Cross-Modal-Domain Generalization Through Semantically Aligned Discrete Representations

Supplementary Material

8. Notation Library

Conventions: $[\cdot;\cdot]$: channel-wise concatenation; $\|\cdot\|_2$: ℓ_2 norm; $\mathrm{sg}[\cdot]$: stop–gradient operator; $\mathbb{1}[\cdot]$: indicator function;

Modalities, indices, sizes: $m \in \{a, v, t\}$: modality (audio, video, text); $m_1 \neq m_2$: CMG train/test modalities; i: sample index; $t \in \{1{:}T\}$: time index; N: number of samples; T: timesteps per sample; D: embedding dimensionality; K: number of codewords per codebook; B: batch size; H: CPC prediction horizon;

Data and encoders: $\mathbf{X} = \{(x_i^a, x_i^v, x_i^t)\}_{i=1}^N$: paired multimodal dataset; $\mathbf{x}_i^m$: input of modality m for sample i; $\mathbf{y}_i^m$: label of modality m for sample i; $\Psi^m(\cdot)$: modality-specific encoder; $z_i^m = \Psi^m(x_i^m) \in \mathbb{R}^{T\times D}$: encoder embedding sequence; $z_{i,t}^m \in \mathbb{R}^D$: encoder embedding frame at timestep t;

Quantization and codebooks: $\mathrm{VQ}(\cdot)$: vector quantizer; $\mathbf{E}_m = \{\mathbf{e}_m(k)\}_{k=1}^K \in \mathbb{R}^{K\times D}$: codebook of modality m; $\mathbf{e}_m(k) \in \mathbb{R}^D$: codeword k of modality m; $k_{i,t} = \arg\min_j \|z_{i,t}^m - \mathbf{e}_m(j)\|_2^2$: nearest-neighbour codeword index; $\hat{z}_{i,t}^m = \mathrm{VQ}(z_{i,t}^m) = \mathbf{e}_m(k_{i,t})$: quantized latent at timestep t; $\mathbf{E} = [\mathbf{E}_a; \mathbf{E}_v; \mathbf{E}_t]$: concatenated unified vocabulary;

Decoders, projections, heads: $P_m(\cdot)$: modality-specific projection; $\mathcal{D}_m(\cdot)$: reconstruction decoder for modality m; $\mathcal{G}(\cdot)$: modality-invariant downstream task head; $\mathcal{E}_{\mathcal{L}}(\cdot)$: downstream evaluation loss;

CPC (cross-modal information maximization): (a, b): ordered modality pair (predict b from a); c_t^a: context from uniLSTM over modality a at time t; W_h^a: step-specific linear map at horizon step h for modality a; z_{t+h}^b: future latent of modality b at offset h; $\{z_j^b\}$: negatives drawn from modality b; H: number of future steps predicted;

DQA (Discrete Quantization Alignment): $p_m^i \in \mathbb{R}^K$: sequence-level code-usage distribution for sample embedding z_i^m and modality m; $\mathcal{L}_{\mathrm{CMCM}}$ [22]: cross-modal code-matching loss aligning $\{p_m^i\}$ across modalities;

$$\mathcal{L}_{\mathrm{CMCM}} = -\frac{1}{N}\sum_i \log \frac{\exp\big(\langle p_a^i, \log p_b^i\rangle + \langle p_b^i, \log p_a^i\rangle\big)}{\sum_j \exp\big(\langle p_a^i, \log p_b^j\rangle + \langle p_b^j, \log p_a^i\rangle\big)}, \tag{11}$$

where $p_m^i \in \Delta^{K-1}$ is the time-averaged soft assignment over indices $k = 1..K$ for modality m and embedding sample i.

DTA (Discrete Temporal Alignment with EMA): $\pi_m(i,t)$: hard nearest-neighbour codeword index for (i,t) in modality m; $q_{m,i,t}(k) \in \{0,1\}$: one-hot responsibility that codeword k was selected at (i,t); $\{n,p\} = \{a,v,t\} \setminus \{m\}$: the two modalities other than m; $\mathbf{U}_m^{\mathrm{self}}(k)$: accumulator of modality-m features assigned to k; $\mathbf{U}_m^{\mathrm{cross}}(k)$: time-aligned accumulator of (n,p) features assigned to k; $\lambda_{\mathrm{self}} = 0.6$: self-term mixing weight; $\lambda_{\mathrm{cross}} = 0.2$: cross-term mixing weight (per other modality); $\rho \in (0,1)$: EMA decay factor; $\varepsilon > 0$: numerical stabilizer; $\mathbf{H}_m(k)$: mixed accumulator for k (self + cross); $\mathbf{S}_m(k)$: EMA-weighted sum of assigned features for k; $C_m(k)$: EMA-weighted assignment count for k; $\mathbf{S}_m^{\mathrm{new}}(k)$, $C_m^{\mathrm{new}}(k)$: updated EMA statistics for k; $\mathbf{e}_m^{\mathrm{new}}(k)$: updated centroid/codeword at index k;

CSA (Cascading Semantic Alignment): $\mathbf{e}_v^0(k)$, $\mathbf{e}_a^0(k)$, $\mathbf{e}_t^0(k)$: post-DTA (pre-cascade) centroids for video, audio, text at index k; $\mathbf{c}^0(k) = \frac{1}{3}(\mathbf{e}_v^0(k) + \mathbf{e}_a^0(k) + \mathbf{e}_t^0(k))$: cross-modal semantic anchor at index k; $\mathbf{e}_t^1(k)$, $\mathbf{e}_a^1(k)$, $\mathbf{e}_v^1(k)$: centroids after cascade (applied in order $t \to a \to v$); ^{0}, ^{1}: superscripts denoting pre-/post-cascade values within the same iteration; k: codeword index shared across modalities;

Reconstruction: $\hat{z}_i^m \in \mathbb{R}^{T\times D}$: quantized sequence for modality m; $\hat{z}_i^{\mathrm{tri}} = [\hat{z}_i^a; \hat{z}_i^v; \hat{z}_i^t] \in \mathbb{R}^{T\times 3D}$: tri-modal quantized code; $\tilde{\mathbf{x}}_i^m = \mathcal{D}_m([\hat{z}_i^{\mathrm{tri}}; P_m(\mathbf{x}_i^m)])$: reconstruction of modality m for sample i;

Other: N_{re}: reset threshold for dead codes (consecutive batches unselected);

9. Implementation Details

9.1. Pretraining Setup

Backbone features: Following [32], for every 1 s video segment, we sample 16 RGB frames and extract pool5 activations from a VGG-19 model [30]. The 16 frame-wise

tensors are averaged using global average pooling to yield a $7 \times 7 \times 512 - D$, $512 = D_v$, visual descriptor per second. Audio is encoded at 1 s granularity with a VGG-style network pretrained on AudioSet, producing $128 - D = D_a$ features [13] for every second. For text, we use a short descriptive sentence for event labels using handwritten templates from [39] and encode tokens with BERT to obtain $768 - D$ word features [7]. Then we use a bidirectional LSTM to temporally contextualize the tokens, yielding $256 - D = D_t$ text context vectors.

Modal-Specific Encoders:

Video Encoder $\Psi^v(\cdot)$: Following the architecture from [40], we employ a two-stage video encoding process. First, a channel-spatial attention mechanism processes the input video features $\mathbf{x}^v \in \mathbb{R}^{B\times T\times H\times W\times D_v}$ to produce spatially-aggregated features $\mathbf{g}^v_{att} \in \mathbb{R}^{B\times T\times D_v}$ while simultaneously preserving spatial information through convolutional transformations, yielding $\mathbf{g}^v_{spatial} \in \mathbb{R}^{B\times T\times H'\times W'\times D'_v}$. The aggregated features $\mathbf{g}^v_{att}$ are first projected to the model dimension $D = 256$ through linear transformations, then processed through an Internal Temporal Relation Module consisting of a 2-layer transformer encoder with 4-head self-attention to capture temporal dependencies, producing the final video representation $\mathbf{z}^v \in \mathbb{R}^{B\times T\times D}$.

Audio Encoder $\Psi^a(\cdot)$ and Text Encoder $\Psi^t(\cdot)$: Both audio and text modalities employ the Internal Temporal Relation Module directly. The audio features $\mathbf{x}^a \in \mathbb{R}^{B\times T\times D_a}$ and text features $\mathbf{x}^t \in \mathbb{R}^{B\times T\times D_t}$ are first projected to the model dimension $D = 256$ through linear transformations, then processed through the same 2-layer transformer architecture seperately to capture temporal relationships, yielding $\mathbf{z}^a \in \mathbb{R}^{B\times T\times D}$ and $\mathbf{z}^t \in \mathbb{R}^{B\times T\times D}$ respectively.

Modality-Specific Projections and Decoders:

Projection Layers: Due to the inherent spatial complexity of visual data, we employ distinct projection strategies for each modality input features $x^m, m \in \{a, v, t\}$. For video, we leverage the spatially preserved features $\mathbf{g}^v_{spatial}$ from the encoder's intermediate stage, which serve as $P_v(\cdot)$ and maintain the spatial structure necessary for accurate reconstruction. For audio and text modalities, we apply simple linear projections $P_a(\cdot) : \mathbb{R}^{D_a} \to \mathbb{R}^D$ and $P_t(\cdot) : \mathbb{R}^{D_t} \to \mathbb{R}^D$ respectively, as these modalities lack the spatial dimensionality that requires preservation.

Video Decoder $\mathcal{D}^v(\cdot)$: The video decoder addresses the spatial reconstruction challenge through a convolutional architecture. The trimodal codes for every batch, $\hat{\mathbf{z}}^{\text{tri}}_i = [\hat{\mathbf{z}}^a_i; \hat{\mathbf{z}}^v_i; \hat{\mathbf{z}}^t_i] \in \mathbb{R}^{B\times T\times 3D}$, are first projected and spatially broadcasted to match the spatial dimensions of $\mathbf{h}^v_{spatial}$. After concatenation along the channel dimension, a series of transposed convolutions with residual connections progressively upsample the features: $\hat{\mathbf{x}}^v = \text{ConvTranspose}([\text{Broadcast}(\hat{\mathbf{z}}^{\text{tri}}); \mathbf{h}^v_{spatial}])$, where the decoder reconstructs the original spatial resolution $\mathbb{R}^{B\times T\times H\times W\times D_v}$.

Audio Decoder $\mathcal{D}^a(\cdot)$ and Text Decoder $\mathcal{D}^t(\cdot)$: Both audio and text decoders employ an identical architecture to reconstruct their respective modality features from the batch-wise trimodal representations $\hat{\mathbf{z}}^{\text{tri}}_i = [\hat{\mathbf{z}}^a_i; \hat{\mathbf{z}}^v_i; \hat{\mathbf{z}}^t_i] \in \mathbb{R}^{B\times T\times 3D}$. For audio reconstruction, the trimodal codes are first projected through two successive linear transformations: $\hat{\mathbf{z}}^a_{proj} = \text{Linear}(\text{Linear}(\hat{\mathbf{z}}^{\text{tri}}))$, then concatenated with the audio projection features $P_a(\mathbf{x}^a)$ and passed through a final linear layer to reconstruct the original audio: $\hat{\mathbf{x}}^a = \text{Linear}([\hat{\mathbf{z}}^a_{proj}; P_a(\mathbf{x}^a)]) \in \mathbb{R}^{B\times T\times D_a}$. Similarly, the text decoder processes the trimodal codes through $\hat{\mathbf{z}}^t_{proj} = \text{Linear}(\text{Linear}(\hat{\mathbf{z}}^{\text{tri}}))$ and combines them with the text projection features to reconstruct: $\hat{\mathbf{x}}^t = \text{Linear}([\hat{\mathbf{z}}^t_{proj}; P_t(\mathbf{x}^t)]) \in \mathbb{R}^{B\times T\times D_t}$, where $P_t(\cdot)$ and $P_a(\cdot)$ represent the text-specific and audio-specific projection.

Cross-modal prediction CPC: We employ Cross-CPC between modality pairs to inject fine-grained temporal cross-modal signals [35]. The prediction horizon is set to $H{=}2$ for both the audio–visual stage and the audio–visual–text (trimodal) stage, and we use a single-layer unidirectional LSTM as our autoregressive model to contextualize past embedding frames of each modality.

Optimization: We pre-train for 6 epochs with Adam at a learning rate of $4{\times}10^{-4}$, batch size 80, and EMA decay $\rho{=}0.99$ for codebook updates. Inactive entries are reset when they have not been selected for $N_{\text{re}}{=}300$ consecutive mini-batches. All pretraining runs are performed on a single NVIDIA A100 GPU.

9.2. Downstream Tasks

Across all tasks, encoders and codebooks are *frozen*. Only lightweight task heads are trained on the trimodal codebook. All downstream experiments run on a single NVIDIA A100 GPU.

Cross-modal event classification (AVE): AVE contains 28 classes with 10 s audio–video clips [32]. Using the pretrained encoder, a 10 s video produces a sequence of 10 trimodal discrete vectors. A two-layer MLP maps these to a 28-D class space with softmax and cross-entropy. We use a learning rate of $2.5{\times}10^{-4}$ and a batch size of 256. After training on one source modality (e.g., V→A), we swap the input modality (A) at evaluation time and reuse the same head to evaluate zero-shot transfer; A→V is symmetric.

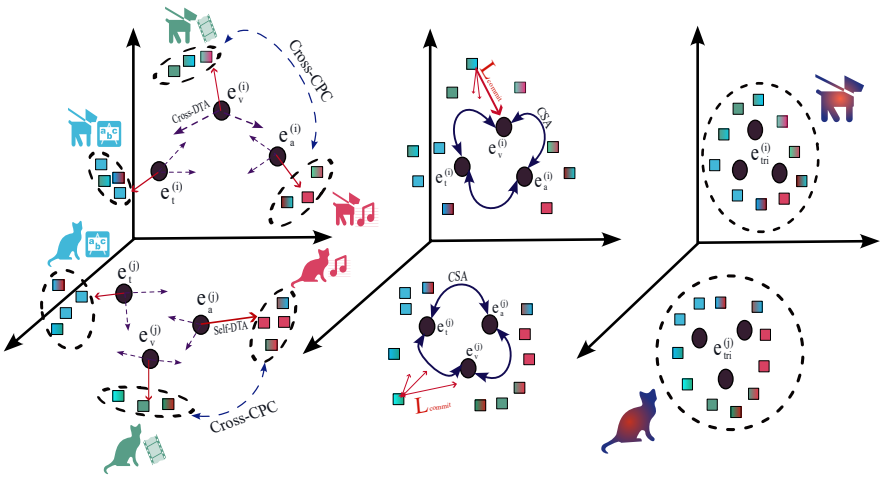

Figure 6. Cross-modal Discrete Alignment Framework

Cross-modal event localization (AVVP): AVVP comprises 25 event types and 10 s clips, but unlike AVE where each clip contains a single event type, AVVP segments can be multi-label [33]. Importantly, the audio and video modalities within the same clip may contain different event labels. For instance, in a single video segment, the audio track might contain events A and B, while the visual information contains events A, C, and D. We reuse the encoder and the trimodal codes and attach a three-layer MLP with sigmoid activations to produce 25 per-segment scores for multi-label classification; a standard multi-label cross-entropy loss is used to handle the presence of multiple simultaneous events. Hyperparameters match AVE.

Cross-modal video segmentation (AVS-S4): AVS-S4 has 4,932 five-second videos over 23 categories [43]. For A→T, we encode audio into the unified discrete space using the AVT settings pretrained encoder. The downstream segmentation task decoder follows the AVS architecture [43] (visual backbone, audio–video interaction, and decoder). We use Adam with a learning rate of 1×10^{-4} and batch size 4. After training, we replace the audio input with text to evaluate T→A generalization (and vice versa for A→T).

Cross-modal zero-shot retrieval (MSCOCO [21], Clotho [8]): MSCOCO contains 5,000 validation images with 5 captions each; Clotho provides 1,045 evaluation audio clips with 5 captions each. Visual features are extracted from VGG-19 pool5 activations and audio features from a VGGish model, using the same pipelines as pretraining. Captions are encoded with BERT and contextualized via a bidirectional LSTM. Each modality input is quantized through the frozen pretrained encoder and codebook, and the resulting discrete representations are temporally averaged to obtain a single embedding per sample. Retrieval is performed via nearest-neighbor lookup in the quantized embedding space without any task-specific fine-tuning.

Cross-modal & cross-dataset localization (AVE→AVVP): To measure generalization across both modality and dataset, we exploit the 12 shared classes between AVE and AVVP (e.g., *dog*, *car*, *helicopter*, *violin*, *frying*, *motorcycle*, *acoustic guitar*, *banjo*, *baby cry*, *chainsaw*, *cat*, *accordion*) [32, 33]. We train a weakly supervised localization head, a two-layer MLP, on AVE (global events) using either video or audio and evaluate localization on the opposite modality in AVVP (local events) using the F1 score. Other settings mirror the AVVP localization protocol above.

Cross-modal cross-dataset zero-shot transfer (UCF ↔ VGGSound): To evaluate cross-modal-dataset generalization, we conduct bidirectional zero-shot transfer between UCF-101 [31] (video) and VGGSound [44] (audio) datasets, which share 16 common categories, including musical instruments (*acoustic guitar, cello, flute, piano, sitar, tabla, violin*), sports activities (*skiing, table tennis, frisbee, basketball, bowling, volleyball, skateboarding*), and other actions (*typing on typewriter, rope skipping*). We train the model on one modality and evaluate on the other: UCF(v) → VGGSound(a) trains on video features and tests on audio, while VGGSound(a) → UCF(v) performs the reverse. We freeze the pretrained encoders and codebooks and attach a linear 2-layer MLP classifier to the quantized repre-

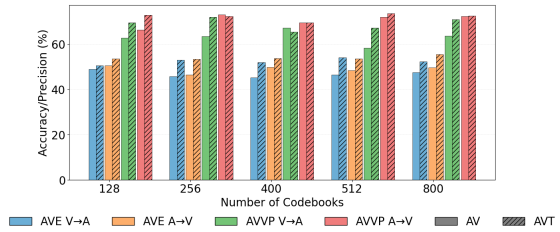

Figure 7. Ablation on different codebook sizes in AV and AVT settings on two downstream tasks

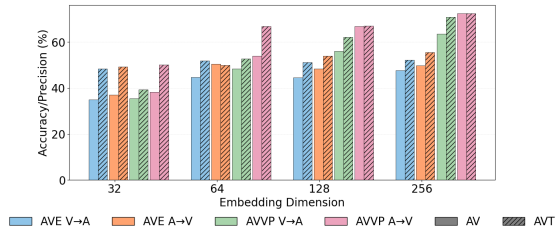

Figure 8. Ablation on different dimension sizes in AV and AVT settings on two downstream tasks

sentations for 16-way classification.

10. Experiments Continued

10.1. Cross-dataset domain transfer evaluation setup:

To assess generalization across datasets and modalities, we further conduct two additional evaluations (Table 8): (i) train on AVE (global classification) on one modality and test zero-shot on AVVP (fine-grained localization) on the other modality, reporting segment-level F1; (ii) train a visual classifier on the 16-class subset of UCF101 [31] and evaluate zero-shot audio classification on the corresponding 16-class subset of VGGSound-AVEL [44], and vice versa, reporting precision.

10.2. Cross-dataset Domain Transfer Comparison with SOTA:

Cross-dataset Domain Transfer: Table 8 evaluates zero-shot transfer across datasets and modalities. On both AVE↔AVVP and UCF(v)↔VGG(a), CoDAAR achieves competitive or higher segment-level F1 and precision than DCID and MICU, reflecting strong cross-modal index reuse and improved representation sharing between visual and auditory domains.

11. Ablations Continued

11.1. Codebook Size Ablation

Figure 7 shows the effect of varying the number of codewords per modality. Performance improves steadily from

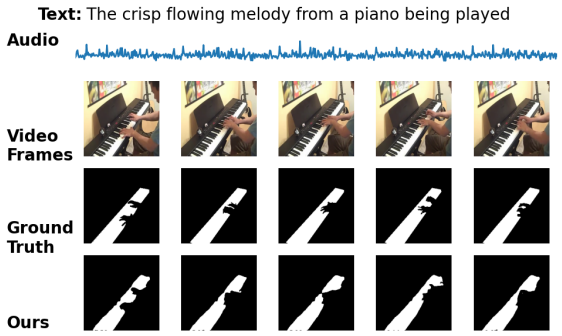

Figure 9. Visualization of text-to-audio generalization on AVS-S4 video segmentation task - Piano playing

128 to 800 entries in both settings, with the largest gains observed for AVVP. Larger codebooks allow finer multimodal quantization and better cross-modal index matching. We therefore adopt 800 entries per modality as a balanced trade-off between accuracy and efficiency.

11.2. Embedding dimension Ablation

Figure 8 shows a consistent trend: bigger dimensions enhance AVE and AVVP in all directions, with significant gains up to **256**. With more embedding dimensions, fine-grained details are captured by each encoder Ψ^m. Hence, we set the embedding dimension to 256 in all experiments.

12. Computational Efficiency Comparison

Table 9 compares computational efficiency and performance accuracy under the same audio-video-text pre-training setup on a single NVIDIA A100 (40 GB) GPU. DCID [39] combines an additional CLUB-based information minimization objective with *cross-attention-guided* EMA updates of a unified codebook, while MICU [16] likewise uses cross-attention-guided EMA and adds further auxiliary constraints (e.g., a jigsaw loss). Consequently, both architectures increase per-epoch training time. In contrast, **CoDAAR** avoids heavy cross-attentional updates and extra objectives, using modality-specific codebooks with a mathematical alignment of DTA+CSA. Our modality-specific codebook design raises peak memory (three codebooks vs. one) but improves throughput (fewer minutes/epoch). Despite the higher memory footprint, CoDAAR attains the best downstream average on AVE classification and AVVP localization (61.7 vs. 58.2 for DCID and 51.6 for MICU), as shown in Table 9 for the AVT setting.

13. More AVS Generalization Visualization:

We visualize cross-modal transfer on AVSBench-S4 [43] using our frozen encoders and trimodal codes, training a downstream query-based segmentation head that follows

Method	VGGSound-AVEL 40K				VGGSound-AVEL 90K				Average
	AVE→AVVP		UCF(v)↔VGG(a)		AVE→AVVP		UCF(v)↔VGG(a)		
	V→A	A→V	V→A	A→V	V→A	A→V	V→A	A→V	
Baseline	1.5	4.3	17.1	12.3	2.5	3.3	19.2	14.4	9.3
DCID [39]	53.0	52.4	67.1	60.6	**57.8**	57.5	69.9	66.5	60.6
MICU [16]	**56.3**	54.9	**75.3**	64.5	54.1	51.6	**73.7**	**67.0**	62.2
CoDAAR (ours)	55.8	**56.5**	73.5	**66.5**	52.3	**60.2**	71.2	65.0	**62.6**

Table 8. AVT setting: comparison with state-of-the-art methods on (i) cross-dataset domain transfer AVE → AVVP (segment-level F1 score) and cross-dataset cross-modal classification UCF(v)↔VGG(a) (Precision).

Method	VGGSound 40K		VGGSound 90K		Total Ep.	Avg.
	GPU mem	Time/ep	GPU mem	Time/ep		
DCID [39]	7.2	22.3	7.4	41.8	5	58.2
MICU [16]	8.4	25.6	8.5	54.5	5	51.6
CoDAAR (ours)	11.6	14.4	13.2	23.8	6	**61.7**

Table 9. Compute profile and downstream average (*AVE+AVVP*) for each pretraining method in the AVT setting. GPU mem is peak memory (GB); Time/ep is minutes per epoch. Total Ep. is the total number of pretraining epochs. Measured on an NVIDIA A100 (40 GB) GPU.

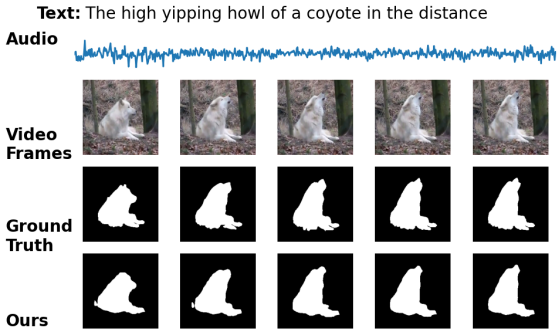

Figure 10. Visualization of text-to-audio generalization on AVS-S4 video segmentation task – Coyote Howling

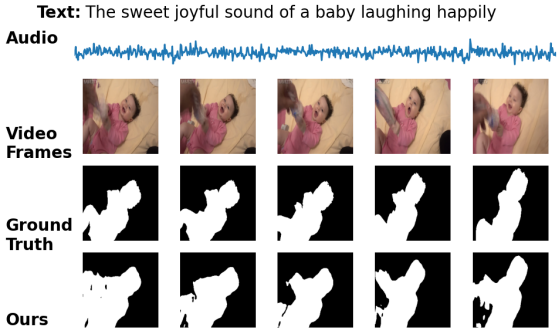

Figure 11. Visualization of audio-to-text generalization on AVS-S4 video segmentation task – Baby Laughing

the AVS architecture [43]. Trained with *text* queries and evaluated with *audio* inputs (T → A), the model accurately segments the visual sources corresponding to the sound, as seen in Fig. 9 (piano playing) and Fig. 10 (coyote howling).

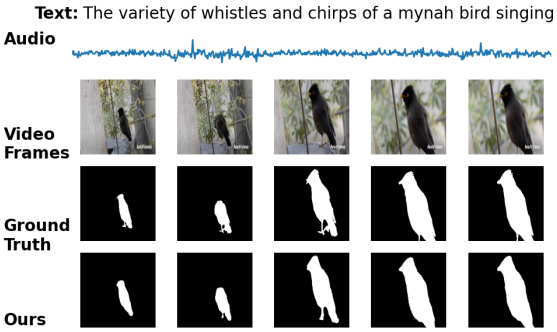

Figure 12. Visualization of audio-to-text generalization on AVS-S4 video segmentation task – Mynah Bird Signing

Conversely, trained with *audio* queries and evaluated with *text* (A → T), the model produces clean masks aligned with the queried semantics, e.g., Fig. 11 (baby laughing) and Fig. 12 (mynah bird singing). This highlights robust cross-modal generalization for video segmentation. The performance advantage is attributed to our tri-modal codes, which preserve richer visual semantics through our Cascading Semantic Alignment module.

14. CSA Closed-Form Weights

Summary: CSA aligns modality-specific centroids at index k by forming a trimodal anchor $\mathbf{c}^0(k)$ ((6)) and applying a sequential T→A→V update ((7)). Fixed non-negative coefficients keep updates inside the convex hull of $\{\mathbf{e}_t^0, \mathbf{e}_a^0, \mathbf{e}_v^0\}$, creating progressive centroids, stabilizing learning and preventing codebook collapse.

Closed-form coefficients: Unrolling (7) yields

$$\begin{aligned}\mathbf{e}_t^1(k) &= \tfrac{1}{3}\mathbf{e}_t^0(k) + \tfrac{1}{3}\mathbf{e}_a^0(k) + \tfrac{1}{3}\mathbf{e}_v^0(k),\\ \mathbf{e}_a^1(k) &= \tfrac{1}{9}\mathbf{e}_t^0(k) + \tfrac{4}{9}\mathbf{e}_a^0(k) + \tfrac{4}{9}\mathbf{e}_v^0(k),\\ \mathbf{e}_v^1(k) &= \tfrac{4}{27}\mathbf{e}_t^0(k) + \tfrac{7}{27}\mathbf{e}_a^0(k) + \tfrac{16}{27}\mathbf{e}_v^0(k).\end{aligned} \tag{12}$$

The video centroid receives the largest self-weight ($16/27$), so it retains the most modality-specific detail while still moving toward the multimodal consensus.

Effect: Together, temporally aligned EMA aggregation (DTA), the CSA cascade above, and the commitment loss (Eq. (8)) yield *distinct multimodal semantic spheres*, as visualized in Fig. 6 and Fig. 3.