizations across paired views, NCE prevents many-to-one collapse by ranking paired strings above hard mismatches, and entropy regularization preserves predictive diversity.

From fixed VQ targets to adaptive symbolic self-alignment. Stage II trains on stationary VQ-derived strings. Stage III replaces them with adaptive EMA-generated tokenizations and jointly refines the encoder–decoder tokenizer. The transition is controlled by separating three computations: full-capacity EMA target generation, bypass-resistant paired fitting, and clean-prefix likelihood contrast against confusable mismatches. The resulting token sequence is no longer restricted to deduplicated nearest-centroid assignment; it becomes an adaptive symbolic string shaped by cross-view predictability and constrained by in-batch discriminability.

3.4.3 Autoregressive Target Generation and Decoding

PairAlign uses the same autoregressive tokenizer in two modes. During Stage III, the EMA teacher generates adaptive self-alignment targets with stochastic top-p sampling. After training, the final tokenizer is decoded deterministically with beam search. This separates training-time target exploration from evaluation-time tokenization stability.

EMA-teacher targets. For an input segment x, the EMA teacher encoder produces $\widetilde{Z} = \widetilde{Enc}(x)$, and the EMA decoder generates a variable-length sequence autoregressively from BOS. The target is therefore not an external label or a fixed VQ string; it is the current EMA tokenizer's own input-conditioned prediction. During Stage III, targets are sampled with nucleus sampling (Holtzman et al., 2019). To keep this free-running process useful, teacher decoding combines repetition control, position-dependent sampling parameters, and an explicit length cap to avoid generic prefixes, short loops, and unbounded continuation.

Repetition control and sampling schedule. Before sampling or beam expansion, repeated non-special tokens receive a count-dependent multiplicative penalty. If token a has appeared c_a times in the current partial sequence, its next-token probability is downweighted by a factor proportional to $\gamma_{\text{rep}}^{-c_a}$, or equivalently its logit receives a count-dependent negative shift. Unseen tokens are unchanged, and BOS, EOS, and PAD are excluded from the count. The penalty only affects next-token selection; it does not edit the generated sequence after decoding. During teacher sampling it shapes the stochastic top-p distribution, while during inference it shapes deterministic beam-search scores.

Teacher sampling then applies nucleus filtering and temperature scaling. The decoder keeps the smallest token set whose cumulative probability exceeds the top-p threshold, preserves the highest-probability token,

and samples from the filtered distribution. Early steps use higher temperature, larger top-p, and weaker repetition penalty because they establish the trajectory from BOS; later steps use sharper settings to stabilize continuation.

Length-constrained EOS handling. The decoded tokenization is not required to emit one token per encoder frame. The decoder learns when to emit EOS, so output length is adaptive. Free-running decoding is bounded by $L_{\max}(x) = \lfloor \rho T_x \rfloor$, where T_x is the valid encoder length and $\rho \in [0, 1]$ controls the maximum token rate. If EOS is emitted before the cap, decoding stops normally; otherwise EOS is forced at the cap. This prevents unbounded generation and controls maximum symbolic bitrate, but it is not a supervised length target and does not use transcript, boundary, or Stage I VQ length information.

Inference-time beam search. After training, tokenization uses beam search rather than stochastic sampling. Given encoder representation Z, decoding starts from BOS and maintains K partial hypotheses until EOS is emitted or the length cap is reached. Inference uses the full decoder: prefix corruption and self-attention dropout are disabled, while the standard conditioning pathways remain active. Repetition control and the length cap are retained. The final tokenization is the highest-scoring completed beam, with K controlling search width, ρ the length cap, γ_{rep} the repetition penalty, and α_{len} optional length normalization.

Summary. Stage III uses stochastic EMA-teacher sampling to obtain adaptive, input-conditioned targets without external symbolic supervision. Final tokenization uses deterministic beam search to produce stable symbolic strings. Pseudocode for teacher sampling and beam-search decoding is provided in Appendices C and D.

3.5 Extracting Timing Information from Autoregressive Decoding

Autoregressive tokenization produces a compact symbolic sequence rather than a frame-synchronous label stream. This supports storage and sequence-level comparison, but removes native token-to-frame timing. When approximate temporal grounding is needed, we extract timestamps from decoder cross-attention after decoding. This is an inference-time post-processing step: it does not modify the training objective, constrain the decoder, or backpropagate through recovered alignments.

The procedure uses monotonicity only as a weak post-hoc bias. Speech-to-symbol correspondence is expected to be approximately ordered in time, and prior work has incorporated this bias directly through local, Gaussian, hard monotonic, monotonic multi-head, location-aware, and attention-prior-guided mechanisms (Tjandra et al., 2017; Hou et al., 2017; Merboldt et al., 2019; Wu & Cotterell, 2019; Ma et al., 2019; Chowdhury & Caragea, 2023; Zhao et al., 2020; Neekhara et al., 2024). PairAlign does not impose such constraints during training; monotonicity is used only to interpret the trained decoder's cross-attention.

Given encoder latents Z and a decoded sequence, we extract decoder–encoder cross-attention from a selected layer or from an average over layers and heads. After removing BOS, EOS, decoder padding, and padded encoder frames, we obtain $W \in \mathbb{R}^{L \times T}$, where L is the number of decoded non-special tokens and T is the number of valid encoder frames. Because raw attention can be diffuse or multi-modal, we apply a weak diagonal 2D Beta-shaped prior inspired by beta-binomial attention guidance (Neekhara et al., 2024). There, the prior is applied to cross-attention during training with an alignment loss to encourage monotonicity in speech-LLM generation. Here, it is used only after decoding to sharpen monotonic structure already present in the learned attention.

The prior-reweighted token-to-frame attention is renormalized across decoded positions for each encoder frame, yielding a frame-to-token posterior $A \in \mathbb{R}^{T \times L}$. We then decode a monotone Viterbi path in which each frame either remains at the current token or advances to the next token, producing a contiguous monotone token-to-frame segmentation. The frame support assigned to each token is converted to start and end times using the window start time, sampling rate, and encoder downsampling factor, giving $\{(u_l, t_l^{\text{start}}, t_l^{\text{end}})\}_{l=1}^{L}$.

The recovered timestamps should be interpreted as approximate temporal grounding for learned token positions, not supervised boundaries. Unlike CTC, RNN-T, monotonic attention, monotonic chunkwise attention, monotonic multi-head attention, or training-time attention-prior methods (Graves et al., 2006; Graves,

2012; Raffel et al., 2017; Chiu & Raffel, 2017; Ma et al., 2019; Wu & Cotterell, 2019; Sak et al., 2017; Zhao et al., 2020; Neekhara et al., 2024), this procedure introduces no alignment latent variable and makes no training-time change. Its reliability depends on the structure of the learned cross-attention; diffuse or non-monotone attention yields less reliable timestamps. We therefore treat timing extraction as an auxiliary diagnostic rather than a core training objective. The full algorithm, quantitative diagnostics, and raw versus prior-reweighted visualizations are given in Appendix E.

4 Experiments

Continuous speech as the operating regime. PairAlign is designed for continuous speech rather than isolated lexical crops. Accordingly, all learned tokenizers and SSL-tokenization baselines use LibriSpeech-trained representations, with no training or adaptation on cross-corpus or cross-lingual evaluation data. We use fixed 3 s speech windows as the primary regime. Very short word-level crops provide too little conditioning context for autoregressive tokenization: in our LibriSpeech word-crop extraction, all retained words were shorter than 0.5 s, and this under-constrained setting led to severe generic-string collapse. A 3 s window provides enough phonetic, speaker, prosodic, and local lexical context for input-grounded decoding, while remaining tractable for cross-attention, teacher-forced likelihoods, EMA-teacher decoding, and in-batch likelihood contrast. This duration is also consistent with codec language-modeling systems that use 3 s of speech context to infer speaker, acoustic, and prosodic information (Borsos et al., 2023; Wang et al., 2023).

Evaluation questions. We evaluate whether cross-paired conditional likelihood induces useful symbolic sequences beyond the training objective. The experiments test five properties: *consistency*, whether augmented views of the same 3 s segment produce compatible strings; *non-collapse*, whether agreement comes from broad, input-dependent vocabulary use rather than generic strings; *retrieval*, whether edit-distance structure supports search over continuous-speech archives; *local sequence evolution*, whether strings change coherently under 100 ms window shifts; and *lexically disjoint transfer*, whether English-trained tokenizers preserve similarity structure on Tamil speech from Shrutilipi (Bhogale et al., 2023). Together, these axes test whether PairAlign learns compact, diverse, input-grounded, and searchable symbolic sequences, while probing whether the induced similarity relation reflects acoustic-symbolic structure beyond English lexical identity.

Evaluation domains. We use three corpora to separate in-domain behavior, cross-corpus English generalization, and lexically disjoint linguistic shift. PairAlign is trained on LibriSpeech `train-clean-360` and `train-other-500`, an 860 h English pool, with model selection on held-out LibriSpeech `test-clean` (Panayotov et al., 2015). In-domain consistency and inventory diagnostics are evaluated on LibriSpeech `train-clean-100`, using speakers and utterances outside the optimization splits.

TIMIT is the main cross-corpus English testbed (Garofolo et al., 1993). It differs from LibriSpeech in speakers, recording conditions, utterance structure, and phonetic coverage, and is used for token consistency, continuous-audio retrieval, and continuous-sweep analysis. The retrieval archive contains approximately 3.94 h of unique speech, segmented into overlapping 3 s windows with a 1.5 s hop; sweep analysis uses a finer 100 ms hop to probe local sequence evolution.

Shrutilipi-Tamil provides the lexically disjoint cross-linguistic probe (Bhogale et al., 2023). Tamil differs from English in lexical inventory, morphology, phonotactics, prosody, and canonical word order (Krishnamurti, 2003; Dryer & Haspelmath, 2013; Steever, 2017). We use it to test whether English-trained tokenizers preserve anchor–positive consistency, token-space retrieval structure, and native-position inventory behavior under linguistic shift. The Tamil retrieval archive contains approximately 10.00 h of unique speech with the same 3 s window and 1.5 s hop protocol.

Training and paired-view protocol. All stages use the same continuous-audio paired-view regime. Training examples are 3.0 s speech segments sampled from LibriSpeech utterances. Longer utterances are cropped; shorter or boundary-adjacent segments are padded only when needed. Padded regions are masked and ignored during encoder conditioning, tokenization, and loss computation. Silence-aware cropping avoids segments that are mostly non-speech.

Each anchor x_i is paired with an augmented positive view $x_i^+ = \mathrm{Aug}(x_i)$. Augmentations preserve spoken content while changing nuisance factors through conservative gain perturbation, additive noise, filtering, reverberation, short time masking, and mild time stretching. Because time stretching changes duration, the positive view can be shorter or longer than the 3.0 s anchor and need not be frame-aligned. Any padding introduced for batching is masked out in conditioning states, token targets, and loss terms.

This variable-duration construction is compatible with PairAlign because the model and losses do not require time-aligned anchor and positive representations. The data protocol is fixed across stages; only the supervision changes. Stage I learns a geometric VQ tokenizer, Stage II learns an autoregressive bridge to fixed VQ-derived strings, and Stage III replaces those strings with adaptive EMA-teacher tokenizations and in-batch likelihood contrast.

Baselines. The primary controlled baseline is the Stage I geometric tokenizer: the same front-end encoder followed by nearest-centroid VQ and consecutive-token deduplication. This isolates the effect of autoregressive sequence generation, paired self-alignment, EMA-teacher refinement, and learned length control. We also include Stage I+, a wav2tok-style extension that adds sequence-consistency pressure while retaining frame-synchronous geometric token induction (Banerjee & Arora, 2022). Finally, HuBERT+VQ and WavLM+VQ discretize pretrained SSL representations using nearest-centroid VQ (Hsu et al., 2021; Chen et al., 2022). For both, we use layer 6 features and deduplicate the resulting streams before edit-distance comparison, so PairAlign is compared against compacted frame-derived symbolic sequences rather than raw frame traces.

Discrete token consistency. We compare token sequences produced from an anchor and its augmented positive view. This directly tests whether content-preserving acoustic variants map to compatible strings. We report unigram Jaccard similarity, normalized edit similarity, exact match, mean sequence length, and edit-operation decomposition. Jaccard measures token-identity reuse; edit similarity additionally accounts for order, substitutions, insertions, deletions, and length mismatch. The edit-operation decomposition distinguishes relabeling from token birth or deletion. Metric definitions are provided in Appendix F.1.

Token inventory and collapse. Consistency and compactness are meaningful only if they are non-degenerate. We therefore measure vocabulary usage, active and dead vocabulary, normalized entropy, effective vocabulary size, top-token concentration, native-position and length-normalized position-wise token usage, low-diversity collapse, and within-stream exact collisions. A sequence is marked low-diversity collapsed when $r_{\mathrm{uniq}}(\mathcal{T}) = |\mathrm{unique}(\mathcal{T})|/|\mathcal{T}| \leq 0.2$, and the collapsed-pair rate counts anchor–positive pairs where either sequence collapses. Within-stream exact collision measures many-to-one assignment across distinct examples. Definitions are provided in Appendix F.2.

Continuous-audio retrieval. We test search over approximately 3.94 h of continuous TIMIT speech. The archive is clean: unaugmented speech is segmented into overlapping 3 s windows with a 1.5 s hop within recording boundaries. We sample 300 clean archive segments as query sources, apply the same content-preserving augmentations used for consistency, and search only the clean archive. Queries and archive segments are tokenized independently and ranked entirely in token space using normalized edit distance. Forced-alignment metadata is used only to define phoneme-based relevance, not for tokenization or ranking.

We use three relevance definitions. *Segment-overlap relevance* marks an archive segment relevant if its clean interval has any positive overlap with the clean query-source segment; with 3 s windows and 1.5 s hop, this includes the source and, away from boundaries, the neighboring overlapping windows. *Phoneme-exact relevance* requires identical silence-filtered aligned phoneme strings. *Phoneme-relaxed relevance* allows normalized phoneme Levenshtein distance at most 0.35.

We report Recall@K, MRR, and mean first relevant rank. We also report archive compactness: total tokens, average tokens per segment, token rate, token-count compression, relative token reduction, minimum and maximum segment length, and serialized archive size. These quantify symbolic storage and comparison cost, not audio-codec bitrate. Definitions are provided in Appendix F.3.

Continuous-sweep tokenization. We probe local symbolic structure by sweeping a 3 s TIMIT window through speech in 100 ms steps. Neighboring windows share 2.9 s of audio and differ only at the entering or leaving boundary. For each neighboring pair, we compare independently decoded strings using normalized edit similarity, unigram Jaccard similarity, sequence length, token rate, $|\Delta L|$, length ratio, relative length change, edit distance, edit-operation counts, and length-normalized edit operation rates. The probe asks whether small context changes induce bounded symbolic updates rather than arbitrary re-tokenization.

The criterion is deliberately weaker than exact shift-equivariance. None of the tokenizers is trained with an externally defined temporally grounded alphabet that would force a 100 ms shift to preserve an exact interior subsequence, e.g., $(a, b, c, d) \rightarrow (b, c, d, e)$. Such behavior would require stronger segmental supervision or boundary-localized symbolic units. Here, tokens may summarize multiple subsegments, split regions across symbols, relabel under context change, or vary with the surrounding window.

We therefore interpret sweep results in a granularity-aware way. Dense frame-derived tokenizers are expected to show stronger fine-grained persistence because many frame assignments are reused across overlapping windows. Compact autoregressive tokenizers may operate at a coarser symbolic scale, with fewer tokens and more context per token. Strong behavior under this probe means bounded symbolic updates, controlled token-count variation, and non-degenerate token usage, not zero edits or strict subsequence tracking. Sweep-specific definitions are given in Appendix F.4.

Lexically disjoint linguistic shift. The Tamil experiments diagnose the similarity relation induced by each tokenizer; they are not component ablations or tests of word-level cross-lingual correspondence. Tamil is Dravidian, while English is Indo-European, and the languages differ in lexical inventory, morphology, phonotactics, prosody, and word order (Krishnamurti, 2003; Dryer & Haspelmath, 2013; Steever, 2017). In particular, Tamil is predominantly SOV and agglutinative, so a 3 s Tamil window can differ from an English window in both acoustic-phonetic and short-range morphosyntactic organization.

We ask whether English-trained tokenizers preserve useful acoustic-symbolic similarity under this shift. We evaluate anchor–positive consistency, token-space retrieval over a Tamil archive, and native-position inventory behavior. The first tests paired-view stability, the second tests edit-distance search, and the third measures whether position-wise token usage shifts smoothly or becomes unstable. For inventory analysis, we compare position-wise token entropy between English and Tamil at each tokenizer's native output scale and summarize entropy residuals as distributional shift.

Ablations. We include three targeted ablations. *Stage II AR Prior* evaluates the deterministic-teacher autoregressive model before EMA self-alignment, testing whether fixed VQ-derived targets are sufficient without adaptive tokenization. *Stage II w/o structured self-attention dropout* removes the dropout on the decoder self-attention residual branch, testing whether teacher-forced decoding can rely on token-prefix continuation rather than acoustic condition. *Full PairAlign w/o encoder-summary conditioning* removes the global summary pathway while retaining cross-attention, testing whether cross-attention alone provides sufficient grounding. This variant is trained without the summary pathway in both Stage II and Stage III to avoid a stage-wise architecture mismatch.

Architecture and training summary. PairAlign is trained on repeated 3.0 s windows from LibriSpeech `train-clean-360` and `train-other-500`. Stage I uses 96-bin log-Mel inputs, a Mamba2-based encoder producing 512-dimensional frame representations, and a 512-entry nearest-centroid VQ codebook. Stages II–III use the same 512-symbol alphabet plus BOS, EOS, PAD, and MASK, with a Whisper-style autoregressive Transformer decoder of hidden dimension 512 and vocabulary size 516. Both EMA-teacher sampling and inference-time beam search use length-cap ratio $\rho = 0.15$.

Stage I uses approximately 667 sampled anchor-segment hours, while Stages II and III each use approximately 400 sampled anchor-segment hours from the same finite 860 h LibriSpeech pool. These are sampled segment-hours, not unique audio hours. HuBERT and WavLM have substantially larger upstream pretraining exposure, so we treat HuBERT+VQ and WavLM+VQ as strong pretrained SSL tokenization references rather than compute-matched baselines.

Dataset	Model	Jaccard Similarity	Edit Similarity	Exact Match Rate	Mean Sequence Length	Active Vocabulary Size
LibriSpeech-100	Stage I Geometric	0.718	0.609	0.264	92.09	512
	Stage I+ Geometric	0.738	0.629	0.265	75.61	512
	PairAlign	0.719	0.630	0.291	35.55	512
TIMIT	Stage I Geometric	0.742	0.616	0.267	78.65	456
	Stage I+ Geometric	0.750	0.643	0.267	58.79	420
	PairAlign	0.753	0.691	0.301	26.19	430

Table 1: Discrete token consistency and compactness on LibriSpeech-100 and TIMIT. Each model independently tokenizes an anchor 3 s segment and a noisy content-preserving positive view. Jaccard Similarity measures unordered unigram token-set overlap. Edit Similarity measures normalized Levenshtein similarity. Exact Match Rate is the fraction of anchor–positive pairs with identical token strings. Mean Sequence Length measures symbolic compactness. Active Vocabulary Size measures how many codebook entries are used.

The autoregressive stages include controls against decoder bypass and collapse: prefix masking, structured self-attention dropout, hard in-batch likelihood contrast, repetition-aware teacher generation, scheduled top-p teacher sampling, explicit length constraints, and separate encoder–decoder learning rates. Training uses Adam with learning rate 5×10^{-5}, weight decay 10^{-4}, and a learning-rate scheduler; when updated, the encoder uses a smaller learning rate than the decoder to preserve acoustic geometry while the autoregressive tokenizer adapts. Full details are given in Appendix F.5.

How to read the experiments. The experiments are complementary. Consistency measures robustness to nuisance variation; inventory diagnostics test whether agreement and compactness are non-degenerate; retrieval evaluates edit-distance search over continuous speech; archive statistics quantify symbolic storage and comparison cost; continuous sweep probes local sequence evolution under small temporal shifts; and Tamil transfer tests whether the induced similarity relation survives a lexically disjoint linguistic shift. Together, these axes evaluate the central claim: PairAlign learns substantially more compact symbolic sequences while preserving enough ordered structure for edit-distance comparison, retrieval, and structural analysis.

5 Results and Discussion

5.1 Probing Consistency, Collapse, and Token-Inventory Usage in Learned Token Sequences

We first test whether the tokenizers produce stable, compact, and non-degenerate symbolic sequences under content-preserving perturbations. Each anchor is a 3 s segment and the positive view is an augmented version of the same segment, so agreement measures robustness to nuisance variation. We compare Stage I Geometric, Stage I+ Geometric, and PairAlign on LibriSpeech-100 and TIMIT. Table 1 reports paired-view consistency, compactness, and active vocabulary size; Table 2 reports low-diversity and exact-collision collapse diagnostics. Supporting edit-operation, vocabulary-usage, native-position entropy, and relative-position entropy diagnostics are provided in Appendix G.

Stage I+ is a strong geometric intermediate. Table 1 shows that the wav2tok-style sequence-consistency constraint improves the geometric tokenizer while retaining frame-synchronous token induction. Relative to Stage I, Stage I+ increases edit similarity from 0.609 to 0.629 on LibriSpeech-100 and from 0.616 to 0.643 on TIMIT, while reducing mean length from 92.09 to 75.61 and from 78.65 to 58.79, respectively. Thus, Stage I+ is not a weak baseline: it already injects order-sensitive pairwise pressure into a geometric tokenizer. PairAlign tests the stronger hypothesis that the token string itself should be generated and self-aligned as a sequence, rather than remain a deduplicated stream of frame-derived assignments.

PairAlign shifts the operating point to compact ordered consistency. PairAlign moves tokenization to a substantially lower-rate regime, with mean length 35.55 on LibriSpeech-100 and 26.19 on TIMIT.

Dataset	Model	Anchor Low-Diversity Sequence Rate	Positive Low-Diversity Sequence Rate	Collapsed Pair Rate	Anchor Exact-Collision Rate	Positive Exact-Collision Rate
LibriSpeech-100	Stage I Geometric	0.0269	0.0381	0.0500	0.0000	0.0000
	Stage I+ Geometric	0.0263	0.0329	0.0450	0.0000	0.0000
	PairAlign	0.0000	0.0000	0.0000	0.0001	0.0001
TIMIT	Stage I Geometric	0.0240	0.0719	0.0803	0.0000	0.0000
	Stage I+ Geometric	0.0136	0.0223	0.0290	0.0000	0.0000
	PairAlign	0.0000	0.0000	0.0000	0.0004	0.0028

Table 2: Collapse behavior of the tokenizers. Low-Diversity Sequence Rate is the fraction of sequences whose unique-token ratio is at most 0.2. Collapsed Pair Rate is the fraction of anchor–positive pairs in which either sequence is low-diversity collapsed. Exact-Collision Rate is the fraction of sequences in a stream that exactly duplicate another sequence from a different example in the same stream. Low-diversity collapse detects repetitive internal degeneration, whereas exact-collision collapse detects many-to-one assignment across examples.

This corresponds to roughly 61% and 67% reductions relative to Stage I, and 53% and 55% relative to Stage I+, respectively. Despite this compression, PairAlign preserves or improves the order-sensitive metric: edit similarity is 0.630 on LibriSpeech-100 and 0.691 on TIMIT, compared with 0.629 and 0.643 for Stage I+.

The Jaccard scores should be read at the correct granularity. Dense geometric tokenizers retain many local frame-derived symbols, which can inflate unigram overlap under augmentation. PairAlign emits far fewer tokens, so each identity change has larger effect on unordered overlap. The main result is therefore not maximal overlap on every metric, but stronger ordered compatibility at a much lower symbolic rate, with the clearest cross-corpus gain on TIMIT.

Higher exact agreement is not explained by collapse. Tables 1 and 2 separate robustness from degeneracy. PairAlign increases exact string match to 0.291 on LibriSpeech-100 and 0.301 on TIMIT, compared with 0.264–0.267 for the geometric systems. This agreement is not explained by low-diversity strings: under the criterion $r_{\text{uniq}} \leq 0.2$, PairAlign has zero measured collapsed-pair rate on both datasets, whereas Stage I has rates 0.0500 and 0.0803, and Stage I+ reduces them to 0.0450 and 0.0290.

Exact-collision rates test many-to-one assignment across distinct examples. PairAlign remains near zero, with anchor/positive rates 0.0001/0.0001 on LibriSpeech-100 and 0.0004/0.0028 on TIMIT. The higher TIMIT positive-stream rate indicates that augmentation can make some compact strings more confusable, but the absolute rate remains very small. Thus, PairAlign's higher exact anchor–positive agreement is not accompanied by broad many-to-one collapse.

Inventory, edit-operation decomposition, and position-wise diagnostics rule out trivial compactness. Appendix G shows that PairAlign's compactness is not explained by unstable edits, narrow vocabulary use, generic prefixes, or many-to-one assignment. First, edit-operation decomposition shows that mean anchor–positive edit distance falls from 39.25 for Stage I and 31.17 for Stage I+ to 14.58 for PairAlign on LibriSpeech-100, and from 33.72 and 23.90 to 9.54 on TIMIT. The reduction occurs across substitutions, insertions, and deletions, indicating a shorter but still aligned symbolic trajectory rather than a strategy that merely suppresses token birth/death.

The residual profile is also well behaved. Substitutions remain the dominant residual operation, as expected under content-preserving acoustic perturbations: segment extent is mostly preserved, but token identities may change under nuisance variation. On TIMIT, substitutions decrease from 18.36 and 11.81 to 4.37, insertions from 8.57 and 5.72 to 1.77, and deletions from 6.79 and 6.36 to 3.39 for Stage I, Stage I+, and PairAlign respectively.

Second, global inventory diagnostics show broad token usage despite the lower rate. On LibriSpeech-100, all systems activate the full 512-token alphabet. PairAlign has slightly lower normalized entropy, as expected for a shorter autoregressive sequence, but entropy remains high and top-10 token mass stays below 0.09. On

Relevance definition	Model	R@1	R@5	R@10	R@20	MRR	Mean FRR
Segment overlap	Geometric	0.75	0.83	0.85	0.87	0.78	36.40
	PairAlign	0.71	0.79	0.80	0.82	0.74	53.99
Phoneme exact	Geometric	0.75	0.83	0.85	0.86	0.78	48.22
	PairAlign	0.71	0.78	0.80	0.82	0.74	73.40
Phoneme relaxed	Geometric	0.75	0.83	0.85	0.87	0.78	38.55
	PairAlign	0.71	0.79	0.80	0.84	0.74	48.95

Table 3: Retrieval comparison on TIMIT (Garofolo et al., 1993) between the Stage I geometric tokenizer and PairAlign. R@K denotes Recall@K, MRR denotes Mean Reciprocal Rank, and FRR denotes First Relevant Rank. Median first relevant rank is 1 for both models under all three relevance definitions, and HitRate is 1.0 throughout; these quantities are omitted for compactness.

TIMIT, PairAlign is broader than the geometric systems: normalized entropy increases to 0.813 and effective vocabulary to approximately 160, compared with 0.786/135 for Stage I and 0.766/119 for Stage I+.

Third, native-position and relative-position entropy diagnostics show structured trajectory formation rather than persistent low-entropy prefix collapse. All three tokenizers exhibit an early-position entropy ramp: token usage is more constrained at the first positions, becomes more diverse over the early trajectory, and then plateaus at a broadly similar entropy level. This indicates that position-dependent structure is not unique to PairAlign. PairAlign shows the strongest early constraint because decoding starts from BOS and the first emitted tokens anchor the autoregressive trajectory. However, entropy broadens after the initial positions and remains distributed across the beginning, middle, and end of the normalized sequence. Together with broad vocabulary usage, zero measured low-diversity collapse, near-zero exact-collision rates, and nontrivial paired-view edit consistency, these diagnostics support the interpretation that PairAlign learns compact input-conditioned token trajectories rather than generic prefixes or collapsed codes.

Overall interpretation. PairAlign changes the operating regime from dense frame-derived overlap to compact ordered symbolic consistency. Stage I and Stage I+ are strong geometric baselines because they retain many local frame assignments and produce long deduplicated streams. PairAlign instead learns length, order, and termination through autoregressive self-alignment. It does not maximize every overlap metric; dense tokenizers have an expected advantage on unigram overlap and local persistence. The central result is that PairAlign produces much shorter, input-grounded, broad-vocabulary strings that preserve order-sensitive edit consistency, remove measured low-diversity collapse, maintain near-zero many-to-one collision rates, and remain stable under cross-corpus evaluation.

5.2 Probing Edit-Distance Retrieval Under Geometric and Self-Aligned Tokenization

Retrieval setup. We next test whether PairAlign's lower-rate symbolic space remains useful for cross-corpus retrieval. Both tokenizers are trained on LibriSpeech and evaluated on the same clean TIMIT archive constructed from approximately 3.94 hours of speech. The archive contains overlapping 3 s windows extracted with a 1.5 s hop within recording boundaries. We sample 300 clean archive segments as query sources and apply the same content-preserving augmentation pipeline used in the consistency experiments; the augmented query waveform is not inserted into the archive. Ranking is performed entirely in token space using normalized edit distance. Retrieval accuracy is reported in Table 3, and symbolic archive compactness is reported in Table 4.

Two symbolic rates. This comparison intentionally contrasts two tokenization regimes rather than two tokenizers of identical granularity. The Stage I geometric tokenizer emits a dense deduplicated trace derived from frame-level nearest-centroid assignments, whereas PairAlign emits a shorter autoregressive sequence whose length and ordering are learned through self-alignment. This difference in symbolic rate is central to retrieval. A higher bitrate stores more local acoustic detail and creates more opportunities for edit-distance matching to discriminate among nearby candidates. PairAlign therefore gives up some rank-level discrimination in exchange for a substantially smaller symbolic search space.

Archive statistic	Geometric	PairAlign
Total tokens N_{tok}	800,611	360,723
Average tokens / segment $\overline{L}$	84.62	38.13
Token rate R_{tok}	28.21 tok/s	12.71 tok/s
Token-count compression C_{tok}	–	2.22×
Relative token reduction r_{red}	–	54.94%
Minimum tokens / segment	12	5
Maximum tokens / segment	154	44
Archive cache size	5.90 MB	4.85 MB

Table 4: Archive compactness comparison on TIMIT (Garofolo et al., 1993) between the Stage I geometric tokenizer and PairAlign. Both tokenizers use a vocabulary of size $|\mathcal{A}| = 512$.

Rate–precision trade-off. Table 3 shows the expected rate–precision trade-off. The geometric tokenizer is sharper at the top of the ranking: across all three relevance definitions, Recall@1 is 0.75 for the geometric tokenizer and 0.71 for PairAlign, while MRR is 0.78 versus 0.74. The same ordering holds at Recall@5, Recall@10, and Recall@20. This gap is expected because the geometric tokenizer retains a higher-bitrate local trace: overlapping or phonetically similar segments share more frame-derived symbols, which directly benefits normalized edit-distance ranking.

Coverage and retrieval tail. PairAlign nevertheless preserves retrieval coverage. HitRate is 1.0 for both tokenizers under all relevance definitions, and the median first relevant rank is 1 throughout. The difference appears in the harder tail of the query distribution. Mean first relevant rank increases from 36.40 to 53.99 under segment-overlap relevance, from 48.22 to 73.40 under phoneme-exact relevance, and from 38.55 to 48.95 under phoneme-relaxed relevance. The largest degradation occurs for phoneme-exact relevance, where small symbolic deviations can move the first exactly matching segment farther down the ranked list. This is consistent with PairAlign's lower-rate representation: each token covers more context, so individual substitutions, insertions, or deletions have a larger effect on normalized edit distance.

Archive compactness. Table 4 quantifies the corresponding compactness gain. Both archives contain the same 9,461 TIMIT windows, but PairAlign reduces the stored token count from 800,611 to 360,723, a 54.94% reduction and a 2.22× compression in token sequence length. Average sequence length drops from 84.62 to 38.13 tokens per segment, and the token rate drops from 28.21 to 12.71 tokens/s. The reduction is meaningful because the geometric baseline is already deduplicated, so PairAlign is not merely removing repeated frame labels.

Interpretation. The 1.5 s archive hop replaces half of the 3 s context between adjacent windows, so retrieval is not a near-duplicate lookup. In this setting, dense frame-derived tokenization is naturally favored whenever query and archive segments share local acoustic regions. PairAlign performs the same edit-distance search with less than half as many tokens. Thus, Tables 3 and 4 should be read together: PairAlign maintains full retrieval coverage, median first relevant rank of 1, broad inventory usage under TIMIT shift, and meaningful edit-distance structure while reducing symbolic archive length by 54.94%. The retrieval gap is therefore a bitrate-mediated rate–precision trade-off, not evidence of collapse or arbitrary re-tokenization.

5.3 Probing Local Symbolic Compositionality Under Continuous Acoustic Context Shifts

Continuous-sweep setup. We test how token sequences change under small shifts in acoustic context. The sweep probe uses 3 s TIMIT windows with a 100 ms hop, so adjacent windows share 2.9 s of audio and differ only at the entering and exiting boundaries. This setting asks whether neighboring windows remain symbolically related, rather than being re-tokenized as unrelated segment-level codes. We compare PairAlign with the deduplicated Stage I geometric tokenizer. Summary statistics, distributional length and overlap statistics, and edit-operation decompositions are reported in Tables 5, 6, and 7.

Dense geometric tokens preserve stronger normalized overlap. Table 5 shows that the deduplicated geometric tokenizer has higher adjacent-window overlap: edit similarity is 0.536 versus 0.414, and unigram Jaccard is 0.595 versus 0.479. This is expected for a dense frame-derived tokenizer, which emits 85.08 tokens

Metric	Geometric	PairAlign
Mean sequence length	85.08	25.48
Approx. tokens / s	28.36	8.49
Edit similarity	0.536 ± 0.217	0.414 ± 0.218
Unigram Jaccard	0.595 ± 0.227	0.479 ± 0.211
Adjacent length change $\lvert\Delta L\rvert$	10.30 ± 11.18	6.21 ± 6.95
Length ratio	0.888 ± 0.111	0.793 ± 0.182
Relative length change	12.1%	24.4%

Table 5: Continuous-sweep tokenization statistics on TIMIT (Garofolo et al., 1993) for adjacent 3 s windows with a 100 ms sweep hop. Adjacent length change is the absolute difference in decoded sequence length between neighboring windows. Relative length change is mean $|\Delta L|$ divided by mean sequence length. The geometric tokenizer is evaluated after consecutive-token deduplication.

Distributional statistic	Geometric	PairAlign
Median edit similarity	0.596	0.409
Edit similarity ≥ 0.5	70.1%	36.9%
Edit similarity ≥ 0.4	80.3%	52.2%
Median $\lvert\Delta L\rvert$	7	4

Table 6: Distributional view of adjacent-window length behavior in the continuous-sweep experiment.

per window (28.36 tokens/s) and therefore has many opportunities to reuse local symbols across highly overlapping windows. PairAlign emits a much shorter sequence, 25.48 tokens per window (8.49 tokens/s). This rate can differ from the retrieval-archive rate because PairAlign is a generative autoregressive tokenizer with adaptive length; decoded token counts can vary across runs rather than being fixed deterministically by encoder stride. Each PairAlign token therefore carries more context, so each identity change has larger normalized effect. Under normalized overlap, the geometric tokenizer is the stronger fine-grained local-persistence baseline.

PairAlign preserves a more stable coarse sequence scale. Tables 5 and 6 show the complementary length behavior. PairAlign has smaller absolute adjacent length changes despite learning length autoregressively: median $|\Delta L|$ is 4 versus 7, and mean $|\Delta L|$ is 6.21 ± 6.95 versus 10.30 ± 11.18. The distributional statistics show the same pattern: 15.6% of PairAlign transitions have no length change, compared with 4.2% for the geometric tokenizer; 48.1% have $|\Delta L| \leq 3$, compared with 27.1%; and 80.0% have $|\Delta L| \leq 10$, compared with 66.3%. Thus, although PairAlign has lower normalized overlap, neighboring windows remain closer in absolute decoded sequence scale.

PairAlign requires fewer absolute symbolic edits. Table 7 gives the clearest evidence for bounded local symbolic change relative to the previous 100 ms-shifted window. PairAlign has mean adjacent edit distance 17.48 ± 10.47 with median 15, whereas the geometric tokenizer has mean edit distance 42.57 ± 24.06 with median 36. The thresholded distribution is similarly separated: 65.8% of PairAlign transitions have $ED \leq 20$, compared with 13.2% for the geometric tokenizer, and 31.3% have $ED \leq 10$, compared with 1.0%. PairAlign edits are dominated by substitutions (9.22 on average), with smaller insertion and deletion counts (3.91 and 4.35), indicating context-sensitive relabeling within a controlled sequence scaffold rather than frequent large-scale sequence restructuring. Although normalized edit-operation rates are higher because the strings are shorter, the absolute edit script between adjacent windows is much smaller.

Granularity-aware interpretation. The sweep results support a compactness–locality trade-off. Dense geometric tokenization preserves stronger normalized overlap because it retains more frame-level redundancy. PairAlign gives up some fine-grained reuse, so a small context shift can affect a larger fraction of its shorter string. However, the model is not re-tokenizing arbitrarily: it has smaller absolute length changes, fewer absolute edit operations, nontrivial adjacent continuity, and broad token usage under TIMIT shift. PairAlign therefore behaves as a coarser sequence-level symbolic interface, not as a collapsed code or unstable holistic segment identifier.

Edit-operation statistic	Geometric	PairAlign
Edit distance ED	42.57 ± 24.06	17.48 ± 10.47
Median ED	36	15
Substitutions S	23.22 ± 20.43	9.22 ± 7.32
Insertions I	9.56 ± 8.84	3.91 ± 5.90
Deletions D	9.79 ± 9.46	4.35 ± 5.64
Substitution rate r_S	0.253 ± 0.204	0.314 ± 0.197
Insertion rate r_I	0.104 ± 0.088	0.127 ± 0.165
Deletion rate r_D	0.107 ± 0.095	0.145 ± 0.157

Table 7: Edit-operation decomposition for adjacent windows in the continuous-sweep experiment. S, I, and D denote substitutions, insertions, and deletions in an optimal Levenshtein edit script. Rates are normalized by the maximum of the two adjacent sequence lengths.

Variant	Jaccard Similarity	Edit Similarity	Exact Match	Active Vocabulary	Collapsed Pair	Exact Collision
Stage I Geometric	0.742	0.616	0.267	456	0.0803	0.0000/0.0000
Stage I+ Geometric	0.750	0.643	0.267	420	0.0290	0.0000/0.0000
Stage II AR Prior	0.682	0.593	0.248	506	0.0000	0.0000/0.0000
w/o structured self-attn dropout	1.000	1.000	1.000	6	1.0000	1.0000/1.0000
Full PairAlign	0.753	0.691	0.301	430	0.0000	0.0004/0.0028
w/o encoder-summary conditioning	0.687	0.517	0.251	508	0.0000	0.0000/0.0000

Table 8: Ablation of PairAlign on TIMIT discrete-token consistency. Collapsed Pair is the fraction of pairs in which either sequence has low-diversity ratio $r_{\text{uniq}} \leq 0.2$; Exact Collision reports anchor/positive within-stream exact-collision rates across different examples. The Stage II variant without structured self-attention dropout is a failure-mode ablation: perfect similarity indicates generic-token collapse, not useful invariance.

5.4 Ablation Study: Stage Progression and Decoder-Bypass Controls

The ablation study probes which design choices are responsible for PairAlign's operating point: compact strings that remain paired-view consistent, input-dependent, and non-collapsed. We report paired-view consistency, vocabulary use, collapse behavior, and exact collisions in Table 8; retrieval and compactness ablations are provided in Appendix H. These diagnostics are interpreted jointly: perfect agreement can reflect generic-string collapse, retrieval can benefit from greater sensitivity to local differences, and compactness is useful only when the resulting strings remain input-dependent and meaningful under edit-distance comparison.

Fixed VQ targets are only a bridge. The Stage II AR Prior evaluates the decoder before adaptive target generation. Here, the encoder and VQ codebook are frozen, and the decoder predicts deterministic Stage I VQ-derived strings across paired views; the targets remain fixed products of the geometric tokenizer. This variant is non-collapsed, with 506 active tokens, zero measured collapsed-pair rate, and zero exact collisions, but its paired-view consistency remains below Stage I+: Jaccard 0.682 versus 0.750, edit similarity 0.593 versus 0.643, and exact match 0.248 versus 0.267. Thus, autoregressive decoding over deterministic VQ targets is a useful bridge, but does not by itself produce the final PairAlign token space.

Stage III replaces these fixed VQ targets with adaptive EMA-teacher strings generated by the full encoder–decoder tokenizer. This allows the targets, decoder, and encoder-conditioned sequence geometry to co-evolve under cross-view self-alignment. With this refinement, Full PairAlign reaches edit similarity 0.691 and exact match 0.301 while keeping the collapsed-pair rate at 0.0000.

Structured self-attention dropout prevents decoder bypass. Removing structured self-attention dropout exposes the main decoder-side shortcut. The ablation obtains perfect Jaccard, edit similarity, and exact match, but this is not useful invariance: the decoder collapses to nearly input-invariant generic sequences, using only six active tokens and assigning identical strings across examples. In the reported run, this corresponds to collapsed-pair rate 1.0 and exact-collision rates 1.0/1.0. This failure occurs because teacher-forced autoregressive training can be solved through token-prefix continuation rather than acoustic

conditioning. Structured self-attention dropout weakens this continuation path, increasing prediction pressure on the encoder-conditioned pathways and allowing Stage II to remain non-collapsed before Stage III adaptive self-alignment.

Encoder-summary conditioning stabilizes ordered trajectories. Removing encoder-summary conditioning does not collapse the model, but it substantially weakens paired-view consistency. Jaccard drops from 0.753 to 0.687, edit similarity from 0.691 to 0.517, and exact match from 0.301 to 0.251, while active vocabulary increases to 508 and collapse diagnostics remain zero. Because this variant is trained without the summary pathway in both Stage II and Stage III, the comparison isolates the conditioning pathway rather than a stage-wise architecture mismatch. The result indicates that cross-attention alone is insufficient to stabilize the free-running compact trajectory; a global encoder summary helps preserve full-sequence acoustic grounding at every decoding step.

Retrieval and stability are distinct objectives. Appendix H shows that retrieval does not monotonically follow paired-view consistency. Stage I+ improves consistency over Stage I but retrieves worse under segment-overlap relevance. Similarly, the no-summary PairAlign variant gives slightly sharper retrieval than Full PairAlign, but much weaker paired-view edit similarity (0.517 versus 0.691). This reveals a retrieval–stability trade-off: greater sensitivity can improve archive discrimination under normalized edit distance while reducing consistency across views of the same segment. Full PairAlign selects the more stable operating point.

Compactness does not explain the consistency gap. The compactness ablation in Appendix H shows that the no-summary variant has nearly the same symbolic rate as Full PairAlign: 40.11 tokens per segment and 13.37 tok/s, compared with 38.13 and 12.71 tok/s. Thus, the edit-similarity drop from 0.691 to 0.517 is not a length effect. Both variants remain low-rate compared with Stage I (84.62 tokens, 28.21 tok/s) and Stage I+ (61.15 tokens, 20.38 tok/s); the difference is better explained by weaker acoustic conditioning of the autoregressive trajectory.

Ablation conclusion. Full PairAlign is not best on every isolated metric; it is the most balanced sequence tokenizer. Stage II provides a non-collapsed autoregressive bridge but is limited by fixed VQ targets. Removing structured self-attention dropout causes complete generic-string collapse. Removing encoder-summary conditioning preserves compactness and can sharpen retrieval, but substantially weakens ordered paired-view consistency. The full model is therefore preferred because it jointly satisfies the requirements of a reusable symbolic tokenizer: compactness, broad vocabulary use, no measured low-diversity collapse, strong ordered agreement under perturbation, and meaningful edit-distance retrieval.

5.5 Stress Tests of the Learned Similarity Geometry

We include two stress tests of PairAlign's learned similarity geometry. First, we test whether the English-trained similarity relation remains meaningful under lexically disjoint Tamil transfer. Second, we compare with HuBERT+VQ and WavLM+VQ, which serve as strong high-exposure geometric references rather than controlled in-pipeline ablations. These comparisons clarify scope: Stage I and Stage I+ remain the controlled baselines because they share PairAlign's preprocessing, encoder family, VQ alphabet, training regime, and evaluation protocol. Full tables, compactness measurements, and entropy visualizations are provided in Appendices I and J.

Lexically disjoint Tamil transfer probes the learned notion of similarity. All tokenizers are trained on LibriSpeech-100 and transferred directly to Shrutilipi-Tamil without adaptation. The goal is not word-level cross-lingual matching, but to test whether the induced token-space similarity relation remains stable under a different phonetic, phonotactic, prosodic, and morphosyntactic distribution. The relevant question is whether paired Tamil views remain close in token space without collapse, narrow vocabulary use, or generic strings.

PairAlign preserves this relation almost unchanged. On Tamil, it obtains Jaccard similarity 0.7194, edit similarity 0.6295, and exact match 0.2905, compared with 0.719, 0.630, and 0.291 on LibriSpeech-100. It

uses the full 512-token vocabulary, has zero measured collapsed-pair rate, and has no within-stream exact collisions. Since collapse diagnostics can be data-dependent, this is important: PairAlign shows near-zero collapse on TIMIT and remains fully non-collapsed under the Tamil shift. Its native-position entropy profile is also nearly invariant across LibriSpeech and Tamil, with mean entropy residual 0.001, mean absolute residual 0.001, and maximum absolute residual 0.003. Thus, the transfer result is not explained by degenerate strings or many-to-one assignment.

The geometric tokenizers also transfer, but their similarity relation is more sensitive to the shift. Stage I drops from 0.718 to 0.6937 in Jaccard and from 0.609 to 0.5867 in edit similarity; Stage I+ drops from 0.738 to 0.6991 and from 0.629 to 0.5995. Both activate the full vocabulary, but their Tamil collapsed-pair rates increase to 0.0843 and 0.1051. Active vocabulary alone is therefore insufficient; the relevant test is whether the similarity geometry remains stable under the new linguistic distribution.

Tamil retrieval shows the complementary effect of token granularity. Stage I and Stage I+ perform better under segment-overlap relevance, with Recall@1/MRR 0.87/0.90 and 0.85/0.87, compared with PairAlign's 0.68/0.73. This is consistent with their higher token rates: the geometric tokenizers emit 113.15 and 101.15 tokens per 3 s Tamil segment, whereas PairAlign emits 37.93 tokens. Longer frame-derived traces retain more local acoustic variation, and a modest reduction in paired-view invariance can help overlap-based retrieval by preserving evidence that distinguishes nearby archive segments. PairAlign represents the complementary lower-rate operating point: it substantially compresses the symbolic sequence while preserving a stable paired-view similarity relation and non-collapsed transfer behavior.

Pretrained SSL tokenizers are strong high-exposure geometric references. HuBERT+VQ and WavLM+VQ test a different question from the controlled PairAlign ablations. They are acoustic-symbolic tokenizers in the same broad sense as PairAlign: continuous speech is mapped to discrete strings whose similarities can be compared by symbolic sequence metrics. However, their token strings are obtained by discretizing pretrained frame-synchronous SSL features, whereas PairAlign learns a lower-rate autoregressive sequence generator. PairAlign and its Stage I/Stage I+ baselines are trained within the specified LibriSpeech pool, with approximately 667 sampled segment-hours for Stage I and 400 sampled segment-hours each for Stages II and III. HuBERT+VQ and WavLM+VQ should therefore be read as strong external larger-data-exposure references for discretized pretrained feature spaces, not compute-matched baselines.

Despite this scale difference, PairAlign remains competitive on paired-view consistency. On TIMIT, HuBERT/WavLM reach Jaccard 0.767/0.773 and edit similarity 0.726/0.730, while PairAlign reaches 0.753 and 0.691. On Tamil, WavLM remains stronger in average overlap, with Jaccard 0.766 and edit similarity 0.718, compared with PairAlign's 0.7194 and 0.6295. This is expected: pretrained SSL encoders learn robust contextual acoustic representations through large-scale masked prediction, and WavLM further incorporates denoising-oriented pretraining. After VQ discretization, these representations yield stable frame-derived acoustic-symbolic strings under clean and augmented views. The important point is that PairAlign reaches a comparable consistency regime with much smaller in-pipeline exposure and a substantially lower token rate.

The retrieval gap is also expected. HuBERT+VQ and WavLM+VQ are near-saturated on TIMIT segment-overlap retrieval, with Recall@1 0.943/0.977 and MRR 0.961/0.985, compared with PairAlign's Recall@1 0.710 and MRR 0.740. This reflects both pretraining scale and token rate. Large-scale SSL training produces contextual representations that are highly discriminative for local acoustic and phonetic structure before discretization, and the resulting frame-synchronous VQ strings preserve this evidence densely. HuBERT and WavLM operate at 28.99 and 32.06 tokens/s, compared with PairAlign's 12.71 tokens/s. Normalized edit-distance retrieval therefore favors the pretrained SSL tokenizers because their higher-rate, pretrained, frame-derived strings provide more locally discriminative evidence for top-rank archive matching.

At the same time, PairAlign achieves the highest exact anchor–positive string match on both datasets: 0.301 on TIMIT and 0.2905 on Tamil, compared with 0.263/0.264 for HuBERT/WavLM on TIMIT and 0.255/0.262 on Tamil. Exact match is a stricter compact-string criterion: the entire decoded string must remain unchanged under a content-preserving perturbation. This separates dense local overlap from string-level stability. HuBERT+VQ and WavLM+VQ show the strength of pretrained geometric acoustic-symbolic

representations; PairAlign shows that a geometric tokenization prior can be extended into an autoregressive sequence-level tokenizer that learns adaptive length, termination, and compact ordered symbolic structure while retaining meaningful acoustic-symbolic similarity.

Takeaway. These stress tests support a bounded interpretation of PairAlign. PairAlign does not claim to dominate large pretrained SSL tokenizers on dense retrieval or noise-robust paired-view consistency. HuBERT+VQ and WavLM+VQ are strong high-exposure acoustic-symbolic references: their pretrained contextual geometry and higher token rates provide more locally discriminative evidence for segment-overlap retrieval and strong robustness under perturbation. PairAlign addresses a complementary regime. It shows that a frame-synchronous geometric prior can be extended into a compact autoregressive sequence tokenizer through paired self-alignment, yielding adaptive length, termination, broad vocabulary use, and non-collapsed ordered symbolic strings. The Tamil probe shows that this learned similarity geometry remains meaningful under lexically disjoint transfer. The SSL comparison also suggests a natural scaling path: stronger pretrained SSL teachers could initialize Stage I, while the same Stage II autoregressive bridge and Stage III EMA self-alignment procedure could be used to learn lower-rate PairAlign-style tokenizations from stronger geometric representations.

6 Conclusion

This paper introduced PairAlign, a sequence-level approach to audio tokenization. The central premise is that when discrete audio tokens are used as strings for comparison, retrieval, memory, generation, or reasoning, the tokenizer should learn sequence-level structure in addition to token identity: ordering, length, termination, and edit geometry. PairAlign realizes this through cross-view self-alignment. Starting from a stable VQ-style geometric tokenizer, it trains an autoregressive decoder so that content-preserving views induce mutually predictable token strings while unrelated examples remain separated. Alignment is therefore not a post-hoc operation applied after tokenization; it is part of how the symbolic interface is induced.

PairAlign can also be viewed as sequence-symbolic predictive learning. Like JEPA-style objectives, it predicts an abstract target associated with another view rather than reconstructing the raw input. Unlike standard continuous latent prediction, the target is a learned variable-length symbolic sequence with its own vocabulary, order, length, and termination behavior. PairAlign therefore extends predictive self-supervision from representation learning to symbolic interface learning: the model learns the discrete coordinates in which acoustic structure is serialized, compared, indexed, and consumed by sequence models.

Empirically, PairAlign learns compact, non-degenerate token strings with strong ordered paired-view consistency. Its behavior is best understood as a compactness–locality trade-off. Dense geometric and pretrained SSL tokenizers retain stronger local overlap and sharper top-rank retrieval because they emit higher-rate frame-derived streams. PairAlign instead occupies a lower-rate symbolic regime: it reduces token count and comparison cost while preserving meaningful edit-distance structure, bounded local sequence evolution, broad vocabulary use, and no measured low-diversity collapse under the reported criterion. The Tamil probe further shows that the learned paired-view geometry is not merely an English lexical artifact, remaining stable under a lexically disjoint linguistic shift.

The comparison with HuBERT+VQ and WavLM+VQ clarifies the scope of the claim. PairAlign does not claim to dominate large pretrained acoustic-symbolic tokenizers on average overlap, noise-robust paired-view consistency, or rank-1 retrieval. Those systems benefit from stronger encoders, large-scale SSL pretraining, and higher-rate frame-derived traces. PairAlign contributes a complementary mechanism: it shows that a frame-synchronous geometric prior can be extended into an autoregressive sequence tokenizer through paired self-alignment. The result is a compact, ordered, non-collapsed symbolic sequence with learned length, termination, and paired-view stability. This view also suggests a natural scaling path: stronger SSL+VQ tokenizers could serve as Stage I geometric teachers, while the Stage II autoregressive bridge and Stage III EMA self-alignment procedure learn lower-rate PairAlign-style tokenizations from stronger acoustic-symbolic initializers.

PairAlign has clear limitations. It trades native frame-level timing for compact sequence-level structure, so cross-attention-based timing recovery should be viewed as an approximate post-hoc diagnostic rather

than supervised or explicitly monotonic alignment. The continuous-sweep probe should likewise be read within its intended scope: it evaluates bounded symbolic change, controlled token-count variation, and non-degenerate token usage under small window shifts. It does not require exact shift-equivariant subsequence tracking, a property not imposed during training for any tokenizer considered here, including geometric VQ, HuBERT/WavLM-based, or PairAlign tokenizers. Denser frame-derived tokenizers may show stronger local persistence because of finer temporal granularity, but this is an empirical granularity effect rather than guaranteed equivariance. Future work should study monotonic or alignment-aware decoders, stronger temporal grounding, larger multilingual and non-speech settings, and downstream uses in retrieval, generation, editing, memory, and multimodal modeling.

More broadly, PairAlign suggests that self-supervised audio systems need not end at continuous embeddings or frame-synchronous codes. They can also learn the symbolic interfaces through which continuous experience is compressed, searched, compared, composed, and eventually reasoned over.

7 Broader Impact

PairAlign studies self-supervised induction of compact symbolic sequences from continuous audio. Its most direct applications are speech and audio retrieval, indexing, matching, and comparison over large archives, especially when transcripts or reliable ASR are unavailable. Such representations may support spoken document search, low-bit rate speech processing, acoustic monitoring, audio organization, and communication-efficient storage or transmission. Because PairAlign learns variable-length token strings, it also suggests adaptive-rate symbolic coding for objectives where the goal is not waveform reconstruction, but preservation of relations needed for retrieval, alignment, generation, memory, or downstream reasoning.

The methodological implication is broader than speech. Symbolic interfaces need not be manually specified or inherited from fixed-stride tokenizers. Similar sequence-level self-alignment objectives could be studied for music, video, robotics, sensor streams, medical signals, biological recordings, and scientific time series, where meaningful discrete units are often unavailable. For generative and multimodal systems, compact and stable audio-token sequences may provide useful substrates for generation, editing, long-context modeling, and multimodal grounding.

These opportunities also introduce risks. Compact searchable audio tokens could make large-scale indexing and analysis easier, including for private or sensitive recordings. PairAlign tokens should not be assumed anonymous: they may preserve speech content, speaker information, environmental context, or other sensitive attributes. Any deployment should therefore require consent-aware data collection, access control, privacy analysis, and compliance with relevant domain regulations. If such tokens are used in generative or editing systems, additional safeguards may be needed against deceptive synthesis, impersonation, or unauthorized voice manipulation, including provenance tracking, watermarking where appropriate, and consent-aware training data.

The present work is limited to speech tokenization and retrieval-oriented analysis. The learned tokens are not shown to correspond to phonemes, words, events, or human-interpretable units, and PairAlign trades native frame-level timing for compact sequence-level structure. Its broader significance is therefore methodological rather than deployable as-is: it demonstrates one route to learning symbolic sequence interfaces from continuous data, while any new domain or application will require separate validation of utility, interpretability, privacy, and safety.

8 Future Work

PairAlign opens directions in sequence-level tokenization, self-supervised symbolic representation learning, and low-rate perceptual coding. The present work shows that token identity, order, length, and cross-view consistency can be shaped by predictive self-alignment rather than inherited only from local quantization. We outline several directions.

Learned-length, adaptive-rate, and semantic compression. PairAlign makes output length a learned decision rather than a direct consequence of encoder stride, frame rate, or post hoc collapse. Future work should study how this learned length relates to acoustic and semantic complexity, and how explicit rate or information-budget constraints can control the compactness–fidelity trade-off. This would help determine when a compact token string is sufficient, and when preserving finer acoustic or semantic detail requires a higher token rate.

Scaling symbolic self-alignment beyond short speech. This work uses short speech segments and controlled content-preserving augmentations. A natural next step is to increase the input context length and study richer view construction, including retrieval-based positives, weak semantic matching, cross-modal context, repeated acoustic patterns, and TTS- or generative-model-guided augmentations. Scaling to longer audio, multilingual and lexically disjoint speech, music, environmental sound, and mixed scenes would test whether sequence-level self-alignment remains stable beyond short-window English speech. Similar objectives may also be useful for other time series where discrete structure is present but symbolic units are not given.

Symbolic compression algorithms. A natural direction is to extend PairAlign-style tokenization toward symbolic sequence compression. In this work, PairAlign maps continuous speech to compact discrete strings while preserving sequence-level relations. More generally, the same principle could apply to already-discrete audio streams: a PairAlign-style model could map a long input token sequence to a shorter compressed token sequence that preserves relations needed for retrieval, reconstruction, prediction, or downstream reasoning. In this view, the tokenizer becomes a learned sequence compressor from high-rate symbolic representations to lower-rate symbolic codes.

Such compressed sequences could serve as targets, memory states, or conditioning interfaces for models operating directly over audio tokens. Rather than reasoning over raw frames, dense acoustic features, or long codec-token streams, downstream systems could operate over compact learned symbolic structure. This may support more efficient long-context audio modeling, memory construction, retrieval-aware generation, symbolic summarization, and codec-native audio intelligence.

Sequence-level objectives, symbolic geometries, and stabilization. PairAlign shapes edit-distance behavior indirectly through cross-paired likelihood and in-batch contrast. Future work should study more direct sequence-level objectives for symbolic similarity, while improving teacher stability, entropy control, length control, and anti-collapse regularization. These directions directly target the main failure modes observed in this work: decoder bypass, generic prefixes, unstable length, and teacher drift.

Temporal grounding, streaming, and online tokenization. The current implementation recovers approximate timing post hoc. Future work should make temporal grounding part of the model through monotonic attention, monotonic chunkwise attention, hard monotonic transduction, transducer-style decoders, duration-explicit decoders, or other architectures with explicit time-token structure. This is important for segmentation, localization, streaming retrieval, and time-aware editing. An online PairAlign variant would also need to emit tokens incrementally while controlling latency, boundary decisions, EOS behavior, and possible re-tokenization as additional context arrives.

Hierarchical, disentangled, and interpretable symbolic units. Future work could learn multiple symbolic streams at different temporal and semantic scales. For speech, these streams could capture phonetic, lexical/prosodic, speaker, and channel information; for music, they could encode rhythm, timbre, melody, harmony, and performance style. Disentanglement could be encouraged by applying different view transformations to different streams, such as content-preserving speaker perturbations for lexical tokens or speaker-preserving transformations for speaker tokens. Future work should study the semantics of learned tokens and subsequences through temporal alignment and controlled generation or editing.

PairAlign-style objectives for low-bitrate audio-language, generative, and agentic systems. Retrieval is only one test of whether a symbolic interface is compact, stable, and comparison-friendly. Future applications include audio-language interaction, audio question answering, audio chat (Chen et al.,

2026), instruction-conditioned audio understanding, symbolic audio editing, long-context audio modeling, and compact memory over perceptual streams. PairAlign-style objectives are promising because they learn low-bitrate symbolic sequences while preserving relations needed for comparison, prediction, and downstream reasoning. Even when the specific tokens do not transfer, the learning principle can: token identity, order, length, and cross-view consistency can be organized by predictive self-alignment.

Reasoning-trace alignment and symbolic predictive world models. PairAlign can be viewed as a sequence-symbolic analogue of JEPA-style predictive learning. Instead of predicting a fixed-dimensional continuous target for another view, it predicts a learned variable-length token sequence with an induced vocabulary, EOS decision, and sequence geometry. This suggests a broader direction in which perceptual systems learn compact symbolic traces that are mutually predictable across views. In multimodal settings, audio, vision, text, and action streams could keep modality-specific token spaces while being aligned through cross-view prediction. This would extend self-supervision from embedding invariance toward alignment of structured symbolic traces across modalities.

Privacy-preserving and communication-efficient representations. Low-rate symbolic tokenization may enable communication-efficient and privacy-aware processing, but it also raises risks because tokens may preserve speech content, speaker identity, or environmental context. Future work should measure which attributes PairAlign tokens preserve or discard, and study privacy constraints, adversarial objectives, factorized streams, or rate limits on sensitive attributes within the alignment objective.

Toward a theory of self-supervised symbolic induction. PairAlign raises theoretical questions beyond audio. What properties of positive views support stable symbol emergence? What forms of contrast prevent collapse without removing useful invariance? How do learned length, EOS decisions, vocabulary usage, entropy, rate, discriminability, and edit geometry interact? From a JEPA-style perspective, PairAlign asks how predictive self-supervision changes when the prediction target is a learned variable-length symbolic sequence rather than a fixed-dimensional continuous embedding. PairAlign provides one concrete instance of such a system: a self-supervised tokenizer that induces a low bit-rate symbolic interface for comparison and retrieval. A longer-term goal is to understand when similar principles can induce compact symbolic interfaces for memory, reasoning, and multimodal event structure.

9 Declaration of LLM Usage.

LLM is used only to aid or polish writing and does not impact the core methodology, scientific rigorousness, or originality of the research.

References

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15619–15629, 2023.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pp. 4447–4455. PMLR, 2024.

Alexei Baevski, Steffen Schneider, and Michael Auli. vq-wav2vec: Self-supervised learning of discrete speech representations. *arXiv preprint arXiv:1910.05453*, 2019.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022a.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022b.

Adhiraj Banerjee and Vipul Arora. wav2tok: Deep sequence tokenizer for audio retrieval. In *The Eleventh International Conference on Learning Representations*, 2022.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

Kaushal Bhogale, Abhigyan Raman, Tahir Javed, Sumanth Doddapaneni, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh M Khapra. Effectiveness of mining audio and text pairs from public data for improving asr systems for low-resource languages. In *Icassp 2023-2023 ieee international conference on acoustics, speech and signal processing (icassp)*, pp. 1–5. IEEE, 2023.

Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, et al. Audiolm: a language modeling approach to audio generation. *IEEE/ACM transactions on audio, speech, and language processing*, 31: 2523–2533, 2023.

Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, et al. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1505–1518, 2022.

William Chen, Prem Seetharaman, Rithesh Kumar, Oriol Nieto, Shinji Watanabe, Justin Salamon, and Zeyu Jin. Audiochat: Unified audio storytelling, editing, and understanding with transfusion forcing, 2026. URL `https://arxiv.org/abs/2602.17097`.

Chung-Cheng Chiu and Colin Raffel. Monotonic chunkwise attention. *arXiv preprint arXiv:1712.05382*, 2017.

Jishnu Ray Chowdhury and Cornelia Caragea. Monotonic location attention for length generalization. In *International Conference on Machine Learning*, pp. 28792–28808. PMLR, 2023.

Yu-An Chung, Yu Zhang, Wei Han, Chung-Cheng Chiu, James Qin, Ruoming Pang, and Yonghui Wu. W2v-bert: Combining contrastive learning and masked language modeling for self-supervised speech pre-training. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 244–250. IEEE, 2021.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438*, 2022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. *Advances in neural information processing systems*, 32, 2019.

Matthew S. Dryer and Martin Haspelmath (eds.). *WALS Online (v2020.4)*. Zenodo, 2013. doi: 10.5281/zenodo.13950591. URL `https://doi.org/10.5281/zenodo.13950591`.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

Zhengcong Fei, Mingyuan Fan, and Junshi Huang. A-jepa: Joint-embedding predictive architecture can listen. *arXiv preprint arXiv:2311.15830*, 2023.

John S Garofolo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, and David S Pallett. Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon technical report n*, 93:27403, 1993.

Alex Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376, 2006.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

Junfeng Hou, Shiliang Zhang, and Li-Rong Dai. Gaussian prediction based attention for online end-to-end speech recognition. In *Proc. Interspeech 2017*, pp. 3692–3696, 2017.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM transactions on audio, speech, and language processing*, 29:3451–3460, 2021.

Zeqian Ju, Yuancheng Wang, Kai Shen, Xu Tan, Detai Xin, Dongchao Yang, Yanqing Liu, Yichong Leng, Kaitao Song, Siliang Tang, et al. Naturalspeech 3: Zero-shot speech synthesis with factorized codec and diffusion models. *arXiv preprint arXiv:2403.03100*, 2024.

Bhadriraju Krishnamurti. *The Dravidian Languages*. Cambridge Language Surveys. Cambridge University Press, 2003.

Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. High-fidelity audio compression with improved rvqgan. *Advances in Neural Information Processing Systems*, 36:27980–27993, 2023.

Yann LeCun et al. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 7871–7880, 2020.

Xutai Ma, Juan Pino, James Cross, Liezl Puzon, and Jiatao Gu. Monotonic multihead attention. *arXiv preprint arXiv:1909.12406*, 2019.

André Merboldt, Albert Zeyer, Ralf Schlüter, and Hermann Ney. An analysis of local monotonic attention variants. In *Interspeech*, pp. 1398–1402, 2019.

Pooneh Mousavi, Gallil Maimon, Adel Moumen, Darius Petermann, Jiatong Shi, Haibin Wu, Haici Yang, Anastasia Kuznetsova, Artem Ploujnikov, Ricard Marxer, et al. Discrete audio tokens: More than a survey! *arXiv preprint arXiv:2506.10274*, 2025.

Paarth Neekhara, Shehzeen Hussain, Subhankar Ghosh, Jason Li, Rafael Valle, Rohan Badlani, and Boris Ginsburg. Improving robustness of llm-based speech synthesis by learning monotonic alignment. *arXiv preprint arXiv:2406.17957*, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5206–5210. IEEE, 2015.

Ofir Press, Noah A Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pp. 28492–28518. PMLR, 2023.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Colin Raffel, Minh-Thang Luong, Peter J Liu, Ron J Weiss, and Douglas Eck. Online and linear-time attention by enforcing monotonic alignments. In *International conference on machine learning*, pp. 2837–2846. PMLR, 2017.

Hasim Sak, Matt Shannon, Kanishka Rao, and Françoise Beaufays. Recurrent neural aligner: An encoder-decoder neural network model for sequence to sequence mapping. In *Interspeech*, volume 8, pp. 1298–1302, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 464–468, 2018.

Anup Singh, Kris Demuynck, and Vipul Arora. Best-std: Bidirectional mamba-enhanced speech tokenization for spoken term detection. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*, 2019.

Sanford B. Steever. *The Dravidian Language Family*, pp. 887–910. Cambridge Handbooks in Language and Linguistics. Cambridge University Press, 2017.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. Local monotonic attention mechanism for end-to-end speech and language processing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 431–440, 2017.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, et al. Neural codec language models are zero-shot text to speech synthesizers. *arXiv preprint arXiv:2301.02111*, 2023.

Shijie Wu and Ryan Cotterell. Exact hard monotonic attention for character-level transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1530–1537, 2019.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30: 495–507, 2021.

Xin Zhang, Dong Zhang, Shimin Li, Yaqian Zhou, and Xipeng Qiu. Speechtokenizer: Unified speech tokenizer for speech large language models. *arXiv preprint arXiv:2308.16692*, 2023.

Yingzhu Zhao, Chongjia Ni, Cheung-Chi Leung, Shafiq R Joty, Eng Siong Chng, and Bin Ma. Cross attention with monotonic alignment for speech transformer. In *Interspeech*, pp. 5031–5035, 2020.

A Stage I Details: Encoder Pretraining and Geometric VQ Tokenization

Stage I trains the encoder and VQ codebook before any autoregressive sequence-level modeling. The encoder produces contextual frame embeddings $Z_i = [z_{i,t}]_{t=1}^{T_i}$ and $Z_i^+ = [z_{i,t}^+]_{t=1}^{T_i^+}$ for paired views.

Frame-level contrastive loss. Following BEST-STD (Singh et al., 2025), paired views are aligned with DTW. Let $t^+(t)$ denote the frame in the positive view aligned to anchor frame t. The contrastive loss for example i is

$$\mathcal{L}_{\text{contrast}}^{(i)} = -\frac{1}{T_i}\sum_{t=1}^{T_i}\log\frac{\exp\Big(z_{i,t}^\top z_{i,t^+(t)}^+/\tau\Big)}{\exp\Big(z_{i,t}^\top z_{i,t^+(t)}^+/\tau\Big)+\sum_{n\in\mathcal{N}_{i,t}}\exp\big(z_{i,t}^\top z_n/\tau\big)}. \tag{21}$$

Here τ is the temperature and $\mathcal{N}_{i,t}$ denotes unrelated negative frames from the minibatch.

Nearest-centroid VQ assignment. Let $C = \{c_a : a \in \mathcal{A}\}$ be the VQ codebook. Each frame is assigned to its nearest centroid:

$$\tau_{i,t} = \arg\min_{a\in\mathcal{A}} \|z_{i,t} - c_a\|_2^2, \qquad z_{i,t}^q = c_{\tau_{i,t}}. \tag{22}$$

Commitment loss. The commitment loss is

$$\mathcal{L}_{\text{commit}}^{(i)} = \frac{1}{T_i}\sum_{t=1}^{T_i}\|z_{i,t} - z_{i,t}^q\|_2^2 + \frac{1}{T_i^+}\sum_{t=1}^{T_i^+}\|z_{i,t}^+ - z_{i,t}^{q,+}\|_2^2. \tag{23}$$

Stage I objective. The Stage I training objective is

$$\mathcal{L}_{\text{StageI}} = \frac{1}{B}\sum_{i=1}^{B}\left(\mathcal{L}_{\text{contrast}}^{(i)} + \lambda_{\text{commit}}\mathcal{L}_{\text{commit}}^{(i)}\right). \tag{24}$$

EMA codebook update. For token a, let n_a be the number of assigned frames in the minibatch and m_a the sum of their encoder states. The EMA statistics are updated as

$$N_a \leftarrow \mu N_a + (1-\mu)n_a, \qquad M_a \leftarrow \mu M_a + (1-\mu)m_a, \tag{25}$$

and the centroid is updated by

$$c_a \leftarrow \frac{M_a}{N_a + \varepsilon}. \tag{26}$$

Deduplicated geometric token sequence. The raw frame-synchronous token sequence is

$$\mathcal{T}_i^{\text{raw}} = [\tau_{i,1}, \ldots, \tau_{i,T_i}]. \tag{27}$$

For sequence-level comparison and for Stage II targets, we apply run-length deduplication:

$$\mathcal{T}_i = \phi(\mathcal{T}_i^{\text{raw}}). \tag{28}$$

B Stage I+ Details: Sequence-Consistent Geometric Tokenization

Stage I+ keeps the Stage I frame-synchronous VQ tokenizer but adds a symmetric no-blank CTC sequence-consistency term between paired views.

Framewise token posteriors. For each encoder state, we compute a posterior over the VQ alphabet:

$$p_{i,t}(a) = \frac{\exp(z_{i,t}^\top c_a)}{\sum_{a'\in\mathcal{A}}\exp(z_{i,t}^\top c_{a'})}, \qquad a \in \mathcal{A}. \tag{29}$$

The positive-view posteriors $p_{i,t}^{+}(a)$ are defined analogously.

No-blank CTC likelihood. Let $\mathcal{T} = (\tau_1, \ldots, \tau_L)$ be a deduplicated target sequence and let $P = \{p_t(a)\}_{t=1}^{T}$ be framewise token posteriors. The no-blank CTC likelihood is

$$p_{\text{CTC}}(\mathcal{T} \mid P) = \sum_{\pi:\phi(\pi)=\mathcal{T}}\prod_{t=1}^{T} p_t(\pi_t), \tag{30}$$

where $\phi(\cdot)$ removes consecutive repetitions.

Forward recursion. Let $\alpha_t(l)$ be the probability of all paths up to frame t that collapse to prefix $(\tau_1, \ldots, \tau_l)$. The recursion is

$$\alpha_1(1) = p_1(\tau_1), \qquad \alpha_1(l) = 0 \ \text{ for } l > 1, \tag{31}$$

$$\alpha_t(l) = [\alpha_{t-1}(l) + \alpha_{t-1}(l-1)]\, p_t(\tau_l), \tag{32}$$

with out-of-range terms set to zero. The likelihood is $p_{\text{CTC}}(\mathcal{T} \mid P) = \alpha_T(L)$. The recursion is implemented in log space.

Backward recursion. Let $\beta_t(l)$ be the probability of all suffix paths from frame t to T that collapse to suffix $(\tau_l, \ldots, \tau_L)$. The initialization is

$$\beta_T(L) = p_T(\tau_L), \qquad \beta_T(l) = 0 \quad \text{for } l < L. \tag{33}$$

For $t \le T-1$,

$$\beta_t(l) = p_t(\tau_l)\,[\beta_{t+1}(l) + \beta_{t+1}(l+1)]\,, \tag{34}$$

again with out-of-range terms set to zero. The recursions are implemented in log space for numerical stability.

Algorithm 1 EMA-Teacher Target Generation

Require: Teacher encoder states $\widetilde{Z}$, teacher decoder $\widetilde{Dec}$, valid encoder length T_x, rate cap ρ, schedules (p_l, τ_l, γ_l)
Ensure: Sampled teacher target $\widehat{\mathcal{T}}$

```
Initialize prefix T̂ ← [BOS]
Set length cap L_cap ← ⌊ρ T_x⌋
for l = 1, ..., L_cap do
    Compute next-token logits ℓ_l ← Dec(T̂_<l, Z̃)
    Count prefix occurrences c_l(v) for each token v
    Set c_l(v) = 0 for special symbols excluded from repetition control
    Apply compound repetition penalty:
        ℓ̃_l(v) ← ℓ_l(v) − c_l(v) log γ_l
    if l = L_cap then
        Force τ̂_l ← EOS
    else
        Sort ℓ̃_l in descending order
        Retain the smallest nucleus whose cumulative probability mass exceeds p_l
        Always retain the highest-probability token
        Mask all non-retained logits to −∞
        Apply temperature scaling by τ_l
        Sample τ̂_l from the resulting categorical distribution
    end if
    Append τ̂_l to T̂
    if τ̂_l = EOS then
        break
    end if
end for
Remove BOS and return tokens up to and including EOS
```

Symmetric pairwise sequence loss. For paired views, Stage I+ scores each deduplicated target sequence under the other view's framewise posteriors:

$$\mathcal{L}_{\text{seq}}^{(i)} = \frac{1}{2}\left[-\log p_{\text{CTC}}(\mathcal{T}_i^{+} \mid P_i) - \log p_{\text{CTC}}(\mathcal{T}_i \mid P_i^{+})\right]. \tag{35}$$

Adaptive CTC weight. The sequence term is weighted by

$$\lambda_{\text{CTC}} = \gamma \frac{\overline{\mathcal{L}}_{\text{contrast}}}{\overline{\mathcal{L}}_{\text{seq}} + \varepsilon}, \tag{36}$$

where γ controls the relative strength of the sequence term and ε is a numerical stabilizer.

Stage I+ objective. The complete Stage I+ objective is

$$\mathcal{L}_{\text{StageI+}} = \frac{1}{B}\sum_{i=1}^{B}\left(\mathcal{L}_{\text{contrast}}^{(i)} + \lambda_{\text{commit}}\mathcal{L}_{\text{commit}}^{(i)} + \lambda_{\text{CTC}}\mathcal{L}_{\text{seq}}^{(i)}\right). \tag{37}$$

C Teacher Target Generation via Top-p Sampling with Compound Repetition Penalty and Differential Stochasticity

This appendix gives the Stage III EMA-teacher sampling procedure. The role of teacher targets, top-p sampling, repetition control, and length-constrained EOS handling is described in Section 3.4.3.

Compound repetition penalty. At decoding step l, let $\ell_l(v)$ be the teacher logit for token v and let $c_l(v)$ be the number of times token v appears in the generated prefix. Special symbols such as BOS, EOS, and PAD are excluded by setting their counts to zero. Given repetition factor $\gamma_l > 1$, repeated tokens are downweighted in probability space as

$$\widetilde{p}_l(v) \propto p_l(v)\gamma_l^{-c_l(v)}.$$

Since $p_l(v) \propto \exp(\ell_l(v))$, this is implemented by the equivalent logit shift

$$\widetilde{\ell}_l(v) = \ell_l(v) - c_l(v)\log\gamma_l.$$

The penalty therefore compounds with repeated use of the same token, leaves unseen tokens unchanged, and modifies only the next-token sampling distribution.

Step-dependent schedule. The sampling parameters are position dependent:

$$(\tau_l, p_l, \gamma_l) = \begin{cases} (\tau_{\text{early}}, p_{\text{early}}, \gamma_{\text{early}}), & l \le K_{\text{early}}, \\ (\tau_{\text{late}}, p_{\text{late}}, \gamma_{\text{late}}), & l > K_{\text{early}}. \end{cases}$$

We use more stochastic early decoding and sharper later decoding:

$$\tau_{\text{early}} > \tau_{\text{late}}, \qquad p_{\text{early}} > p_{\text{late}}, \qquad \gamma_{\text{early}} < \gamma_{\text{late}}.$$

D Inference-Time Beam Search

This appendix gives the inference-time beam-search procedure. The role of beam search, EOS constraints, and length normalization is described in Section 3.4.3. Beam search uses the same compound repetition penalty as Appendix C: repeated non-special tokens receive the count-dependent logit shift $-c_l(v)\log\gamma_{\text{rep}}$ before beam expansion. In teacher sampling, this penalty shapes the stochastic top-p distribution; at inference, it shapes deterministic beam scores. In both cases, it affects only next-token selection and does not edit the decoded sequence post hoc.

Length-normalized ranking. When length normalization is enabled, a beam with cumulative log-probability S and decoded length L is ranked using $S_{\text{rank}} = \frac{S}{L^{\alpha_{\text{len}}}}$. Setting $\alpha_{\text{len}} = 0$ recovers standard unnormalized beam search. Length normalization changes only beam ranking and final selection; the beam score remains the cumulative sum of token log-probabilities.

E Timing Recovery from Autoregressive Decoding

This appendix describes the inference-time timestamp extraction procedure used for the timing diagnostics in Section E.1. The method is purely post hoc: it does not modify the training objective, introduce an alignment loss, constrain the decoder during learning, or backpropagate through recovered alignments. It uses decoder–encoder cross-attention as a soft token-to-frame association signal and converts it into an approximate monotone token segmentation.

Diagonal Beta attention prior. Let $W \in \mathbb{R}^{L\times T}$ denote the token-to-frame cross-attention matrix after removing BOS, EOS, decoder padding, and padded encoder frames, where L is the number of decoded non-special tokens and T is the number of valid encoder frames. Since raw cross-attention may be diffuse or multi-modal, we apply a weak diagonal prior before extracting a hard monotone path.

For token position l and encoder frame t, define normalized coordinates

$$\bar{l} = \frac{l-1}{L-1}, \qquad \bar{t} = \frac{t-1}{T-1},$$

for $L, T > 1$. We define a row-wise Beta-shaped prior over encoder frames as

$$P_{l,t} \propto \bar{t}^{\,\alpha_l - 1}(1-\bar{t})^{\,\beta_l - 1}, \qquad \alpha_l = 1 + a\bar{l}, \qquad \beta_l = 1 + b(1-\bar{l}),$$

Algorithm 2 Inference-Time Beam Search with Repetition Control

Require: Conditioning latent Z, decoder Dec, beam size K, valid encoder length T_x, rate cap ρ, repetition factor γ_{rep}, optional length-normalization exponent α_{len}

Ensure: Decoded token sequence $\widehat{\mathcal{T}}$

Initialize beam set with one active hypothesis [BOS] of score 0 and $K-1$ inactive hypotheses of score $-\infty$
Set length cap $L_{\text{cap}} \leftarrow \lfloor \rho T_x \rfloor$
for $l = 1, \ldots, L_{\text{cap}}$ **do**
  Initialize candidate set $\mathcal{C} \leftarrow \emptyset$
  for each beam hypothesis $k = 1, \ldots, K$ **do**
    if beam k is finished **then**
      Add the unchanged finished beam to $\mathcal{C}$
      continue
    end if
    Compute next-token logits $\ell_l^{(k)} \leftarrow Dec(\mathcal{T}_{<l}^{(k)}, Z)$
    Count prefix occurrences $c_l^{(k)}(v)$ for each token v
    Set $c_l^{(k)}(v) = 0$ for special symbols excluded from repetition control
    Apply repetition penalty:

$$\widetilde{\ell}_l^{(k)}(v) \leftarrow \ell_l^{(k)}(v) - c_l^{(k)}(v) \log \gamma_{\text{rep}}.$$

    Disallow PAD for unfinished beams
    if $l = L_{\text{cap}}$ **then**
      Force EOS by masking all non-EOS tokens
    end if
    Convert adjusted logits to log-probabilities:

$$\log p_l^{(k)}(v) = \log \operatorname{softmax}(\widetilde{\ell}_l^{(k)})(v).$$

    for each vocabulary token v **do**
      Add candidate $\mathcal{T}^{(k)} \circ v$ with score $S^{(k)} + \log p_l^{(k)}(v)$ to $\mathcal{C}$
    end for
  end for
  Retain the top K candidates from $\mathcal{C}$, using length-normalized ranking when $\alpha_{\text{len}} > 0$
  Mark beams ending in EOS as finished
  if all retained beams are finished **then**
    break
  end if
end for
Select the highest-ranking completed beam; if none is completed, select the highest-ranking current beam
Remove BOS and return tokens up to and including EOS if present

followed by row normalization: $\sum_{t=1}^{T} P_{l,t} = 1$.

This prior assigns earlier decoded token positions more mass near earlier encoder frames and later decoded token positions more mass near later encoder frames. The parameters a and b control the strength and asymmetry of this diagonal preference.

We apply the prior multiplicatively:

$$\widetilde{W}_{l,t} = \frac{W_{l,t} P_{l,t}^{\omega}}{\sum_{t'=1}^{T} W_{l,t'} P_{l,t'}^{\omega} + \varepsilon},$$

Algorithm 3 Post-Hoc Timing Recovery from Decoder Cross-Attention

Require: Input-window start time $t^{\text{win-start}}$, decoded token sequence $\mathcal{T}$, decoder–encoder cross-attention tensor, encoder-valid mask, sampling rate F_s, encoder downsampling factor M, prior strength ω

Ensure: Time-stamped token sequence $\{(u_l, t_l^{\text{start}}, t_l^{\text{end}})\}_{l=1}^{L}$

1: Remove BOS, EOS, decoder padding, and padded encoder frames

2: Average selected attention heads and layers to obtain $W \in \mathbb{R}^{L\times T}$

3: Construct the diagonal Beta attention prior $P \in \mathbb{R}^{L\times T}$

4: Apply prior reweighting:

$$\widetilde{W}_{l,t} = \frac{W_{l,t} P_{l,t}^{\omega}}{\sum_{t'=1}^{T} W_{l,t'} P_{l,t'}^{\omega} + \varepsilon}.$$

5: Convert token-to-frame attention into a frame-to-token posterior:

$$A_{t,l} = \frac{\widetilde{W}_{l,t}}{\sum_{l'=1}^{L} \widetilde{W}_{l',t} + \varepsilon}.$$

6: Decode a monotone Viterbi path π^* through A:

$$\pi^* = \arg\max_{\pi\in\mathcal{M}} \sum_{t=1}^{T} \log(A_{t,\pi_t+1} + \varepsilon),$$

where $\mathcal{M}$ contains paths that either stay at the current token or advance to the next token.

7: Set encoder frame duration $\Delta t = M/F_s$

8: **for** $l = 1, \ldots, L$ **do**

9: Collect assigned frames $\mathcal{I}_l = \{t : \pi_t^* = l - 1\}$

10: Convert assigned frame support to timestamps:

$$t_l^{\text{start}} = t^{\text{win-start}} + (\min \mathcal{I}_l - 1)\,\Delta t, \qquad t_l^{\text{end}} = t^{\text{win-start}} + \max \mathcal{I}_l\,\Delta t.$$

11: **end for**

12: Return $\{(u_l, t_l^{\text{start}}, t_l^{\text{end}})\}_{l=1}^{L}$

where ω controls the strength of the post-hoc bias and ε is a numerical stabilizer. This reweighting is applied only during timestamp extraction; it does not alter the learned decoder attention mechanism.

The recovered timestamps should be interpreted as approximate temporal grounding for learned autoregressive token positions, not as supervised phone boundaries or semantic labels for the token inventory. The method can summarize near-monotone structure already present in decoder cross-attention, but it cannot create reliable timing structure when the attention map is diffuse or strongly non-monotone.

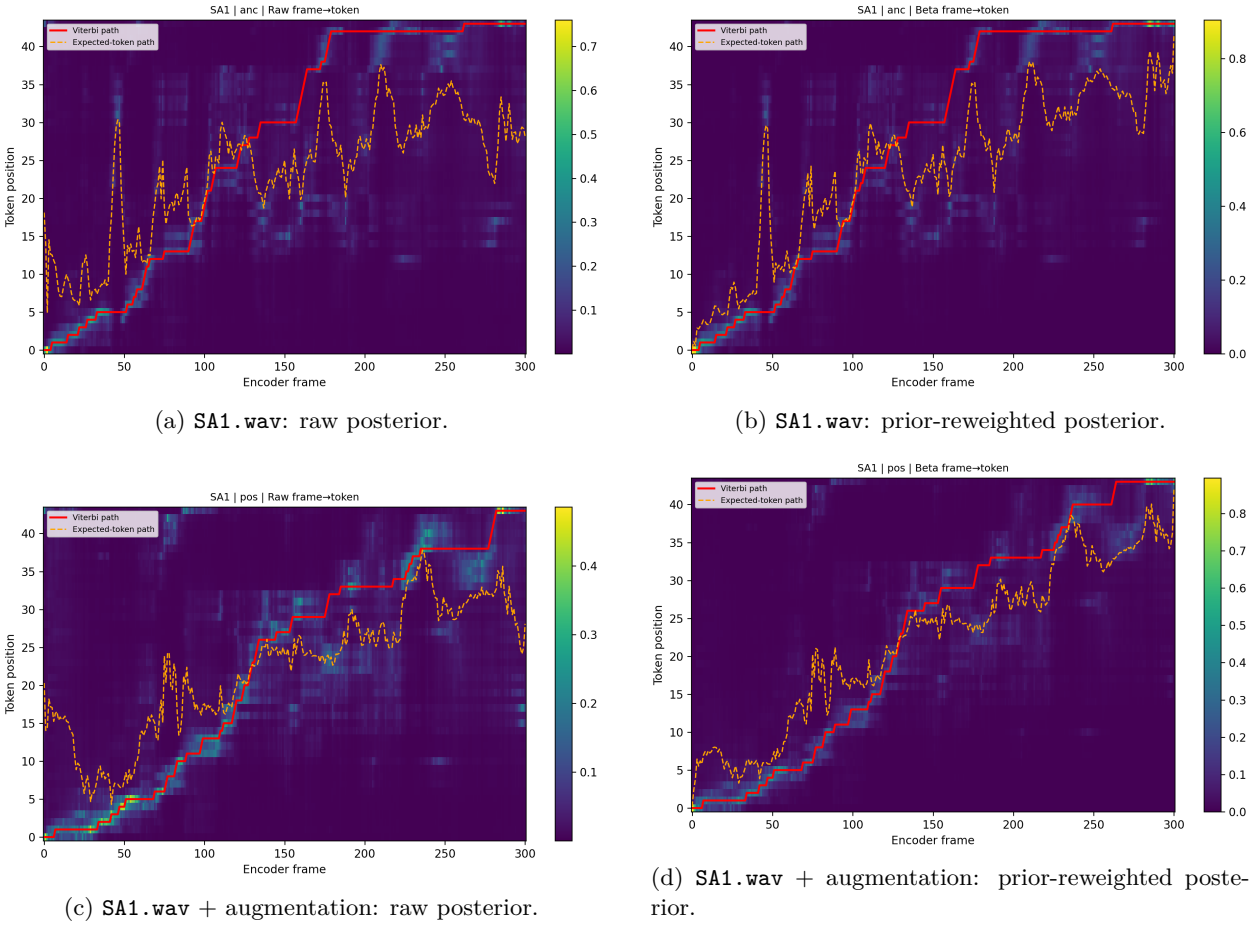
E.1 Post-Hoc Timing Recoverability

PairAlign emits compact autoregressive token sequences rather than frame-synchronous labels, so token timing must be recovered post hoc. This appendix evaluates whether a weak diagonal Beta-shaped prior improves token-to-time recovery from decoder–encoder cross-attention. The prior is applied only after decoding, before monotone Viterbi alignment, and serves to sharpen monotonic structure already present in the attention map. It is not used during training and does not modify the tokenizer.

Experimental setting. We evaluate 1,000 TIMIT examples and their augmented positive views, yielding 2,000 decoded sequences. Each input is a 3 s speech window with 301 valid encoder frames. Decoded length is variable, with average length 26.66 for anchors and 24.89 for positives. For each example, we extract decoder cross-attention, convert it to a frame-to-token posterior, and compare the raw posterior with the posterior obtained after applying the diagonal Beta attention prior from Appendix E.

Frame-to-token metric	Raw	Prior	Prior − Raw	Median change	Frac. improved
Expected violation rate ↓	0.4734	0.3769	−0.0966	−0.0467	0.991
Peak violation rate ↓	0.0609	0.0605	−0.0004	0.0000	0.431
Mean absolute diagonal error ↓	0.2626	0.2492	−0.0134	−0.0129	1.000
Mean peak diagonal error ↓	0.2862	0.2809	−0.0053	−0.0011	0.665
Mean Viterbi log-score ↑	−1.4895	−1.4024	+0.0870	+0.0792	0.964

Table 9: Frame-to-token timing-recovery diagnostics over anchor and positive views. The diagonal Beta attention prior improves soft monotonicity, diagonal consistency, and Viterbi path support. Peak-based metrics change only slightly, indicating that the prior mainly regularizes probability mass around the decoder's existing attention structure rather than replacing the dominant local assignment.



(a) `SA1.wav`: raw posterior.

(b) `SA1.wav`: prior-reweighted posterior.

(c) `SA1.wav` + augmentation: raw posterior.

(d) `SA1.wav` + augmentation: prior-reweighted posterior.

Figure 1: Frame-to-token timing-recovery visualization for a 3 s TIMIT segment from `SA1.wav` and an augmented version of the same segment. The x-axis denotes encoder frame index and the y-axis denotes decoded token timestep. The heatmap shows the posterior probability of assigning each encoder frame to each decoded token timestep. The solid curve is the monotone Viterbi path, and the dashed curve is the expected-token path.

Metrics. Timestamp recovery assigns encoder frames to decoded token timesteps, so we evaluate the frame-to-token direction. We report expected violation rate, peak violation rate, mean absolute diagonal error, mean peak diagonal error, and Viterbi log-score. Lower violation and diagonal-error values indicate stronger compatibility with a forward-moving temporal assignment; higher Viterbi log-score indicates stronger support for a globally monotone path.

Results. Table 9 shows that the diagonal Beta prior improves the soft frame-to-token posterior. Expected violation rate decreases from 0.4734 to 0.3769, improving in 99.1% of examples. Mean absolute diagonal error also decreases, and mean Viterbi log-score improves from -1.4895 to -1.4024, improving in 96.4% of examples. The reweighted posterior is therefore more compatible with a forward-moving monotone assignment from encoder frames to decoded token timesteps.

Peak-based metrics change much less. Peak violation rate changes only from 0.0609 to 0.0605, with zero median change. Thus, the prior usually does not replace the dominant token assignment at each frame; it mainly regularizes the soft posterior around temporal structure already present in the decoder cross-attention.

Figure 1 illustrates the same behavior on a 3 s TIMIT segment from `SA1.wav` and its augmented view. Prior reweighting produces clearer diagonal organization and stronger support for the monotone path, but the recovered alignment remains approximate. Decoded token timesteps should not be interpreted as phones, syllables, or known linguistic units; the maps ground learned token positions in encoder time, not semantic labels.

Overall, decoder cross-attention provides a usable but limited timing signal. The diagonal Beta prior improves soft monotonicity and Viterbi path support, while the weak change in hard peak metrics shows that it mainly sharpens monotonic structure already present in the learned attention. The recovered paths should therefore be interpreted as approximate token-to-time assignments, not as validated frame-level boundaries. Timing recovery is an auxiliary diagnostic for temporal grounding, visualization, segmentation probes, and time-stamped retrieval analysis, not part of the core PairAlign training objective.

F Full Experimental Details

F.1 Discrete Token Consistency

For each anchor segment x_i and augmented positive view $x_i^+ = \mathrm{Aug}(x_i)$, we independently decode token sequences $\mathcal{T}_i$ and $\mathcal{T}_i^+$. All metrics are averaged over N anchor–positive pairs.

Unigram Jaccard similarity. Let $\mathcal{G}^{(1)}(\mathcal{T})$ denote the set of unique tokens appearing in sequence $\mathcal{T}$. We compute

$$\mathrm{Jaccard}(x_i, x_i^+) = \frac{\left|\mathcal{G}^{(1)}(\mathcal{T}_i) \cap \mathcal{G}^{(1)}(\mathcal{T}_i^+)\right|}{\left|\mathcal{G}^{(1)}(\mathcal{T}_i) \cup \mathcal{G}^{(1)}(\mathcal{T}_i^+)\right|}. \tag{38}$$

This measures unordered token-identity overlap and ignores order and repeated occurrences.

Normalized edit similarity. Let $ED(\mathcal{T}_i, \mathcal{T}_i^+)$ be the Levenshtein edit distance. We define

$$\mathrm{Sim}_{\mathrm{edit}}(\mathcal{T}_i, \mathcal{T}_i^+) = 1 - \frac{ED(\mathcal{T}_i, \mathcal{T}_i^+)}{\max\left\{|\mathcal{T}_i|, |\mathcal{T}_i^+|\right\}}. \tag{39}$$

This penalizes substitutions, insertions, deletions, ordering differences, and length mismatch.

Exact match rate. We report

$$\mathrm{ExactMatch} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\left[\mathcal{T}_i = \mathcal{T}_i^+\right]. \tag{40}$$

Mean sequence length. We report the average decoded length over both streams:

$$\overline{L} = \frac{1}{2N}\sum_{i=1}^{N}\left(|\mathcal{T}_i| + |\mathcal{T}_i^+|\right). \tag{41}$$

Edit-operation decomposition. For an optimal Levenshtein alignment, let S_i, I_i, and D_i denote the numbers of substitutions, insertions, and deletions. Then

$$ED(\mathcal{T}_i, \mathcal{T}_i^+) = S_i + I_i + D_i. \tag{42}$$

We report these counts to separate token relabeling from token insertion or deletion.

F.2 Token Inventory and Collapse Analysis

Let $\mathcal{D}_{\text{eval}}$ be an evaluation set and let $\mathcal{T}_i = [\tau_{i,1}, \ldots, \tau_{i,L_i}]$ denote the decoded token sequence for example i. Let $\mathcal{A}$ be the tokenizer vocabulary.

Global token distribution. The empirical token frequency is

$$p(a) = \frac{\sum_i \sum_{l=1}^{L_i} \mathbb{1}[\tau_{i,l} = a]}{\sum_i L_i}, \qquad a \in \mathcal{A}. \tag{43}$$

We compute global entropy and normalized entropy as

$$H_{\text{global}} = -\sum_{a \in \mathcal{A}} p(a) \log p(a), \qquad \widetilde{H}_{\text{global}} = \frac{H_{\text{global}}}{\log |\mathcal{A}|}. \tag{44}$$

The effective vocabulary size is

$$V_{\text{eff}} = \exp(H_{\text{global}}). \tag{45}$$

Active vocabulary and concentration. The active vocabulary size is

$$V_{\text{active}} = |\{a \in \mathcal{A} : p(a) > 0\}|, \tag{46}$$

and the dead-token rate is

$$r_{\text{dead}} = 1 - \frac{V_{\text{active}}}{|\mathcal{A}|}. \tag{47}$$

For the top-q most frequent tokens, we report

$$M_q = \sum_{a \in \text{Top}q} p(a). \tag{48}$$

In the main results, we use this statistic for top-token concentration.

Native-position token usage. For absolute output position l, define

$$\mathcal{I}_l = \{i : L_i \geq l\}. \tag{49}$$

The token distribution at position l is

$$p_l(a) = \frac{\sum_{i \in \mathcal{I}_l} \mathbb{1}[\tau_{i,l} = a]}{|\mathcal{I}_l|}. \tag{50}$$

We compute

$$H_l = -\sum_{a \in \mathcal{A}} p_l(a) \log p_l(a), \qquad \widetilde{H}_l = \frac{H_l}{\log |\mathcal{A}|}. \tag{51}$$

Relative-position token usage. To compare tokenizers with different sequence lengths, each position l in sequence i is assigned to a relative-position bin

$$b(l, i) = \left\lfloor B_{\text{pos}} \frac{l-1}{L_i} \right\rfloor, \tag{52}$$

where B_{pos} is the number of bins. For each bin b, we compute the corresponding token distribution $p_b(a)$, entropy H_b, and normalized entropy $\widetilde{H}_b$ using the same definitions as above.

Low-diversity collapse. For a decoded sequence $\mathcal{T}$,we define the unique-token ratio

$$r_{\text{uniq}}(\mathcal{T}) = \frac{|\text{unique}(\mathcal{T})|}{|\mathcal{T}|}. \tag{53}$$

A sequence is marked low-diversity collapsed when $r_{\text{uniq}}(\mathcal{T}) \leq 0.2$.

For anchor and positive streams, we report the fraction of decoded sequences satisfying this criterion. We also report the collapsed-pair rate:

$$\text{CollapsedPair} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\left[r_{\text{uniq}}(\mathcal{T}_i) \leq 0.2 \;\text{ or }\; r_{\text{uniq}}(\mathcal{T}_i^{+}) \leq 0.2\right]. \tag{54}$$

Within-stream exact collisions. For a set of decoded sequences $\mathcal{S} = \{\mathcal{U}_i\}_{i=1}^{N}$, the within-stream exact-collision rate is

$$\text{Collision}(\mathcal{S}) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\left[\exists j \neq i : \mathcal{U}_i = \mathcal{U}_j\right]. \tag{55}$$

We report this quantity separately for the anchor stream $\mathcal{S}_{\text{anchor}} = \{\mathcal{T}_i\}_{i=1}^{N}$ and the positive stream $\mathcal{S}_{\text{positive}} = \{\mathcal{T}_i^{+}\}_{i=1}^{N}$. These metrics distinguish desirable anchor–positive robustness from undesirable many-to-one assignment across distinct examples.

F.3 Long-Form Segment Retrieval on Continuous Audio

This appendix gives only the ranking and compactness metrics used for the continuous-audio retrieval experiments. The archive construction, query construction, relevance notions, and token-space ranking protocol are defined in the main text.

Recall@K. Let $\mathcal{R}_i$ be the set of relevant archive indices for query i, and let $\text{TopK}(i)$ be the top-K retrieved archive indices. We compute

$$\text{Recall@}K = \frac{1}{Q}\sum_{i=1}^{Q} \mathbb{1}\left[\mathcal{R}_i \cap \text{TopK}(i) \neq \emptyset\right]. \tag{56}$$

Mean reciprocal rank. Let r_i be the rank of the first relevant archive segment for query i. We compute

$$\text{MRR} = \frac{1}{Q}\sum_{i=1}^{Q} \frac{1}{r_i}. \tag{57}$$

Mean first relevant rank. We report

$$\text{MeanFRR} = \frac{1}{Q}\sum_{i=1}^{Q} r_i. \tag{58}$$

Archive token count and token rate. For archive segment n, let $L_n = |\mathcal{T}_n|$. The total archive token count and average segment length are

$$N_{\text{tok}} = \sum_{n=1}^{N} L_n, \qquad \overline{L} = \frac{1}{N}\sum_{n=1}^{N} L_n. \tag{59}$$

For segment duration $T_{\text{ctx}} = 3\,\text{s}$, the token rate is

$$R_{\text{tok}} = \frac{\overline{L}}{T_{\text{ctx}}}. \tag{60}$$

Compression and relative token reduction. Given a baseline token count $N_{\text{tok}}^{\text{base}}$ and a PairAlign token count $N_{\text{tok}}^{\text{pa}}$, we compute

$$C_{\text{tok}} = \frac{N_{\text{tok}}^{\text{base}}}{N_{\text{tok}}^{\text{pa}}}, \qquad r_{\text{red}} = 1 - \frac{N_{\text{tok}}^{\text{pa}}}{N_{\text{tok}}^{\text{base}}}. \tag{61}$$

We also report the minimum and maximum segment lengths and the serialized archive size in the implemented cache format.

F.4 Continuous-Sweep Tokenization Analysis

This appendix defines only sweep-specific quantities. Unigram Jaccard similarity, normalized edit similarity, mean sequence length, edit distance, and the substitution/insertion/deletion decomposition are defined in Appendix F.1.

For the m-th 3 s sweep window, let $\mathcal{T}^{(m)} = [\tau_1^{(m)}, \ldots, \tau_{L^{(m)}}^{(m)}]$ be the decoded non-special token sequence, with length $L^{(m)} = |\mathcal{T}^{(m)}|$. The adjacent sequence $\mathcal{T}^{(m+1)}$ is decoded independently from the next sweep window, whose start time is shifted by 100 ms.

Adjacent length change. For each adjacent pair, we compute

$$|\Delta L|^{(m)} = \left| L^{(m+1)} - L^{(m)} \right|. \tag{62}$$

Length ratio. We report the adjacent length ratio

$$\rho_L^{(m)} = \frac{\min\left(L^{(m)}, L^{(m+1)}\right)}{\max\left(L^{(m)}, L^{(m+1)}\right)}. \tag{63}$$

Relative length change. The relative length change reported in Table 5 is

$$\frac{\frac{1}{N_{\text{pairs}}} \sum_m |\Delta L|^{(m)}}{\frac{1}{N_{\text{pairs}}} \sum_m L^{(m)}}. \tag{64}$$

Edit-operation rates. For each adjacent pair, let $S^{(m)}$, $I^{(m)}$, and $D^{(m)}$ denote the substitution, insertion, and deletion counts in the optimal Levenshtein edit script. We report length-normalized operation rates:

$$r_S^{(m)} = \frac{S^{(m)}}{\max\left(L^{(m)}, L^{(m+1)}\right)}, \qquad r_I^{(m)} = \frac{I^{(m)}}{\max\left(L^{(m)}, L^{(m+1)}\right)}, \tag{65}$$

$$r_D^{(m)} = \frac{D^{(m)}}{\max\left(L^{(m)}, L^{(m+1)}\right)}. \tag{66}$$

Distributional summaries. All sweep statistics are aggregated over adjacent-window pairs. We report means, standard deviations, medians, and thresholded fractions for the reported quantities in Tables 5, 6, and 7.

F.5 Architecture and Training Details

Table 10 reports the architecture, optimization, decoding, sampling, and implementation details for the controlled PairAlign experiments. These details are deferred from the main text to preserve readability while maintaining reproducibility.

Component	Configuration
Input representation	Audio is sampled at 16 kHz and converted to 96-bin log-Mel spectrograms using a 25 ms analysis window and a 10 ms hop. Training uses 3.0 s segments.
Training corpus	Models are trained on segments sampled from LibriSpeech `train-clean-360` and `train-other-500`, an 860 h pool. Exposure is reported as sampled anchor-segment hours, since windows are repeatedly drawn from this finite pool.
Stage I encoder	Mamba2 contextual speech encoder with input projection $96 \rightarrow 512$, 8 Mamba2 layers, state dimension 128, convolution width 4, expansion factor 8, and head dimension 48. Output representation is L-2 normalized.
Stage I VQ tokenizer	Nearest-centroid vector quantizer with 512 codewords of dimension 512. Consecutive duplicate VQ assignments are removed by run-length deduplication before sequence comparison and before using Stage I strings as Stage II targets.
Stages II–III decoder	Whisper-style autoregressive Transformer decoder with context length 512, hidden dimension 512, 8 attention heads, 6 decoder layers, and dropout 0.1. The decoder vocabulary consists of the Stage I VQ alphabet plus BOS, EOS, PAD, and MASK.
Acoustic conditioning	Each decoder block contains causal self-attention, cross-attention to encoder states, and a feed-forward sublayer. Encoder-summary conditioning is added by mean-pooling encoder states, applying layer normalization, projecting to the decoder dimension, and adding the summary vector to token embeddings at every decoder position.
Optimization	Adam optimizer with weight decay 10^{-4} and learning-rate scheduling. The decoder/base learning rate is 5×10^{-5}. During Stage III encoder–decoder training, the encoder learning rate is 5×10^{-6}, i.e., $10\times$ smaller than the decoder learning rate, so that the acoustic representation is updated more conservatively.
Stage I objective and training	Stage I uses paired augmented views with in-batch negatives, optimizing frame-level contrastive learning together with VQ commitment and code-prediction terms. Loss weights are 1.0 for the contrastive loss, 1.0 for the VQ loss. The codebook is updated by EMA with decay 0.7, and the commitment loss weight is $\beta = 0.1$. Training uses 10 epochs, 5000 optimization steps per epoch, and batch size 16, corresponding to approximately 667 sampled anchor-segment hours.
Stage II objective and training	Stage II trains the autoregressive decoder by cross-paired teacher forcing on deterministic Stage I VQ-derived strings. The encoder and symbolic alphabet are inherited from Stage I. Training uses 6 epochs, 5000 optimization steps per epoch, and batch size 16, corresponding to approximately 400 sampled anchor-segment hours.
Stage III objective and training	Stage III replaces fixed Stage I targets with adaptive EMA-teacher tokenizations and trains the full encoder–decoder tokenizer with paired self-alignment, hard in-batch likelihood contrast, entropy regularization, repetition-aware target generation, and explicit length constraints. It uses the same duration as Stage II, corresponding to approximately 400 sampled anchor-segment hours. The EMA teacher is updated every 1000 optimization steps with decay 0.995.
EMA-teacher sampling	Stage III teacher targets are generated by scheduled top-p sampling with length-cap ratio $\rho = 0.15$. Global parameters are top-$p = 0.88$, temperature 0.90, and repetition penalty $\gamma_{\text{rep}} = 1.15$. For the first 4 decoding steps, top-p is annealed from 0.97 to 0.88, temperature from 1.30 to 1.00, and repetition penalty from 1.03 to 1.30.
Inference decoding	Final tokenization uses deterministic beam search with beam size 4, length-cap ratio $\rho = 0.15$, minimum cap 2, repetition penalty $\gamma_{\text{rep}} = 1.30$, and length-normalization exponent $\alpha_{\text{len}} = 0.0$. Special symbols are excluded from repetition counting where applicable.
Anti-bypass and anti-collapse schedules	Prefix masking and structured self-attention dropout are used to reduce teacher-forcing bypass through token-prefix continuation. Self-attention dropout is annealed from 0.6 to 0.2 in Stage II and from 0.2 to 0.1 in Stage III. Prefix masking is annealed from 0.6 to 0.3 in Stage II and from 0.3 to 0.1 in Stage III. Stage III additionally uses hard in-batch likelihood contrast, entropy regularization, repetition-aware teacher sampling, and length-constrained decoding to discourage many-to-one or generic-string collapse.

Table 10: Architecture, optimization, training, sampling, and decoding details for the controlled PairAlign experiments.

G Edit, Inventory, and Position-Entropy Diagnostics

This appendix provides the diagnostic plots summarized in Section 5.1. The goal is to check whether the compact PairAlign strings remain meaningful symbolic sequences rather than arising from unstable edit behavior, narrow vocabulary use, generic prefixes, or position-wise entropy collapse. The diagnostics are complementary: edit operations test how anchor–positive strings differ, global inventory statistics test vocabulary concentration, and position-wise entropy tests whether token usage remains diverse across the decoded trajectory.

Edit-operation decomposition. Figure 2 decomposes normalized edit behavior into absolute substitutions, insertions, and deletions. This distinguishes genuine symbolic stability from a degenerate shortening strategy: a compact tokenizer should not obtain lower edit distance merely by deleting unstable regions or by creating large insertion/deletion imbalance across paired views.

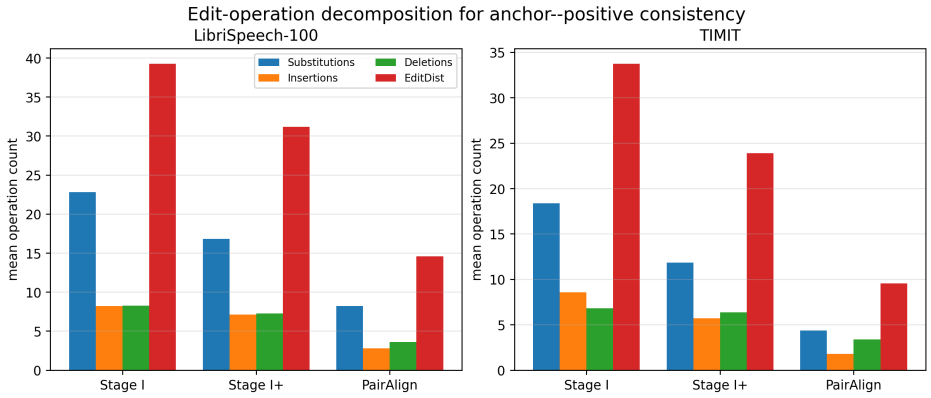


Figure 2: Edit-operation decomposition for anchor–positive token consistency. PairAlign requires fewer absolute edit operations across substitutions, insertions, and deletions, indicating that compactness is not obtained through unstable token birth/death.

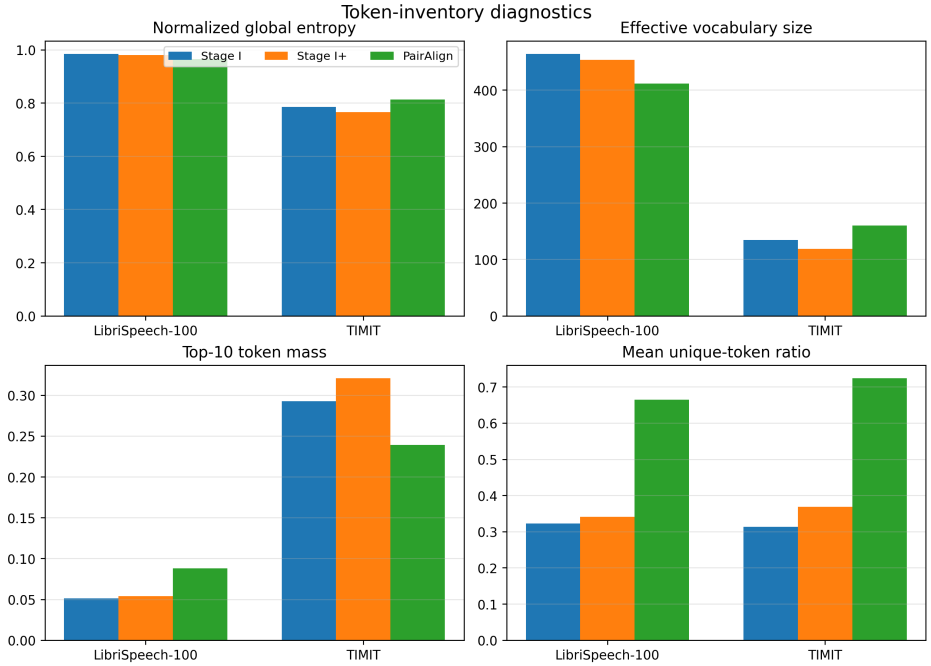


Figure 3: Global token-inventory diagnostics. PairAlign remains broad-vocabulary rather than collapsed: it uses the full vocabulary on LibriSpeech-100 and has higher normalized entropy and effective vocabulary than the geometric systems on TIMIT despite shorter sequences.

Global token-inventory usage. Figure 3 tests whether compactness is achieved without vocabulary collapse. High paired-view consistency is only meaningful if the tokenizer remains input-dependent rather than assigning generic strings to many examples. We therefore report normalized global token entropy, effective vocabulary size, top-10 token mass, and mean unique-token ratio, which jointly measure whether token usage is broad or concentrated in a small dominant set.

Native-position token entropy. Figure 4 measures token entropy at each absolute decoded position. These plots are shown at each model's own output length scale, because the geometric tokenizers produce longer deduplicated frame-derived streams while PairAlign produces shorter autoregressive strings. The

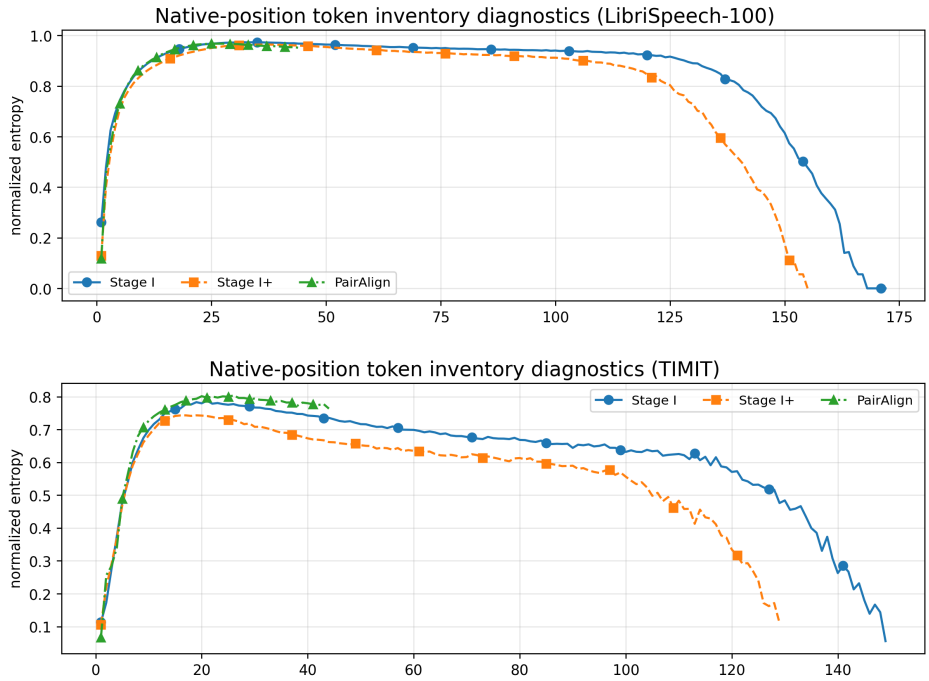


Figure 4: Native-position token-entropy diagnostics on LibriSpeech-100 and TIMIT. The top panel shows LibriSpeech-100 and the bottom panel shows TIMIT. Models are shown at their own absolute output-length scales.

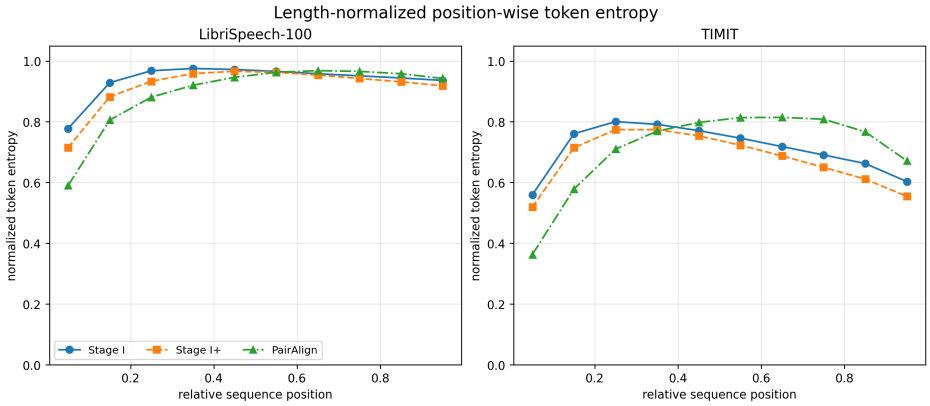


Figure 5: Length-normalized position-wise token entropy. Relative-position bins compare the beginning, middle, and end of each sequence independently of absolute sequence length.

diagnostic tests whether early decoded positions become generic or low-entropy in a way that would indicate prefix collapse.

Length-normalized position-wise token entropy. Figure 5 complements the native-position view by mapping each decoded sequence to a common beginning–middle–end axis. This removes the confound of absolute sequence length and tests whether diversity is distributed across the compact sequence rather than concentrated only in a late suffix.

Variant	R@1	R@5	R@10	R@20	MRR	Mean FRR
Stage I Geometric	0.75	0.83	0.85	0.87	0.78	36.40
Stage I+ Geometric	0.67	0.75	0.79	0.81	0.71	51.05
Full PairAlign	0.71	0.79	0.80	0.82	0.74	53.99
w/o encoder-summary conditioning	0.71	0.80	0.85	0.88	0.75	28.82

(a) Retrieval ablation on the TIMIT archive under segment-overlap relevance.

Variant	Avg. Length	Token Rate
Stage I Geometric	84.62	28.21 tok/s
Stage I+ Geometric	61.15	20.38 tok/s
Full PairAlign	38.13	12.71 tok/s
w/o encoder-summary conditioning	40.11	13.37 tok/s

(b) Compactness of ablated tokenizers on the TIMIT retrieval archive.

Table 11: Retrieval and compactness ablations on the TIMIT archive. Top: segment-overlap retrieval. Bottom: average token sequence length and token rate on the same archive.

Dataset	Variant	Jaccard Similarity	Edit Similarity	Exact Match	Active Vocabulary	Collapsed Pair	Exact Collision
LibriSpeech-100	Stage I Geometric	0.718	0.609	0.264	512	0.0500	0.0/0.0
	Stage I+ Geometric	0.738	0.629	0.265	512	0.0450	0.0/0.0
	PairAlign	0.719	0.630	0.291	512	0.0000	0.0001/0.0001
Shrutilipi-Tamil	Stage I Geometric	0.6937	0.5867	0.2425	512	0.0843	0.0/0.0
	Stage I+ Geometric	0.6991	0.5995	0.2421	512	0.1051	0.0/0.0
	PairAlign	0.7194	0.6295	0.2905	512	0.0000	0.0/0.0

Table 12: Tamil token consistency under content-preserving perturbations.

H Retrieval and Compactness Ablations

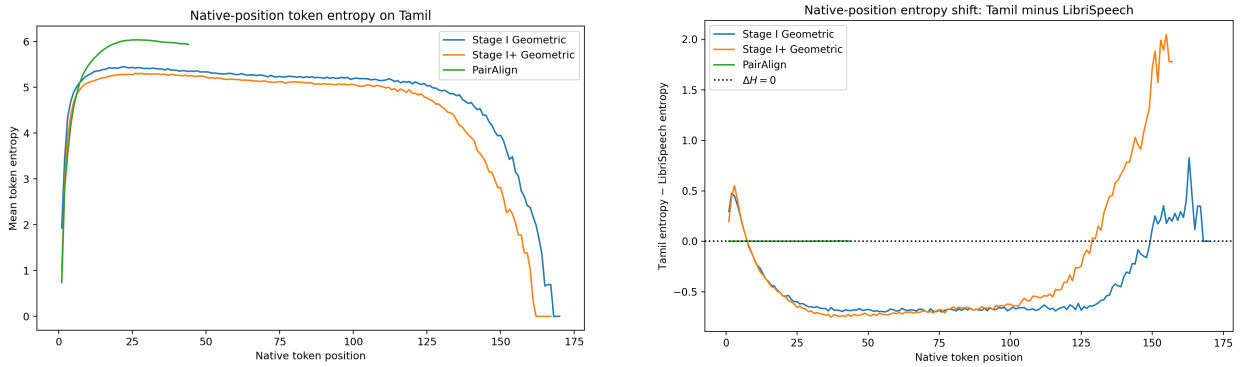
This appendix reports the retrieval and compactness ablations summarized in Section 5.4. Retrieval is evaluated on the TIMIT archive under segment-overlap relevance using normalized edit distance in token space. Stage II variants are omitted from retrieval because Stage II is a fixed-target bridge; the Stage II variant without structured self-attention dropout collapses in Table 8, making edit-distance retrieval non-discriminative.

Interpretation. The retrieval ablation separates archive discrimination from paired-view stability. Stage I+ improves paired-view consistency over Stage I but retrieves worse, showing that sequence consistency and segment-overlap retrieval are not identical objectives. The no-summary PairAlign variant retrieves more sharply than Full PairAlign, but its paired-view edit similarity is much lower (Table 8). This indicates that removing the summary pathway increases sensitivity to local acoustic differences, which can help edit-distance archive discrimination while reducing consistency under content-preserving perturbations.

The compactness results show that this stability gap is not explained by a large change in symbolic rate. Full PairAlign and the no-summary variant both remain low-rate autoregressive tokenizers, while Stage I and Stage I+ produce longer frame-derived traces. Full PairAlign is therefore selected as the more stable operating point: it preserves compactness and meaningful retrieval while substantially improving ordered paired-view consistency.

I Probing the Learned Similarity Geometry Under Lexically Disjoint Tamil Transfer

Lexically disjoint transfer setting. The Shrutilipi-Tamil experiment tests whether the similarity relation induced by each LibriSpeech-trained tokenizer remains meaningful outside English. This is not a component ablation: Stage I Geometric, Stage I+ Geometric, and PairAlign are evaluated unchanged. The goal is not word-level cross-lingual matching, but robustness of the learned token-space geometry under a different phonetic, phonotactic, prosodic, and morphosyntactic distribution. Consistency, collapse, native-



(a) Native-position token entropy on Tamil across the three tokenizers.

(b) Native-position entropy shift, $DeltaH_m = H_m^{\text{Tamil}} - H_m^{\text{LibriSpeech}}$.

Figure 6: Native-position entropy analysis under the Tamil linguistic-shift setting. Left: token entropy across native decoding positions on Tamil. Right: position-wise entropy residual relative to LibriSpeech.

Tokenizer	**Mean** ΔH	**Mean** $\lvert\Delta H\rvert$	**Max** $\lvert\Delta H\rvert$
Stage I Geometric	-0.454	0.539	0.827
Stage I+ Geometric	-0.270	0.653	2.047
PairAlign	0.001	0.001	0.003

Table 13: Native-position entropy shift between Tamil and LibriSpeech, where $\Delta H_m = H_m^{\text{Tamil}} - H_m^{\text{LibriSpeech}}$.

position entropy, retrieval, and compactness are reported in Table 12, Figure 6, and Tables 13, 14a, and 14b.

PairAlign preserves its paired-view geometry under Tamil shift. Table 12 shows that PairAlign transfers with essentially no loss in paired-view consistency. On Tamil, it obtains Jaccard 0.7194, edit similarity 0.6295, and exact match 0.2905, nearly matching its LibriSpeech-100 values of 0.719, 0.630, and 0.291. It also uses the full 512-token vocabulary, has zero measured collapsed-pair rate, and has no within-stream exact collisions. Thus, the result is not explained by generic strings, narrow vocabulary use, or many-to-one assignment; PairAlign preserves a compact ordered acoustic-symbolic similarity relation under lexically disjoint transfer.

Geometric tokenizers transfer but are more sensitive to local shift. The geometric systems remain usable on Tamil, but degrade more clearly from LibriSpeech. Stage I drops from 0.718 to 0.6937 in Jaccard and from 0.609 to 0.5867 in edit similarity; Stage I+ drops from 0.738 to 0.6991 and from 0.629 to 0.5995. This should not be interpreted as lexical coding. Their symbols remain tied to local VQ partitions learned on English speech, so Tamil phone distributions, transitions, syllabic patterns, prosody, and morphophonology can shift frame representations across centroid boundaries. Both geometric systems activate the full vocabulary, yet their Tamil collapsed-pair rates increase to 0.0843 and 0.1051, showing that active vocabulary alone is not a sufficient non-collapse diagnostic.

Position-wise inventory usage is stable for PairAlign. Figure 6 and Table 13 show that PairAlign's native-position token uncertainty is nearly unchanged between LibriSpeech and Tamil: mean entropy residual 0.001, mean absolute shift 0.001, and maximum absolute shift 0.003. Stage I and Stage I+ show much larger shifts, with mean residuals -0.454 and -0.270, indicating more concentrated position-wise token usage on Tamil. This diagnostic does not imply that English and Tamil share linguistic structure; it measures inventory stability over populated decoded positions. Under that criterion, PairAlign preserves both full vocabulary use and position-wise uncertainty.

Variant	R@1	R@5	R@10	R@20	MRR	Mean FRR
Stage I Geometric	0.87	0.93	0.96	0.97	0.90	11.93
Stage I+ Geometric	0.85	0.90	0.91	0.93	0.87	9.19
PairAlign	0.68	0.79	0.84	0.87	0.73	63.60

(a) Retrieval under segment-overlap relevance.

Variant	Avg. Len.	Tok/s
Stage I Geometric	113.15	37.72
Stage I+ Geometric	101.15	33.72
PairAlign	37.93	12.64

(b) Token-stream compactness.

Table 14: Tamil retrieval and compactness comparison. Left: retrieval on the Tamil archive under segment-overlap relevance. Right: average sequence length and token rate on the same archive.

Tamil retrieval follows the same rate dependence. Table 14a shows the complementary effect of token granularity under segment-overlap relevance. Stage I reaches Recall@1/MRR 0.87/0.90, Stage I+ reaches 0.85/0.87, and PairAlign reaches 0.68/0.73. This is consistent with normalized edit-distance retrieval and the token rates in Table 14b: Stage I and Stage I+ emit 113.15 and 101.15 tokens per 3 s Tamil segment, whereas PairAlign emits 37.93. Longer frame-derived geometric tkenizer traces retain more local acoustic variation, and a modest reduction in paired-view invariance can help overlap-based retrieval by preserving evidence that distinguishes nearby archive segments. PairAlign represents the complementary lower-rate operating point: it substantially compresses the symbolic sequence while preserving a stable paired-view similarity relation and non-collapsed transfer behavior.

Consistency, retrieval, and symbolic rate separate cleanly. The Tamil results reinforce that paired-view stability, retrieval sharpness, and symbolic density are distinct. Stage I+ improves Tamil edit consistency over Stage I, but Stage I retrieves slightly better, showing that sequence consistency and archive discrimination are not identical objectives. PairAlign keeps the same compact operating regime across languages: 12.64 tok/s on Tamil versus 12.71 tok/s on TIMIT, with comparable retrieval behavior. Its Tamil behavior is therefore not driven by a token-density shift; it preserves a lower-rate retrieval operating point while maintaining stronger consistency stability under lexical shift.

J PairAlign Versus Pretrained Semantic Tokenizers

Pretrained SSL tokenizers as strong geometric references. HuBERT+VQ and WavLM+VQ are strong frame-synchronous semantic tokenizers. As shown in Table 15a, they discretize pretrained SSL features using $k = 512$ KMeans units followed by consecutive duplicate removal. Their strength comes from the pretrained encoders: HuBERT uses masked prediction of cluster targets (Hsu et al., 2021), while WavLM adds denoising-oriented pretraining and gated relative position bias (Chen et al., 2022). Thus, contextual sequence information is already present in the continuous SSL representation before discretization, but token length and ordering remain tied to the frame-synchronous feature stream. PairAlign tests a different regime, where compact token sequences, adaptive length, and termination are learned through paired sequence likelihood.

Consistency reflects stronger pretrained feature spaces. Table 15a shows that HuBERT+VQ and WavLM+VQ obtain stronger average overlap than PairAlign. On TIMIT, HuBERT/WavLM reach Jaccard 0.767/0.773 and edit similarity 0.726/0.730, compared with PairAlign's 0.753 and 0.691; on Tamil, WavLM reaches 0.766 and 0.718, compared with PairAlign's 0.7194 and 0.6295. This is expected: large pretrained SSL encoders provide a stronger contextual feature space for frame-synchronous quantization. PairAlign tests a different axis: whether a controlled in-pipeline geometric prior can be converted into a compact ordered sequence tokenizer through paired autoregressive self-alignment.

Exact string stability favors PairAlign. Although SSL tokenizers have higher average Jaccard and edit similarity, PairAlign obtains the highest exact anchor–positive string match on both datasets: 0.301 on TIMIT and 0.2905 on Tamil, compared with 0.263/0.264 for HuBERT/WavLM on TIMIT and 0.255/0.262 on Tamil. Exact match is a stricter compact-string criterion: the entire decoded string must remain unchanged under a content-preserving perturbation. This separates dense local robustness from string-level stability.

Dataset	Variant	Jaccard Similarity	Edit Similarity	Exact Match	Active Vocabulary	Collapsed Pair	Exact Collision
TIMIT	HuBERT	0.767	0.726	0.263	464	0.0000	0.0/0.0
	WavLM	0.773	0.730	0.264	397	0.0000	0.0/0.0
	PairAlign	0.753	0.691	0.301	430	0.0000	0.0004/0.0028
Shrutilipi-Tamil	HuBERT	0.726	0.681	0.255	460	0.0000	0.0/0.0
	WavLM	0.766	0.718	0.262	340	0.0000	0.0/0.0
	PairAlign	0.7194	0.6295	0.2905	512	0.0000	0.0/0.0

(a) English and Tamil paired-view consistency.

Variant	R@1	R@5	R@10	R@20	MRR	Mean FRR
HuBERT	0.943	0.977	0.990	0.997	0.961	1.29
WavLM	0.977	0.997	1.000	1.000	0.985	1.05
PairAlign	0.710	0.790	0.800	0.820	0.740	53.99

(b) TIMIT segment-overlap retrieval.

Variant	Avg. Len.	Tok/s
HuBERT	86.97	28.99
WavLM	96.17	32.06
PairAlign	38.13	12.71

(c) Token-stream compactness.

Table 15: Comparison against pretrained SSL-based semantic tokenizers. Top: paired-view consistency on TIMIT and Shrutilipi-Tamil. Bottom left: TIMIT segment-overlap retrieval. Bottom right: token-stream compactness on the TIMIT retrieval archive. Avg. Len. is the average number of tokens per 3 s segment, and Tok/s is the symbolic token rate.

PairAlign preserves less frame-level evidence, but more often maps paired views to the same compact ordered symbolic sequence.

Retrieval benefits from high-rate pretrained geometry. Table 15b shows that HuBERT+VQ and WavLM+VQ are near-saturated on TIMIT segment-overlap retrieval, with Recall@1 0.943/0.977 and MRR 0.961/0.985, compared with PairAlign's Recall@1 0.710 and MRR 0.740. This reflects both representation scale and token rate. HuBERT and WavLM are pretrained on much larger speech corpora with objectives that learn contextual, discriminative acoustic representations before discretization. After frame-synchronous VQ, these representations yield dense token strings that preserve substantial local evidence for distinguishing nearby archive segments. Table 15c shows the corresponding rate difference: HuBERT and WavLM emit 86.97 and 96.17 tokens per 3 s segment, whereas PairAlign emits 38.13. Normalized edit-distance retrieval therefore favors the pretrained SSL tokenizers because their longer, pretrained, frame-derived strings provide more locally discriminative evidence for top-rank archive matching.

Controlled baselines versus external references. HuBERT+VQ and WavLM+VQ are useful external references, but not controlled PairAlign ablations. They differ in encoder architecture, pretraining objective, optimization scale, and effective data exposure. In contrast, PairAlign, Stage I, and Stage I+ use the specified LibriSpeech training pool, with approximately 667 sampled segment-hours for Stage I and 400 sampled segment-hours each for Stages II and III. Stage I and Stage I+ therefore remain the controlled in-pipeline baselines: they share PairAlign's preprocessing, encoder family, VQ alphabet, training regime, and evaluation protocol.

Scaling path. The SSL results suggest an extension rather than a contradiction. PairAlign does not require Stage I to be a modest VQ encoder trained from scratch. A pretrained semantic tokenizer such as HuBERT+VQ or WavLM+VQ could serve as the Stage I teacher, after which the same Stage II autoregressive bridge and Stage III EMA self-alignment could learn compact PairAlign-style tokenizations. This would combine stronger SSL features with PairAlign's sequence-level mechanism for compact ordering, learned length, termination, and paired-view agreement.