CACHESEP: AMORTIZING TEXT-QUERIED SOUND EXTRACTION ACROSS QUERIES IN CODEC LATENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Text-queried target sound extraction models, including recent ones that operate in the compact latent space of a neural audio codec, rerun a query-conditioned separator for every query, even though a recording is often queried repeatedly for different sources. Each request thus repeats work on a mixture that has not changed. We present CacheSep, a codec-domain extractor that runs a query-independent Transformer trunk over the mixture's DAC codec latent once and caches it, then answers each query with a shorter tail conditioned on a frozen CLAP text embedding. In the codec domain, decoding the latent back to audio dominates each query's cost, so we also train a lightweight decoder for the DAC latent, 19.37× faster than DAC's original decoder at comparable reconstruction quality, and share it across all codec-latent systems. We train and evaluate on 1,227 hours of speech, music, environmental, machine, and animal recordings from 25 public datasets, mapped to 89 sound classes. On 8,900 held-out mixtures, every cache split we evaluate stays within 0.11 dB SI-SDRi of the matched codec-domain baseline CodecSep at comparable training cost, while later queries run up to 2.10× faster end to end. Sparse experts in the tail add a small quality gain at a latency cost.

1 INTRODUCTION

Text-queried target sound extraction lets a user name the source to isolate, from speech and music to environmental, machine, and animal sounds, instead of choosing from a fixed set of stems (Delcroix et al., 2023; Liu et al., 2022; 2025; Ma et al., 2024). The open-domain systems that make this possible are costly to run: Banerjee & Arora (2026) find AudioSep (Liu et al., 2025) too compute-intensive for low-latency edge or codec-mediated deployment. Separating in the latent space of a neural audio codec is an efficient alternative. A codec such as DAC compresses 16 kHz audio into a latent at 50 frames per second (Kumar et al., 2023), a sequence 320 times shorter than the waveform, and codec-based separators are efficient but have targeted speech or a fixed set of stems (Yip et al., 2024; Bie et al., 2025). CodecSep brings text queries to this setting by masking frozen DAC latents with a CLAP-conditioned Transformer, and reports about 25× less separator compute than AudioSep with higher SI-SDR on its benchmarks (Banerjee & Arora, 2026).

These systems still answer one query about one mixture at a time. In practice a recording is often queried repeatedly: a listener isolates the speech and then the music of the same scene, an editor extracts several stems from one clip, and an indexing system asks one clip about many classes. Recomputing the separator for each query repeats work on a mixture that has not changed. CodecSep cannot avoid this, because CLAP-derived FiLM parameters (Perez et al., 2018) condition nearly every layer of its masker, so almost all of its computation depends on the prompt (Banerjee & Arora, 2026). Multi-prompt systems such as TUSS share computation across prompts, but only across a prompt set that is known in advance and processed jointly, so their shared representation changes when a new prompt arrives (Saijo et al., 2025; Paissan et al., 2025; Aihara et al., 2026). Amortizing queries that arrive over time requires a stronger property: reusable state that depends only on the mixture.

Codec latents raise a second obstacle. Cheaper separation moves the cost to the codec decoder, which must synthesize a waveform for every query: DAC's original decoder accounts for 98.3% of the operations and 77.0% of the wall-clock time of a later query in our measurements (Section 3.2).

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

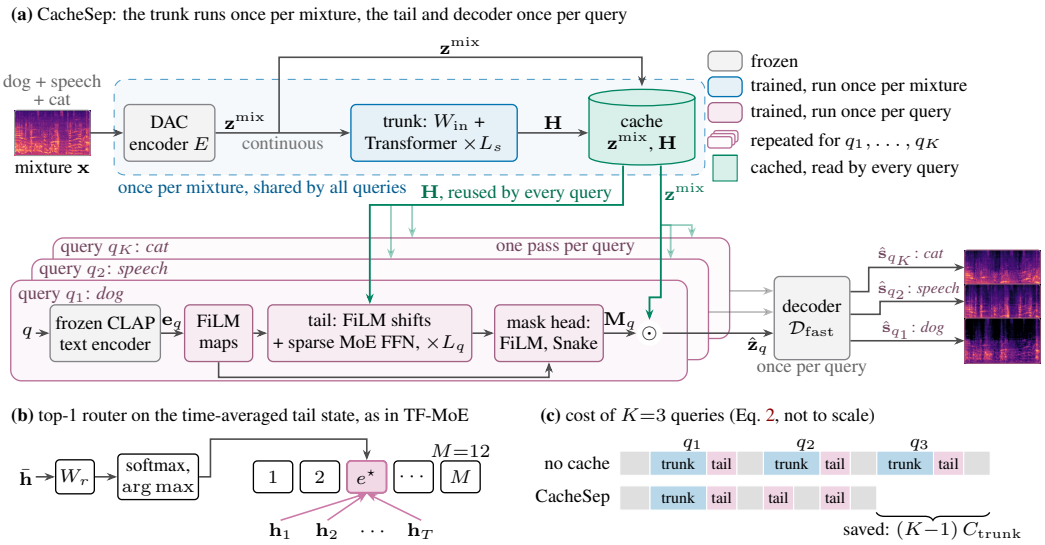


Figure 1: Overview of CacheSep. (a) The encoder and trunk run once per mixture and are cached; each query runs only the CLAP-conditioned tail and the lightweight decoder $\mathcal{D}_{\text{fast}}$. (b) One expert per sparse layer, routed on the time-averaged input. (c) Caching saves $(K-1)\,C_{\text{trunk}}$.

Caching part of the separator therefore cannot speed up extraction end to end unless decoding also becomes cheap (Section 3.1).

We introduce CacheSep, a text-queried extractor designed for this repeated-query setting (Figure 1). A Transformer trunk that receives no query information runs once per mixture and its output is cached; each query then runs only a shorter tail conditioned on a frozen CLAP text embedding. The tail's feed-forward layers can be dense or top-1 sparse experts as in TF-MoE, which add stored capacity while each query executes one expert per layer (Hu et al., 2026); we evaluate both. To remove the decoding bottleneck, we train a lightweight WaveNeXt decoder for the continuous DAC latent (Okamoto et al., 2023a) on diverse sound and decode every codec-latent system through it. We keep CodecSep's latent, mask, CLAP conditioning, and SI-SDR objective, and retrain CodecSep with our data, budget, and decoder, so the comparison isolates where the query enters. Released AudioSep is an external waveform-domain reference.

Our contributions are:

- **Cached, query-independent separation for repeated queries.** We formulate text-queried extraction when one mixture receives many queries over time, and propose CacheSep, whose trunk is query-independent by construction, so its state is computed once and reused, bit for bit, by any later query in any order. Against a matched non-cached baseline, every cache split we evaluate stays within 0.11 dB SI-SDRi at comparable training cost, while later queries run up to 2.10× faster (Sections 3 and 5.2).

- **A fast decoder for DAC latents, trained on diverse sound.** We show that the codec decoder bounds any separator-side saving and remove this bound with a WaveNeXt decoder for the continuous DAC latent, trained on speech, music, environmental, machine, and animal audio and their mixtures. It is 19.37× faster than DAC's original decoder at comparable reconstruction quality and can decode for any separator on this latent, including CodecSep (Sections 3.2 and 5.1).

- **Controlled ablations of what can be cached.** Against CodecSep retrained under identical conditions, we trace where the query can enter, how deep the cached trunk can grow at a fixed per-query cost, and what sparse experts add over a dense tail. Caching half the depth costs no measurable quality, a later entry point buys speed at a small quality cost, and the experts trade latency for a small gain (Sections 5.3 and 5.4).

2 Related Work

Query-conditioned target sound extraction. SoundBeam established class-label and enrollment conditioning for general target sound extraction (Delcroix et al., 2023); LASS-Net and AudioSep extended the task to language queries (Liu et al., 2022; 2025), and CLAPSep uses contrastive audio-language representations for multimodal conditioning (Ma et al., 2024). Recent systems scale to larger sound inventories or richer prompts, including weakly labelled universal separation and generative prompt-conditioned extraction (Kong et al., 2026; Shi et al., 2025) (Appendix C.4). These works primarily optimize the quality of one queried output at a time. Waveformer is structurally closer to our factorization because label conditioning appears after a query-independent encoder, but it is designed for one streaming query and does not reuse the encoder state across later queries (Veluri et al., 2023).

Sharing computation across sources. TUSS jointly models a mixture with a set of requested source prompts, FasTUSS reduces the cost of this formulation, and SUNAC brings the same multi-source idea into a neural audio codec (Saijo et al., 2025; Paissan et al., 2025; Aihara et al., 2026). Their shared computation depends on a prompt set known together; FasTUSS also analyzes prompt-independent mixture features, but as a conditioning ablation rather than a state reused across queries. CacheSep makes this invariance its design: the cached representation depends only on the mixture, so it serves an arbitrary later sequence of queries (Appendix C.5).

Separation in neural codec latents. CodecFormer performs speech separation in neural audio codec embeddings at lower compute than waveform-domain separators (Yip et al., 2024), while SD-Codec modifies the codec itself to organize latents by source type (Bie et al., 2025). CodecSep is the closest antecedent to our setting: it freezes DAC and CLAP, applies CLAP-derived FiLM in all but the first and last blocks of a Transformer masker, estimates a multiplicative latent mask, decodes the masked latent, and trains with waveform SI-SDR (Banerjee & Arora, 2026). Its published analysis finds masking preferable to decoder-style latent generation, and we retain its mask.

Sparse experts in separation. Sparse mixture-of-experts layers increase parameter capacity while activating only a subset of experts per input (Lepikhin et al., 2021; Fedus et al., 2022). TF-MoE applies sparse feed-forward experts to speech separation, routing each sequence as a whole to one top-1 expert chosen from its mean-pooled features (Hu et al., 2026). CacheSep adopts TF-MoE's expert layer unchanged and places it only in the query-conditioned tail, so the pooled features it routes on already carry the query.

Lightweight codec and vocoder decoders. DAC reconstructs waveforms with a convolutional decoder containing learned upsampling stages (Kumar et al., 2023). Vocos instead predicts Fourier coefficients with a ConvNeXt backbone and an inverse STFT, avoiding a deep network at audio rate (Siuzdak, 2024), as do later tokenizers such as WavTokenizer (Ji et al., 2025). WaveNeXt replaces that inverse STFT with a per-frame linear layer emitting waveform samples directly (Okamoto et al., 2023a); our decoder keeps WaveNeXt's published architecture and loss.

3 Method

3.1 Repeated-query formulation and cost

Let a mixture be $\mathbf{x} = \sum_{j=1}^{J} \mathbf{s}_j$ and let a text query q identify one target source $\mathbf{s}_q$. Standard target sound extraction evaluates a query-conditioned network independently for each $(\mathbf{x}, q)$ pair. We instead factor the separator as

$$f(\mathbf{x}, q) = g\big(h(\mathbf{x}), q\big), \tag{1}$$

where h is prohibited from receiving query information. Its output can therefore be cached before any query is known and reused unchanged for every later query on the same mixture. For K queries, the codec-latent path costs

$$C(K) = \underbrace{C_{\text{enc}} + C_{\text{trunk}}}_{\text{once per mixture}} + K \underbrace{(C_{\text{query}} + C_{\text{tail}} + C_{\text{dec}})}_{\text{once per query}}, \tag{2}$$

where C_{query} is zero online when the finite set of canonical CLAP prompt embeddings is precomputed. Dividing Eq. 2 by K and letting K grow, the per-query speedup of caching over recomputing the trunk is

$$S_\infty = \frac{C_{\text{trunk}} + C_{\text{tail}} + C_{\text{dec}}}{C_{\text{tail}} + C_{\text{dec}}} = 1 + \frac{C_{\text{trunk}}}{C_{\text{tail}} + C_{\text{dec}}} < 1 + \frac{C_{\text{trunk}}}{C_{\text{dec}}}, \tag{3}$$

so the decoder caps what any separator-side saving can achieve, however much of the separator is cached. With DAC's original decoder, $C_{\text{dec}} = 278.09$ GFLOPs per query on a 5 s mixture against $C_{\text{trunk}} = 3.79$ GFLOPs for the trunk we adopt (Section 3.3), which limits S_∞ to $1.01\times$ in operations even with infinitely many queries. Reducing C_{tail} or C_{trunk} is therefore useful only once C_{dec} is reduced as well; with the lightweight decoder of Section 3.2, S_∞ becomes $1.37\times$.

3.2 DAC LATENT INTERFACE AND LIGHTWEIGHT DECODER

We use the 16 kHz DAC codec, the codec of published CodecSep, so the matched comparison changes only the separator (Banerjee & Arora, 2026). DAC is a general-audio codec trained on speech, music, and environmental sound (Kumar et al., 2023), which suits an open set of target classes, and its 16 kHz model matches our sample rate. Separation is performed on the continuous latent $\mathbf{z}^{\text{mix}} \in \mathbb{R}^{D\times T}$ of its frozen encoder ($D = 1024$, 50 frames per second), the tensor CodecSep masks; no system in this comparison uses DAC's residual vector quantizer (RVQ). DAC's original waveform decoder is computationally heavy relative to a codec-latent separator: on our hardware it accounts for 77.0% of the wall-clock time and 98.3% of the operations of a later query, so by Eq. 3 caching the trunk cannot pay off end to end unless synthesis also gets cheaper. We therefore train a lightweight decoder $\mathcal{D}_{\text{fast}}$ on the latent the separator masks, using only the training partition of our data (Section 4.1): every class, as isolated sources and as mixtures, for 2,222.2 hours of training audio. $\mathcal{D}_{\text{fast}}$ is the published WaveNeXt vocoder (Okamoto et al., 2023a): a ConvNeXt backbone at codec frame rate (Siuzdak, 2024) followed by a per-frame linear head that emits waveform samples directly, with no inverse STFT and no transposed convolution. We adapt it only to the DAC latent width and our sample rate and keep its published objective, so the result measures the transfer of a published recipe rather than a decoder search of our own. It is then frozen and shared by all codec-latent systems (Appendix B.2).

3.3 QUERY-INDEPENDENT TRUNK

The separator starts with a pointwise projection of $\mathbf{z}^{\text{mix}}$ to model width d and a sinusoidal positional encoding P, followed by a Transformer of L pre-LN blocks (Xiong et al., 2020); with CodecSep's post-LN ordering the output loses its variation over time early in training (Appendix C). We set $L = 16$ to match the depth of the published CodecSep masker (Banerjee & Arora, 2026), then split these blocks into L_s shared blocks and $L_q = L - L_s$ query-conditioned blocks. The shared trunk is

$$\mathbf{H} = \text{Trunk}_{L_s}(W_{\text{in}}\mathbf{z}^{\text{mix}} + P), \tag{4}$$

and receives no text embedding, FiLM parameter, router decision, or query-dependent normalization statistic. Both $\mathbf{H}$ and $\mathbf{z}^{\text{mix}}$ are cached. This restriction is the defining invariant of CacheSep: for a fixed mixture, the cached tensors are bit-identical regardless of which queries arrive or in which order they arrive. The split L_s/L_q is a deployment choice that trades later-query cost against query-conditioned depth. Our reference configuration, fixed before training, is $L_s/L_q = 8/8$, caching half of CodecSep's depth; Section 5.3 traces the trade-off with the 4/12 and 12/4 splits, a 16-block trunk over the same eight-block tail (16/8), and the 8/8 split with dense feed-forward layers. With $L_s = 0$ and dense feed-forward layers, the separator is exactly our retrained CodecSep masker (Appendix B.4), so CacheSep and CodecSep differ only in how many blocks run before the query enters and in whether the tail feed-forward layers are sparse.

3.4 CLAP-CONDITIONED SPARSE TAIL

A frozen CLAP text encoder maps the prompt to $\mathbf{e}_q \in \mathbb{R}^{512}$ (Wu et al., 2023). As in CodecSep, learned linear maps of $\mathbf{e}_q$, zero at initialization, produce feature-wise shifts added after both sublayers of every tail block except the network's first and last, and at two points of the mask head (Section 3.5); no query signal enters the cached trunk. Moving blocks into the trunk reduces the number of query-conditioned dense parameters that can specialize to a target. To add query-specific

capacity without evaluating all of it, each tail feed-forward sublayer is replaced by a sparse mixture of experts that follows TF-MoE, the closest separation MoE (Hu et al., 2026): $M = 12$ experts, the best setting in its expert-count ablation, top-1 routing, and its router. For tail layer ℓ with feed-forward input frames $\mathbf{h}_1^{(\ell)}, \dots, \mathbf{h}_T^{(\ell)}$, the router reads their time average,

$$\bar{\mathbf{h}}^{(\ell)} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{h}_t^{(\ell)}, \qquad \mathbf{p}^{(\ell)} = \mathrm{softmax}\big(W_r^{(\ell)} \bar{\mathbf{h}}^{(\ell)}\big), \qquad e_\ell^\star = \arg\max_e p_e^{(\ell)},$$
$$\mathrm{FFN}^{(\ell)}(\mathbf{h}_t^{(\ell)}) = p_{e_\ell^\star}^{(\ell)}\, E_{e_\ell^\star}^{(\ell)}(\mathbf{h}_t^{(\ell)}), \qquad t = 1, \dots, T, \tag{5}$$

so one expert, weighted by its gate probability, is applied to every latent frame. Because the tail input already carries the FiLM shifts of $\mathbf{e}_q$, the choice depends on both the mixture and the query, while the trunk stays query-independent. Routing is therefore constant across time for each mixture and query, and inference executes one expert per layer rather than all M. As in TF-MoE, each expert has the full size of the feed-forward sublayer it replaces, here CodecSep's (width 1024, ReLU), so the path a query executes has CodecSep's shape and compute up to the router, and the experts add only stored capacity. Our one departure from TF-MoE is the load-balancing coefficient: we add Switch Transformer's balancing loss $\mathcal{L}_{\mathrm{bal}}$ with its coefficient $\lambda_{\mathrm{bal}} = 10^{-2}$ rather than TF-MoE's 10^{-3} (Fedus et al., 2022), because time-averaging makes $\bar{\mathbf{h}}^{(\ell)}$ similar across mixtures and queries, which at the smaller coefficient concentrated routing on few experts (Appendix B.4).

3.5 CodecSep-matched mask and task objective

We use CodecSep's mask head unchanged. The tail output $U_q \in \mathbb{R}^{d \times T}$ is mapped to a mask

$$\mathbf{M}_q = \mathrm{Snake}\big(W_3\,(W_2(W_1 U_q + \boldsymbol{\beta}_{1,q}) + \boldsymbol{\beta}_{2,q}) + \mathbf{b}_3\big), \qquad \hat{\mathbf{z}}_q = \mathbf{M}_q \odot \mathbf{z}^{\mathrm{mix}}, \tag{6}$$

where W_1 projects to width D, W_2 is weight-normalized, $\boldsymbol{\beta}_{1,q}, \boldsymbol{\beta}_{2,q}$ are FiLM shifts computed from $\mathbf{e}_q$, and Snake is the periodic activation of DAC (Kumar et al., 2023). As in CodecSep's released code, the mask is not squashed to $[0, 1]$, so it can rescale or flip the sign of a latent coordinate, which oracle masks show this latent needs (Appendix C). The waveform estimate is $\hat{\mathbf{s}}_q = \mathcal{D}_{\mathrm{fast}}(\hat{\mathbf{z}}_q)$. For the set $\mathcal{Q}(\mathbf{x})$ of source prompts queried on one training mixture, the primary task loss follows CodecSep: negative SI-SDR for every requested source plus a mixture-consistency term that compares the decoded sum of the estimated latents with the sum of the queried sources. Over a batch $\mathcal{B}$ of mixtures, with $N = \sum_{\mathbf{x}\in\mathcal{B}}(|\mathcal{Q}(\mathbf{x})| + 1)$ scored items,

$$\mathcal{L}_{\mathrm{sep}} = -\frac{1}{N}\sum_{\mathbf{x}\in\mathcal{B}}\Big[\sum_{q\in\mathcal{Q}(\mathbf{x})} \mathrm{SI\text{-}SDR}(\mathbf{s}_q, \hat{\mathbf{s}}_q) + \mathrm{SI\text{-}SDR}\Big(\sum_{q\in\mathcal{Q}(\mathbf{x})} \mathbf{s}_q,\ \mathcal{D}_{\mathrm{fast}}\Big(\sum_{q\in\mathcal{Q}(\mathbf{x})} \hat{\mathbf{z}}_q\Big)\Big)\Big]. \tag{7}$$

This is CodecSep's reduction, a single mean over source and mixture items, and its mixture reference, which excludes any non-queried background. CacheSep minimizes $\mathcal{L}_{\mathrm{sep}} + \lambda_{\mathrm{bal}}\mathcal{L}_{\mathrm{bal}}$ and CodecSep minimizes $\mathcal{L}_{\mathrm{sep}}$; the decoder is never fine-tuned.

4 Experimental Setup

4.1 Data, mixtures, and test sets

For training and evaluation we curate a diverse corpus from 25 publicly available datasets of speech, music, environmental, machine, and animal sounds (sources in Table 7, Appendix A.2), and map their recordings to 89 queryable classes. The classes are grouped into 11 categories for reporting (Table 1; per-class hours in Table 5, Appendix A.1). The 89 classes total 1,226.9 hours: 1,082.8 training, 57.9 monitoring, and 86.2 test hours. Speech contributes 52.7% of the audio and 33 classes have less than one hour, so training draws the target class uniformly rather than in proportion to corpus duration. Mixtures are 5 s long at 16 kHz and generated on the fly: a training item samples one source of the drawn class and adds one to three distractors from other eligible classes (constraints in Appendix A.3). Source levels follow the Divide and Remaster loudness recipe (Petermann et al., 2022) with the class-dependent levels and short-event placement of Appendix A.4. Held-out data follow a fixed per-corpus rule: an official test split when one exists, otherwise a validation split, otherwise a deterministic hashed split. Training and held-out recordings are disjoint by recording

or speaker and are checked for cross-corpus leakage (Appendix A.5). The main test set uses each class as the primary target of 100 fixed mixtures, giving 8,900 mixtures, and evaluates every present source query in those mixtures, giving 26,678 present-query pairs, identical for all systems. The decoder's validation set serves only its checkpoint selection and the comparison of Section 5.1.

Table 1: Classes and hours per category over all splits.

Category	Classes	Hours	Largest classes
Human	7	671.6	Human speech, Human singing, Human body actions, ...
Music	15	265.9	Bass, Keyboard, Organ, Guitar, Percussion, Musical ensemble, ...
Machines/tools	23	68.6	Mechanical fan, Pump, Slider, Valve, Home appliances, ...
Animal	17	61.1	Insects, Birds, Dogs, Marmosets, General animal sounds, ...
Background noise	1	54.1	General noise
Vehicles/road	6	36.7	Road vehicles, Vehicle engines, Aircraft, Road ambience, ...
Media	7	22.3	Video-game sound effects, General sound effects, ...
Objects/materials	8	19.8	Doors/drawers, Fire/explosions, Glass/metal sounds, ...
Natural environment	2	15.4	Water, Wind/weather
Alerts/electronics	2	7.5	Alarm/siren/horn, Phone/electronic alert
Miscellaneous	1	4.1	Miscellaneous sounds

4.2 Compared systems

We compare six codec-latent systems (Table 12 in Appendix C): retrained CodecSep ($L_s/L_q = 0/16$, dense tail), the matched baseline without caching; CacheSep (8/8, sparse tail); an 8/8 control with a dense tail, which separates the effect of the split from that of the experts; 4/12 and 12/4 variants that cache less and more of the same 16 blocks; and a 16/8 variant that places eight more cached blocks in front of the 8/8 tail, so its later queries cost the same as those of 8/8. All six share the continuous DAC latent, the frozen CLAP text encoder, CodecSep's mask head and SI-SDR objective, the training mixtures and budget, and the frozen decoder $\mathcal{D}_{\text{fast}}$; they differ only in the split, the trunk depth, and the tail feed-forward layers. We retrain CodecSep from its official implementation on our mixtures, decoding through the same frozen $\mathcal{D}_{\text{fast}}$ as CacheSep, so the training gradient and the decoding cost are identical across codec-latent systems (Banerjee & Arora, 2026). Where its paper and released code disagree we follow the paper, except that the mask stays unbounded as in the code, and all codec-latent systems use pre-LN blocks (Section 3.3); Appendix C lists every deviation. AudioSep is evaluated from its released checkpoint as an external reference, not a matched comparison (Liu et al., 2025). Its architecture and sample rate differ from ours, and it was trained on 14,100 hours of audio, 13.02× the size of our training set, so a quality difference against it does not isolate any design choice. Its training data include WavCaps, from which our AudioSet test clips are drawn, so some of our test sources may have been in its training data.

4.3 Training protocol

The decoder is trained first, selected on its validation SI-SDR, and frozen before any separator run. Each separator is trained once, from the same seed and on the same on-the-fly mixture stream, for 100,000 steps on batches of 16 five-second mixtures (2,222 hours of mixture audio). Published CodecSep trains for 400,000 steps on batches of four two-second crops (Banerjee & Arora, 2026), the same 1.6 million mixtures but 889 hours of audio, since our crops are longer. All runs use CodecSep's optimizer, learning rate, gradient clipping, validation cadence, validation criterion, and checkpoint rule (Appendix B.4).

4.4 Metrics and statistical reporting

We report SI-SDR improvement (SI-SDRi), SI-SDR (Roux et al., 2019) of the estimate minus SI-SDR of the mixture, against two references. The clean source is the reference the task loss optimizes and the only one shared with AudioSep and waveform-domain work, and the text quotes it unless stated otherwise. The codec reference, $\mathcal{D}_{\text{fast}}$'s reconstruction of the clean source, with the decoded mixture as the input term, removes the decoder's own error shared by all codec-latent systems. Because SI-SDR ignores gain errors, we also report SDRi, AudioSep's scale-sensitive metric (Liu

et al., 2025), against both references. The decoder ceiling, $\mathcal{D}_{\text{fast}}$ applied to the clean target's latent and scored against the clean source, bounds every codec-latent system. For targets of the speech class (*human speaking*), we also report wide-band PESQ (Rix et al., 2001) and ESTOI (Jensen & Taal, 2016) against the clean source, with their improvements over the mixture, for every system including AudioSep; as speech metrics, they are not computed for other classes. The main quality table reports the macro average over the 89 classes so that high-resource classes do not dominate. For each matched comparison, we report a paired 95% bootstrap interval over identical held-out targets, resampling distinct target recordings within class rather than treating mixtures that reuse the same target as independent. Because each arm is trained once, this interval covers sampling variation over held-out targets, not over training runs. Because SI-SDR is not invariant to a time shift, every decoder path is scored at its own fixed synthesis delay, with no per-example lag search (Appendix D).

4.5 Compute protocol

We measure the first query and each later query separately because only the first query pays the cached-trunk cost. For one 5 s mixture and $K \in \{1, 2, 4, 8, 16, 32, 64, 89\}$ queries, we report separator operations, decoder operations, wall-clock latency, peak GPU memory, and cache size. Latency is measured with queries arriving one at a time, the setting that caching targets; Appendix E.2 also reports batched timings, where all K prompts are known together and every system may batch them. The codec-latent systems cache the DAC mixture latent and any query-independent separator state, AudioSep has no codec state to cache, and canonical prompts are embedded once by frozen CLAP for every system; Appendix E gives the timing procedure and the cost of an unseen prompt.

5 Results

5.1 Lightweight decoder speed and reconstruction fidelity

Table 2 tests the decoder alone: each decoder reads the latent it was trained on (quantized for DAC's decoder, continuous for $\mathcal{D}_{\text{fast}}$) and reconstructs the same 639 held-out clips, with PESQ and ESTOI on 100 distinct held-out speech recordings. $\mathcal{D}_{\text{fast}}$ uses 51.72× fewer operations and runs 19.37× faster than DAC's original decoder at comparable quality: SDR, median SI-SDR, PESQ, and ESTOI stay close, and mean SI-SDR differs most (−1.73 dB against −0.20 dB in median) because $\mathcal{D}_{\text{fast}}$ has a heavier tail of low-scoring clips (Figure 3 in Appendix B.3). The fidelity it gives up bounds every codec-latent system equally; Table 4 reports that bound.

Table 2: Decoder reconstruction on held-out audio, scored against the clean audio at each decoder's fixed delay, and cost for one 5 s clip at batch 1. Train h: hours of audio seen in training (steps × batch × excerpt length); for DAC, from its released 16 kHz training configuration.

				SI-SDR (dB) ↑					
Decoder	Params (M)	GFLOPs ↓	ms ↓	mean	med.	SDR (dB) ↑	PESQ ↑	ESTOI ↑	Train h
DAC original	52.33	278.09	14.85	5.25	5.76	6.69	3.95	0.908	3,040.0
$\mathcal{D}_{\text{fast}}$ (ours)	10.78	5.38	0.77	3.52	5.56	6.43	3.75	0.904	2,222.2

5.2 Separation quality and per-query cost

Tables 3 and 4 compare the cost and test-set quality of all systems.

Caching preserves separation quality. Across all cached variants, SI-SDRi differs from CodecSep's by at most 0.11 dB against either reference, and every paired 95% interval lies within ±0.16 dB; on speech, every variant changes PESQ by +0.010 to +0.033 and ESTOI by −0.002 to +0.005. Some of these intervals exclude zero, in both directions: the 8/8 reference configuration changes clean-reference SI-SDRi by +0.07 dB ([+0.02, +0.13]) and 16/8 by −0.10 dB ([−0.16, −0.04]). With one training run per system, run-to-run variance is unmeasured (Section 6), so we read differences of this size as matched quality rather than as a ranking.

Table 3: Inference cost for one 5 s mixture with queries arriving one at a time, and training cost. Later-query GFLOPs assume the mixture latent and trunk state are cached; +DAC: the same query decoded by DAC's original decoder (cost only); AudioSep's STFT and resampling are not counted. First latency includes encoding and the trunk. Training: distinct audio, mixture audio processed, and separator GPU hours, AudioSep's audio from its paper.

	Params (M)		Later-query GFLOPs ↓			Latency (ms) ↓		Training		
System (L_s/L_q)	Trained	Active	Sep.	+$\mathcal{D}_{\text{fast}}$	+DAC	First	Later	Data (h)	Seen (h)	GPU-h
AudioSep	39.0	39.0	245.98	–	–	5.8	5.7	14,100	533,333	–
CodecSep (0/16)	20.0	20.0	8.64	14.01	286.73	9.8	6.7	1,083	2,222	7.2
CacheSep (4/12)	88.6	19.2	6.67	12.05	284.77	10.8	7.4	1,083	2,222	7.6
CacheSep (8/8)	64.4	18.2	4.84	10.22	282.93	10.0	5.3	1,083	2,222	7.2
CacheSep (8/8, dense)	18.1	18.1	4.84	10.22	282.93	9.3	3.7	1,083	2,222	6.2
CacheSep (12/4)	40.2	17.1	**3.01**	**8.39**	**281.10**	9.5	**3.2**	1,083	2,222	6.7
CacheSep (16/8)	70.7	24.5	4.84	10.22	282.93	11.7	5.3	1,083	2,222	7.4

Table 4: Test-set separation quality on 8,900 mixtures and 26,678 present-query pairs, macro-averaged over 89 classes; PESQ and ESTOI on the 5,266 speech-target pairs. *Clean*/*Codec*: reference signal. *Speed*: later-query latency relative to CodecSep (Table 3).

	SI-SDRi (dB) ↑		SDRi (dB) ↑				
System (L_s/L_q)	Clean	Codec	Clean	Codec	PESQ ↑	ESTOI ↑	Speed ↑
AudioSep	**2.80**	–	**6.70**	–	**1.74**	**0.623**	1.18×
CodecSep (0/16)	0.78	4.22	4.64	4.85	1.54	0.478	–
CacheSep (4/12)	0.70	4.14	4.61	4.82	1.56	0.476	0.91×
CacheSep (8/8)	0.86	**4.27**	4.70	4.88	1.57	0.482	1.28×
CacheSep (8/8, dense)	0.77	4.21	4.71	**4.88**	1.55	0.483	1.84×
CacheSep (12/4)	0.79	4.15	4.57	4.77	1.57	0.480	**2.10**×
CacheSep (16/8)	0.68	4.11	4.34	4.52	1.56	0.479	1.27×
Mixture	–	–	–	–	1.32	0.551	–
Decoder ceiling	6.95	–	10.64	–	3.76	0.898	–

Later queries become cheaper at comparable training cost. Every variant with an eight-block or shorter tail answers a later query faster than CodecSep: 12/4 by 2.10× with 2.87× fewer separator operations, the dense 8/8 tail by 1.84×, and the sparse 8/8 and 16/8 tails by 1.28× and 1.27×. Only 4/12, which runs twelve sparse blocks per query, is slower (0.91×). Training runs the trunk once per mixture and the tail once per query, as inference does, and every variant trains in 0.87× to 1.06× of CodecSep's GPU hours on the same data and budget (training curves in Appendix B.6). The saving depends on the decoder: with DAC's original decoder, caching would save only 1.3% of an 8/8 later query's operations, because decoding dominates; with $\mathcal{D}_{\text{fast}}$ it saves 27.1%. Against released AudioSep, trained on 13.02× as much data, the 8/8 configuration's clean-reference SI-SDRi differs by −1.94 dB ([−2.17, −1.72]); a later CacheSep query needs 24.07× fewer operations, including decoding, but takes similar time on our GPU (5.3 against 5.7 ms), where one query is bound by launch overhead rather than arithmetic (Appendix E.2).

5.3 Where the query enters: the trade-off curve

The cached variants trace how quality and cost move with the split; since the 8/8 reference was fixed before training, we describe this curve rather than select a configuration on test data. *Caching half the depth costs no measurable quality.* With dense tails, 8/8 changes SI-SDRi relative to CodecSep by −0.01 dB ([−0.06, +0.04]), while each later query runs 1.84× faster and training takes 6.2 against 7.2 GPU hours. *The experts trade latency for a small gain.* At the same split, sparse experts change SI-SDRi by +0.08 dB ([+0.03, +0.14]) and PESQ by +0.021 ([+0.017, +0.024]) at equal separator operations (4.84 GFLOPs) and nearly equal active parameters, but a later query takes 5.3 rather than 3.7 ms, which we attribute to routing and expert gathering in our implementation rather than to arithmetic, and stored parameters grow from 18.1M to 64.4M. *Moving the entry point sets*

the speed-quality trade-off. Entering four blocks later (12/4) makes each later query 2.10× faster than CodecSep, against 1.28× for 8/8, and changes SI-SDRi relative to 8/8 by −0.07 dB ([−0.13, −0.01]); against CodecSep it is unchanged on the clean reference, +0.01 dB ([−0.06, +0.07]), but −0.07 dB ([−0.13, −0.01]) on the codec reference; entering four blocks earlier (4/12) loses both quality, −0.16 dB ([−0.21, −0.10]), and speed. *A deeper cached trunk does not help.* The 16/8 variant places eight more cached blocks in front of the 8/8 tail, so only its first query costs more (11.7 against 10.0 ms), and it changes SI-SDRi by −0.17 dB ([−0.23, −0.12]); at this training budget, where the query enters matters more than how deep the cached trunk is. The split is thus a deployment choice: a later entry point or a dense tail minimizes per-query latency, and sparse experts suit settings where stored parameters are cheap.

5.4 Amortization strengthens as more queries arrive

Because the trunk is paid once, Equation 2 predicts an advantage that grows with the number of queries. With queries arriving one at a time, the sparse 8/8 configuration is faster than CodecSep end to end from $K = 2$ queries, the dense 8/8 variant from $K = 1$, and 12/4 from $K = 1$. At $K = 89$, CodecSep takes 1.27× as long as 8/8, 1.81× as long as dense 8/8, and 2.07× as long as 12/4 (Table 13 in Appendix E.2 gives every curve, batched timings, peak memory, and cache size).

6 Limitations

The evaluation is scoped to repeated queries over a fixed 89-class general-sound taxonomy addressed by canonical class prompts, so it does not establish open-vocabulary extraction or robustness to arbitrary paraphrases, even though CLAP supplies the text embeddings. Mixtures are synthetic sums of isolated recordings under one loudness and placement recipe, without room reverberation, so behaviour on naturally overlapping field recordings is not measured here. Caching only pays when a mixture is queried more than once, sparse experts raise stored parameters, and the cache grows with mixture duration; we do not study very long recordings or cache eviction, and latency is measured on one GPU, not on CPUs or edge devices. We study one codec, the 16 kHz DAC, because the matched comparison requires CodecSep's codec and DAC is a general-audio codec at our sample rate; the cached-trunk design does not depend on it, but other codecs are not evaluated. Our retrained CodecSep is a matched control, not a reproduction of the published system, since it uses our frozen decoder in place of DAC's, and AudioSep is evaluated from its released checkpoint rather than retrained. The expert count, size, and router are inherited from TF-MoE and the balance coefficient from Switch Transformer rather than ablated. Each arm is also trained once rather than repeated across seeds, so our intervals describe variability over held-out targets and leave run-to-run training variance unmeasured. This limits only the quality comparison, since costs depend on the architecture rather than the weights, so we claim quality comparable to CodecSep, not a gain. All codec-latent systems, CodecSep included, improve PESQ but lower ESTOI below the mixture's and stay well below AudioSep's clean-reference SI-SDRi, so our results concern matched rather than absolute quality. Appendix E.1 gives the scope of the controlled design.

7 Conclusion

CacheSep shows that a text-queried codec-domain extractor need not repeat its full computation for every query: its trunk receives no query information, so it runs once per mixture and is cached, and each query runs only a shorter tail at the per-layer cost of CodecSep, decoded by a lightweight decoder 19.37× faster than DAC's original one that can serve any separator on that latent. Against CodecSep retrained with the same latent, conditioning, mask, objective, data, budget, and decoder, every split we evaluate keeps SI-SDRi within 0.11 dB at comparable training cost, later queries run up to 2.10× faster end to end, and the saving grows with every query. Caching half the depth costs no measurable quality, a later entry point makes each query cheaper at a small quality cost, and sparse experts add a small gain at a latency cost (Section 5.3), so where the query enters is a design choice. Future work will train CacheSep on larger corpora and on data-limited domains such as bioacoustics, extend it to open-vocabulary prompts and to real, long recordings (Section 6), and separate quantized codec tokens directly.

Reproducibility statement

Upon acceptance, we will release the training and evaluation code, the exact decoder and separator configurations and seeds, the taxonomy and loudness table, and the trained checkpoints of the lightweight decoder $\mathcal{D}_{\text{fast}}$ and of the separators reported here. The 25 source corpora carry their own licenses, and several do not permit redistribution of their audio, so we will not redistribute audio. Instead we will release, for every recording in every split, the source corpus, the original file name or identifier, the canonical class, and the start and end times of the segment retained by preprocessing (Appendix A.2), together with the scripts that apply these rules to the original corpora, generate the training mixtures, and regenerate the frozen test set. Anyone with access to the original corpora can therefore rebuild the training and evaluation data exactly. Section 4 and Appendix B.4 give the final training configuration, Appendix A.5 defines held-out splits, Appendix C records the CodecSep audit and all baseline deviations, Appendix D specifies scoring, and Appendix E specifies the inference benchmark.

Ethics statement

CacheSep extracts named sources from recordings, and extraction of human speech could support unwanted surveillance if applied to recordings made without consent. We used only publicly available information and datasets: all training and evaluation data come from the 25 public corpora listed in Appendix A.2, used under their stated licenses. Corpora whose license or consent terms are mixed or unstated are reported as such rather than assumed permissive. Our released code, checkpoints, and metadata will not redistribute any existing audio recording; the metadata release described in the reproducibility statement above lets others rebuild the data from the original corpora.

AI use statement

We used large language models (LLMs) to aid and polish the writing of this paper, specifically to check grammar and to improve readability and clarity.

References

Jakob Abeßer, Saichand Gourishetti, András Kátai, Tobias Clauß, Prachi Sharma, and Judith Liebetrau. IDMT-Traffic: An open benchmark dataset for acoustic traffic monitoring research. In *European Signal Processing Conference (EUSIPCO)*, pp. 551–555, 2021.

Ryo Aihara, Yoshiki Masuyama, Francesco Paissan, François G Germain, Gordon Wichern, and Jonathan Le Roux. SUNAC: Source-aware unified neural audio codec. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 14427–14431, 2026.

Adhiraj Banerjee and Vipul Arora. CodecSep: Prompt-driven universal sound separation on neural audio codec latents. *Transactions on Machine Learning Research*, 2026.

Xiaoyu Bie, Xubo Liu, and Gaël Richard. Learning source disentanglement in neural audio codec. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2025.

Jelto Branding, Dieter von Hörsten, Elias Böckmann, Jens Karl Wegener, and Eberhard Hartung. InsectSound1000 an insect sound dataset for deep learning based acoustic insect recognition. *Scientific Data*, 11(1):475, 2024.

cgeorgiaw. Animal sounds collection, 2025. URL https://huggingface.co/datasets/cgeorgiaw/animal-sounds. Dataset card.

Xize Cheng, Siqi Zheng, Zehan Wang, Minghui Fang, Ziang Zhang, Rongjie Huang, Shengpeng Ji, Jialong Zuo, Tao Jin, and Zhou Zhao. OmniSep: Unified omni-modality sound separation with Query-Mixup. In *International Conference on Learning Representations (ICLR)*, 2025.

DCASE 2024 Task 9 organizers. DCASE 2024 task 9 baseline system: Language-queried audio source separation. https://github.com/Audio-AGI/dcase2024_task9_baseline, 2024.

Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *Transactions on Machine Learning Research*, 2023.

Marc Delcroix, Jorge Bennasar Vázquez, Tsubasa Ochiai, Keisuke Kinoshita, Yasunori Ohishi, and Shoko Araki. SoundBeam: Target sound extraction conditioned on sound-class labels and enrollment clues for increased performance and continuous learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:121–136, 2023.

Hao-Wen Dong, Naoya Takahashi, Yuki Mitsufuji, Julian J. McAuley, and Taylor Berg-Kirkpatrick. CLIPSep: Learning text-queried sound separation with noisy unlabeled videos. In *International Conference on Learning Representations (ICLR)*, 2023.

Julie E. Elie and Frédéric E. Theunissen. The vocal repertoire of the domesticated zebra finch: A data-driven approach to decipher the information-bearing acoustic features of communication signals. *Animal Cognition*, 19(2):285–315, 2016.

Julie E. Elie and Frédéric E. Theunissen. Zebra finches identify individuals using vocal signatures unique to each call type. *Nature Communications*, 9(1):4026, 2018.

Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. Neural audio synthesis of musical notes with WaveNet autoencoders. In *International Conference on Machine Learning (ICML)*, pp. 1068–1077, 2017.

William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font, and Xavier Serra. FSD50K: An open dataset of human-labeled sound events. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:829–852, 2022.

Luca Fredianelli, Francesco Artuso, Geremia Pompei, Gaetano Licitra, Gino Iannace, and Andac Akbaba. Environmental noise dataset for sound event classification and detection. *Scientific Data*, 12(1):1712, 2025.

Makoto Fukushima, Alex M. Doyle, Matthew P. Mullarkey, Mortimer Mishkin, and Bruno B. Averbeck. Distributed acoustic cues for caller identity in macaque vocalization. *Royal Society Open Science*, 2(12):150432, 2015.

David Funosas, Elodie Massol, Yves Bas, Svenja Schmidt, David Bennett, Dominik Arend, Alexander Gebhard, Luc Barbaro, Sebastian König, Rafael Carbonell Font, David Sannier, Fernand Deroussen, Jérôme Sueur, Tomi Trilar, Wolfgang Forstmeier, Lucas Roger, Eloïsa Matheu, Piotr Guzik, Julien Barataud, Laurent Pelozuelo, Stéphane Puissant, Sandra Mueller, Björn Schuller, Jose M. Montoya, Andreas Triantafyllopoulos, and Maxime Cauchoix. A finely annotated dataset for the automated acoustic identification of European Orthoptera and Cicadidae. *Scientific Data*, 13(1), 2026.

Carlos Hernandez-Olivan, Marc Delcroix, Tsubasa Ochiai, Naohiro Tawara, and Shoko Araki. Ontology-based target sound extraction. *arXiv preprint arXiv:2609.00752*, 2026.

Shawn Hershey, Daniel P. W. Ellis, Eduardo Fonseca, Aren Jansen, Caroline Liu, R. Channing Moore, and Manoj Plakal. The benefit of temporally-strong labels in audio event classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 366–370, 2021.

Qinzhe Hu, Chenda Li, Wangyou Zhang, Shujie Liu, Yan Lu, and Yanmin Qian. TF-MoE: Time-frequency mixture-of-experts for efficient speech separation. *arXiv preprint arXiv:2606.29575*, 2026.

Lorène Jeantet and Emmanuel Dufourq. Improving deep learning acoustic classifiers with contextual information for wildlife monitoring. *Ecological Informatics*, 77:102256, 2023.

Jesper Jensen and Cees H Taal. An algorithm for predicting the intelligibility of speech masked by modulated noise maskers. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11):2009–2022, 2016.

Shengpeng Ji, Ziyue Jiang, Wen Wang, Yifu Chen, Minghui Fang, Jialong Zuo, Qian Yang, Xize Cheng, Zehan Wang, Ruiqi Li, Ziang Zhang, Xiaoda Yang, Rongjie Huang, Yidi Jiang, Qian Chen, Siqi Zheng, and Zhou Zhao. WavTokenizer: an efficient acoustic discrete codec tokenizer for audio language modeling. In *International Conference on Learning Representations (ICLR)*, 2025.

Stefan Kahl and Ralph Martin. BioDCASE 2026 task 3: Bioacoustics for tiny hardware development set. Zenodo, https://doi.org/10.5281/zenodo.19453065, 2026.

Sofiene Kammoun, Xavier Alameda-Pineda, and Simon Leglaive. Modeling strategies for speech enhancement in the latent space of a neural audio codec. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 17407–17411, 2026.

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Qiuqiang Kong, Ke Chen, Haohe Liu, Xingjian Du, Taylor Berg-Kirkpatrick, Shlomo Dubnov, and Mark D. Plumbley. Universal source separation with weakly labelled data for computational auditory scene analysis. *IEEE Transactions on Audio, Speech and Language Processing*, 34: 206–219, 2026.

Sneha Kudugunta, Yanping Huang, Ankur Bapna, Maxim Krikun, Dmitry Lepikhin, Minh-Thang Luong, and Orhan Firat. Beyond distillation: Task-level mixture-of-experts for efficient inference. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 3577–3599, 2021.

Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. High-fidelity audio compression with improved RVQGAN. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Grégoire Lafay, Emmanouil Benetos, and Mathieu Lagrange. Sound event detection in synthetic audio: Analysis of the DCASE 2016 task results. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 11–15, 2017.

Charly Lamothe, Manon Obliger-Debouche, Paul Best, Régis Trapeau, Sabrina Ravel, Thierry Artières, Ricard Marxer, and Pascal Belin. A large annotated dataset of vocalizations by common marmosets. *Scientific Data*, 12(1):782, 2025.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations (ICLR)*, 2021.

Haoyang Li, Jia Qi Yip, Tianyu Fan, and Eng Siong Chng. Speech enhancement using continuous embeddings of neural audio codec. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2025.

Xubo Liu, Haohe Liu, Qiuqiang Kong, Xinhao Mei, Jinzheng Zhao, Qiushi Huang, Mark D. Plumbley, and Wenwu Wang. Separate what you describe: Language-queried audio source separation. In *Interspeech*, pp. 1801–1805, 2022.

Xubo Liu, Qiuqiang Kong, Yan Zhao, Haohe Liu, Yi Yuan, Yuzhuo Liu, Rui Xia, Yuxuan Wang, Mark D. Plumbley, and Wenwu Wang. Separate anything you describe. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 33:458–471, 2025.

Steven R Livingstone and Frank A Russo. The Ryerson audio-visual database of emotional speech and song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. *PLoS ONE*, 13(5):e0196391, 2018.

Loren Lugosch, Mirco Ravanelli, Patrick Ignoto, Vikrant Singh Tomar, and Yoshua Bengio. Speech model pre-training for end-to-end spoken language understanding. In *Interspeech*, pp. 814–818, 2019.

Hao Ma, Zhiyuan Peng, Xu Li, Mingjie Shao, Xixin Wu, and Ju Liu. CLAPSep: Leveraging contrastive pre-trained model for multi-modal query-conditioned target sound extraction. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:4945–4960, 2024.

Xinhao Mei, Chutong Meng, Haohe Liu, Qiuqiang Kong, Tom Ko, Chengqi Zhao, Mark D. Plumbley, Yuexian Zou, and Wenwu Wang. WavCaps: A ChatGPT-assisted weakly-labelled audio captioning dataset for audio-language multimodal research. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:3339–3354, 2024.

Inma Mohino-Herranz, Joaquín García-Gómez, Miguel Aguilar-Ortega, Manuel Utrilla-Manso, Roberto Gil-Pita, and Manuel Rosa-Zurera. Introducing the ReaLISED dataset for sound event classification. *Electronics*, 11(12):1811, 2022.

Christina A. S. Mumm and Mirjam Knörnschild. The vocal repertoire of adult and neonate giant otters (*Pteronura brasiliensis*). *PLoS ONE*, 9(11):e112562, 2014.

Tsubasa Ochiai, Marc Delcroix, Yuma Koizumi, Hiroaki Ito, Keisuke Kinoshita, and Shoko Araki. Listen to what you want: Neural network-based universal sound selector. In *Interspeech*, pp. 1441–1445, 2020.

Takuma Okamoto, Haruki Yamashita, Yamato Ohtani, Tomoki Toda, and Hisashi Kawai. WaveNeXt: ConvNeXt-based fast neural vocoder without ISTFT layer. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 1–8, 2023a.

Yuki Okamoto, Kanta Shimonishi, Keisuke Imoto, Kota Dohi, Shota Horiguchi, and Yohei Kawaguchi. CAPTDURE: Captioned sound dataset of single sources. In *Interspeech*, pp. 1683–1687, 2023b.

Francesco Paissan, Gordon Wichern, Yoshiki Masuyama, Ryo Aihara, François G. Germain, Kohei Saijo, and Jonathan Le Roux. FasTUSS: Faster task-aware unified source separation. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 1–5, 2025.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, pp. 3942–3951, 2018.

Darius Petermann, Gordon Wichern, Zhong-Qiu Wang, and Jonathan Le Roux. The cocktail fork problem: Three-stem audio separation for real-world soundtracks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 526–530, 2022.

Harsh Purohit, Ryo Tanabe, Takeshi Ichige, Takashi Endo, Yuki Nikaido, Kaori Suefusa, and Yohei Kawaguchi. MIMII dataset: Sound dataset for malfunctioning industrial machine investigation and inspection. In *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, pp. 209–213, 2019.

Nilo Merino Recalde, Andrea Estandía, Loanne Pichot, Antoine Vansse, Ella F Cole, and Ben C Sheldon. A densely sampled and richly annotated acoustic data set from a wild bird population. *Animal Behaviour*, 211:111–122, 2024.

Antony W Rix, John G Beerends, Michael P Hollier, and Andries P Hekstra. Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 749–752, 2001.

Jonathan Le Roux, Scott Wisdom, Hakan Erdogan, and John R. Hershey. SDR - half-baked or well done? In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 626–630, 2019.

Linus Rüttimann, Yuhang Wang, Jörg Rychen, Tomas Tomka, Heiko Hörster, and Richard HR Hahnloser. Multimodal system for recording individual-level behaviors in songbird groups. *PeerJ*, 13: e20203, 2025.

Kohei Saijo, Janek Ebbers, François G. Germain, Gordon Wichern, and Jonathan Le Roux. Task-aware unified source separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2025.

Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. A dataset and taxonomy for urban sound research. In *ACM International Conference on Multimedia*, pp. 1041–1044, 2014.

Bowen Shi, Andros Tjandra, John Hoffman, Helin Wang, Yi-Chiao Wu, Luya Gao, Julius Richter, Matt Le, Apoorv Vyas, Sanyuan Chen, Christoph Feichtenhofer, Piotr Dollár, Wei-Ning Hsu, and Ann Lee. SAM audio: Segment anything in audio. *arXiv preprint arXiv:2512.18099*, 2025.

Hubert Siuzdak. Vocos: Closing the gap between time-domain and Fourier-based neural vocoders for high-quality audio synthesis. In *International Conference on Learning Representations (ICLR)*, 2024.

Ivo Trowitzsch, Jalil Taghia, Youssef Kashef, and Klaus Obermayer. The NIGENS general sound events database. *arXiv preprint arXiv:1902.08314*, 2019.

Nicolas Turpault, Romain Serizel, Ankit Parag Shah, and Justin Salamon. Sound event detection in domestic environments with weakly labeled data and soundscape synthesis. In *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2019.

Bandhav Veluri, Justin Chan, Malek Itani, Tuochao Chen, Takuya Yoshioka, and Shyamnath Gollakota. Real-time target sound extraction. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023.

Helin Wang, Jiarui Hai, Yen-Ju Lu, Karan Thakkar, Mounya Elhilali, and Najim Dehak. SoloAudio: Target sound extraction with language-oriented audio diffusion Transformer. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2025.

Pete Warden. Speech Commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

Scott Wisdom, Efthymios Tzinis, Hakan Erdogan, Ron J. Weiss, Kevin W. Wilson, and John R. Hershey. Unsupervised sound separation using mixture invariant training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the Transformer architecture. In *International Conference on Machine Learning (ICML)*, pp. 10524–10533, 2020.

Sophia Yin and Brenda McCowan. Barking in domestic dogs: Context specificity and individual identification. *Animal Behaviour*, 68(2):343–355, 2004.

Jia Qi Yip, Shengkui Zhao, Dianwen Ng, Eng Siong Chng, and Bin Ma. Towards audio codec-based speech separation. In *Interspeech*, 2024.

Yi Yuan, Xubo Liu, Haohe Liu, Mark D. Plumbley, and Wenwu Wang. FlowSep: Language-queried sound separation with rectified flow matching. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2025.

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, 2022.

Zhiyuan Zhao, Lijian Lin, Ye Zhu, Kai Xie, Yunfei Liu, and Yu Li. LEMAS: Large a 150k-hour large-scale extensible multilingual audio suite with generative speech models. *arXiv preprint arXiv:2601.04233*, 2026.

A Data

A.1 Taxonomy and class statistics

Table 5 lists the 89 queryable classes by category with their hours, and Figure 2 plots them; Table 1 in Section 4.1 summarizes each category. Human speech contributes 646.5 of the 1,226.9 total hours, the median class has 2.54 hours, 33 classes have less than one hour, and 9 classes have less than 0.2 hours.

Table 5: The 89 queryable classes by category, with hours in the train, monitoring, and test manifests combined.

Class	Hours	Class	Hours
Human (7, 671.6 h)			
Human speech	646.49	Human singing	14.09
Human body actions	7.06	General human sounds	2.02
Children playing	1.15	Crowd/human group	0.39
Flatulence	0.36		
Music (15, 265.9 h)			
Bass	53.05	Keyboard	42.20
Organ	31.86	Guitar	25.71
Percussion	24.90	Musical ensemble	20.74
Bowed strings	18.16	Reed instruments	12.91
Brass	11.26	Flute	8.39
Wind/brass instruments	5.36	Synth lead	4.90
Plucked guitar/strings	3.55	Keyboard instruments	2.69
Piano	0.21		
Machines/tools (23, 68.6 h)			
Mechanical fan	12.00	Pump	12.00
Slider	12.00	Valve	12.00
Home appliances	3.23	Mechanical devices	3.06
Power tools	2.94	Mechanisms	2.12
Lawn mower	1.89	Drilling	1.41
Air conditioner	1.11	Jackhammer	1.02
Vacuum cleaner	0.72	Chainsaw	0.50
Dryer	0.49	Mechanical ticking	0.33
Wind turbine	0.32	Saw	0.30
Grinder	0.30	Printer	0.28
Blender	0.26	Appliance fan	0.22
Microwave	0.13		
Animal (17, 61.1 h)			
Insects	24.85	Birds	16.80
Dogs	5.68	Marmosets	4.88
General animal sounds	2.75	Cats	1.86
Great tits	1.40	Primates	0.70
Frogs	0.53	Wild animals	0.52
Horses	0.49	Zebra finches	0.17
Sheep/goats	0.15	Pigs	0.13
Cows	0.12	Rodents	0.05
Animal growling	0.03		
Background noise (1, 54.1 h)			
General noise	54.06		
Vehicles/road (6, 36.7 h)			
Road vehicles	17.29	Vehicle engines	9.51
Aircraft	4.04	Road ambience	3.58
Trains	2.12	Boats	0.14

Class	Hours	Class	Hours
Media (7, 22.3 h)			
Video-game sound effects	8.84	General sound effects	7.46
Television/radio	3.89	Effects unit	0.73
Spray sound	0.64	Sine wave	0.43
Chorus effect	0.35		
Objects/materials (8, 19.8 h)			
Doors/drawers	7.13	Fire/explosions	5.01
Glass/metal sounds	3.83	Impacts/crashes	2.54
Kitchen/dishes/cooking	0.52	Paper/plastic rustling	0.29
Squeaks/creaks	0.28	Screeching/friction	0.18
Natural environment (2, 15.4 h)			
Water	12.53	Wind/weather	2.84
Alerts/electronics (2, 7.5 h)			
Alarm/siren/horn	6.18	Phone/electronic alert	1.29
Miscellaneous (1, 4.1 h)			
Miscellaneous sounds	4.06		

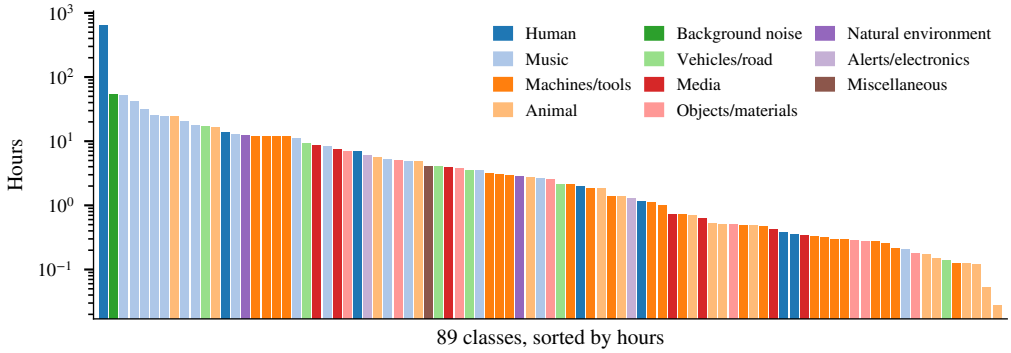


Figure 2: Dataset hours of the 89 queryable classes on a logarithmic scale, colored by category.

A.2 Data sources

The source pools are built from the 25 public datasets of Table 7. Licenses and use terms are reported from the primary dataset record when available; mixed or unstated terms are reported explicitly rather than inferred from an article license.

Each corpus is resampled to 16 kHz and cut to clips of at most 5 s before it enters a source pool; any pool row still longer than 5 s at training time is cropped to 5 s by the dataloader. Two speech corpora (Fluent Speech Commands, Google Speech Commands v0.02) are additionally trimmed by voice-activity detection before pooling. A small number of pools apply a per-dataset duration cap or a minimum-duration filter (for example, a 100 ms floor on the zebra-finch pool) to remove degenerate clips; these thresholds are dataset-specific and are not applied uniformly across all 25 corpora. LEMAS (Zhao et al., 2026), the largest single corpus by hours (Table 8), contributes only to the Human speech class of our taxonomy and is used as a subset of its original release of more than 150,000 hours, capped at a roughly 50-hour training target per language, rather than in full. All 10 languages of the original LEMAS release are kept (German, English, Spanish, French, Indonesian, Italian, Portuguese, Russian, Vietnamese, Mandarin); Table 6 gives the hours contributed per language and split. The monitoring and test splits use the corpus's official evaluation release in full rather than a capped subset, which is why their per-language hours are far smaller than training and roughly even across languages. Independently of pool construction, a class is used as a mixture target only if it has at least 20 training clips (Table 9).

Table 6: LEMAS hours by language and split. All 10 languages of the original release are retained; training is capped at approximately 50 h per language, while monitoring and test use the corpus's official evaluation release in full.

Language	Train (h)	Monitor (h)	Test (h)
German (de)	51.44	0.32	0.28
English (en)	51.65	0.30	0.33
Spanish (es)	51.04	0.32	0.33
French (fr)	51.32	0.30	0.29
Indonesian (id)	52.11	0.35	0.40
Italian (it)	51.44	0.33	0.35
Portuguese (pt)	51.40	0.33	0.34
Russian (ru)	51.15	0.33	0.32
Vietnamese (vi)	52.00	0.29	0.31
Mandarin (zh)	51.06	0.25	0.30
Total	514.62	3.10	3.24

Table 7: Public datasets used to construct the 25 source pools. "Content contributed" summarizes the material retained by our preprocessing, not the full scope of the original dataset.

Dataset	Content contributed	License / terms
AudioSet Strong via WavCaps (Hershey et al., 2021; Mei et al., 2024)	Strongly labelled general sound events	WavCaps: academic research only; AudioSet labels: CC BY 4.0
BioDCASE 2026 Task 3: Bioacoustics for Tiny Hardware (Kahl & Martin, 2026)	Bird vocalizations	CC BY-SA 4.0
BirdPark zebra finch recordings (Rüttimann et al., 2025)	Zebra finch vocalizations	Not stated in Zenodo data record
Xeno-canto 22-species birdsong (Jeantet & Dufourq, 2023)	Birdsong from 22 species	No blanket license stated; upstream Xeno-canto terms apply
CAPTDURE (Okamoto et al., 2023b)	Single-source environmental sounds	CC BY-NC-SA 4.0
DataSEC (Fredianelli et al., 2025)	Isolated environmental noise events	Source-specific CC terms (mixed)
DCASE 2016 Task 2 (Lafay et al., 2017)	Isolated office sound events	CC BY 3.0
DESED (Turpault et al., 2019)	Domestic foreground events	Per-file upstream terms; DESED distributes source attribution/license files
Divide and Remaster (speech stem) (Petermann et al., 2022)	Read speech (LibriSpeech audiobooks)	CC BY 4.0 (v2 record)
EcoSoundSet (Funosas et al., 2026)	Orthoptera and cicada sounds	CC BY-NC 4.0
Fluent Speech Commands (Lugosch et al., 2019)	Command speech	Fluent Speech Commands Public License; academic research only
FSD50K (Fonseca et al., 2022)	General sound events	CC BY 4.0; each clip keeps its Freesound license (CC0, CC BY, CC BY-NC, or CC Sampling+)
Speech Commands v0.02 (Warden, 2018)	Keyword speech	CC BY 4.0
Animal Sounds Collection (cgeorgiaw, 2025) and its source datasets (Yin & McCowan, 2004; Mumm & Knörnschild, 2014; Fukushima et al., 2015; Elie & Theunissen, 2016; 2018)	Dog, giant otter, macaque, zebra finch, and bird vocalizations	CC BY 4.0
IDMT-Traffic (Abeßer et al., 2021)	Road vehicles and traffic	CC BY-NC-ND 4.0
InsectSound1000 (Branding et al., 2024)	Laboratory insect sounds	Not stated in accessible data record

Dataset	Content contributed	License / terms
LEMAS (Zhao et al., 2026)	Multilingual speech	CC BY 4.0
MarmAudio (Lamothe et al., 2025)	Common marmoset vocalizations	Not stated in Zenodo data record
MIMII (Purohit et al., 2019)	Industrial machine sounds	CC BY-SA 4.0
NIGENS (Trowitzsch et al., 2019)	Isolated general sound events	CC BY-NC-ND 4.0
NSynth (Engel et al., 2017)	Musical instrument notes	CC BY 4.0
RAVDESS (Livingstone & Russo, 2018)	Speech and song	CC BY-NC-SA 4.0
ReaLISED (Mohino-Herranz et al., 2022)	Isolated indoor household events	CC BY 4.0
UrbanSound8K (Salamon et al., 2014)	Urban sound events	CC BY-NC 3.0
Wytham Great Tit Song Dataset (Recalde et al., 2024)	Great tit songs	CC BY 4.0

A.3 Source selection and class mapping

Large general-audio corpora such as AudioSet and WavCaps are weakly labelled: a clip-level label says that a sound occurs somewhere in the clip, and other sounds usually occur with it. Such a clip is a poor separation target, because any unlabelled sound in it becomes part of the reference the model is trained to reproduce. Mixture invariant training avoids isolated references altogether (Wisdom et al., 2020), but its outputs carry no text query, so we train on mixtures of homogeneous sources instead. We therefore build the source pools from recordings that are homogeneous, either by construction or after filtering. Most corpora of Table 7 record one source at a time (isolated sound events, single instrument notes, read or command speech, and focal animal recordings) or annotate each event with its onset and offset. From AudioSet we use only the strongly labelled subset (Hershey et al., 2021) and cut each event at its annotated boundaries. An event of a specific class is discarded when it overlaps an event of a different class by more than 0.1 s or by more than 10% of its duration, and repeated events of the same class within one clip are kept together with the gaps between them. Broad AudioSet labels, such as background noise and general sound effects, describe sound that accompanies other events, so they form classes of their own with their own prompts rather than being merged into a specific class. Extracting a sound queried at any level of an ontology, such as animal as well as dog, is a separate task (Hernandez-Olivan et al., 2026); our 89 classes form a single level.

Each corpus's labels are mapped to the 89 classes in two steps. A per-corpus table first maps every original label, such as an AudioSet ontology label, an FSD50K label, or a species name, to a canonical class. A shared class policy then merges aliases, splits broad classes into finer ones where the original labels allow it (for example, keyboard instruments into keyboard, organ, piano, and synth lead), and assigns each class one text prompt and one category. One prompt per class is needed because the corpora name the same sound at different granularity and in different vocabularies, and every system, AudioSep included, receives the same prompt for a class in training and at test time. A label broader than our classes is never assigned to a specific class: AudioSet's growling may come from a dog, a cat, or a wild animal, so it is a class of its own. The same policy constrains which classes may share a mixture: no class appears twice, a general animal-sound distractor is never mixed with a specific animal target, and a cat target is never mixed with general animal sounds or growling, which may contain a cat. When the same recording occurs in more than one corpus, for example a Freesound clip distributed by two datasets, only its first occurrence is kept, matched by source identifier. We will release the per-corpus mapping tables and filtering rules with the code.

A.4 Loudness targets

The loudness recipe of Section 4.1 follows Divide and Remaster (Petermann et al., 2022): each source is normalized to its class's target level (below) with uniform ± 2 dB jitter and a -0.5 dB peak cap, and the resulting mixture is normalized to -27 LUFS. The single speech class is Human speech (-17 LUFS). The 16 music classes (-24 LUFS) are Bass, Bowed strings, Guitar, Plucked guitar/strings, Human singing, Keyboard instruments, Keyboard, Organ, Piano, Synth lead, Musical ensemble, Percussion, Wind/brass instruments, Brass, Flute, and Reed instruments. Every other

class uses the sound-effects level (-21 LUFS). Singing is assigned to music because the music stems of Divide and Remaster are drawn from the Free Music Archive, which includes vocal (sung) tracks alongside instrumental ones, so the "music" stem is not purely instrumental (Petermann et al., 2022).

A.5 Held-out splits and leakage checks

Several corpora lack an official test split, so held-out data are defined by a fixed rule and checked for leakage. Each corpus contributes its official test split when one exists, its validation split when it has only a validation split, and otherwise a test split selected by $\mathrm{SHA1}(\text{recording or speaker identifier}) \bmod 100 < 10$, following the hashing scheme of Speech Commands (Warden, 2018). One class is an exception: the official NSynth test split has no synth-lead recording, so the same hash moves about 10% of the synth-lead recordings of NSynth training data to test and another 10% to monitoring. Monitoring during training uses only validation data of corpora that also have an official test split, so no held-out recording informs any decision. We verified that no recording or speaker identifier occurs in more than one split of a corpus, and training rows that duplicate a held-out recording in another corpus, detected by Freesound identifiers where available and by file hashes, were removed, as were training rows that shared a recording group with a corpus's own held-out split. After all filtering, every queryable class has at least one test row. Table 8 gives the rows and hours each corpus contributes to the train, monitoring, and test manifests.

Table 8: Rows (mixture-eligible recordings after preprocessing) and hours contributed by each of the 25 source corpora to the train, monitoring, and test manifests. A row here is a source recording admitted into a split's pool, not a generated mixture; on-the-fly mixtures are drawn from these pools per Section 4.1. Totals include a small number of rows from marine classes disabled centrally (Appendix A.1), so the grand total is slightly above the 89-class taxonomy total of Table 1.

	Train		Monitor		Test	
Dataset	rows	hours	rows	hours	rows	hours
AudioSet Strong via WavCaps	88,394	92.35	9,858	10.28	21,934	21.87
BioDCASE 2026 Task 3	2,199	1.83	0	0.00	549	0.46
BirdPark zebra finch	166	0.09	31	0.03	118	0.05
Xeno-canto 22-species birdsong	7,903	5.43	948	0.63	951	0.74
CAPTDURE	1,529	1.66	160	0.17	152	0.17
DataSEC	14,910	18.73	1,857	2.33	1,922	2.43
DCASE 2016 Task 2	552	0.18	61	0.02	1,323	0.35
DESED	1,599	1.37	0	0.00	698	0.46
Divide and Remaster (speech)	35,887	49.84	4,873	6.77	9,955	13.83
EcoSoundSet	14,350	12.05	4,250	3.69	3,163	2.65
Fluent Speech Commands	23,119	10.85	3,116	1.42	3,792	1.76
FSD50K	43,006	59.73	6,311	8.77	16,693	23.18
Speech Commands v0.02	84,084	17.55	9,904	2.08	10,948	2.30
Animal Sounds Collection	11,453	4.00	0	0.00	1,300	0.50
IDMT-Traffic	12,379	6.72	1,552	0.84	1,549	0.84
InsectSound1000	8,388	5.82	1,545	1.07	1,583	1.10
LEMAS	444,549	514.62	2,838	3.10	2,960	3.24
MarmAudio	12,386	3.50	2,353	0.68	2,470	0.70
MIMII	28,800	40.00	2,880	4.00	2,880	4.00
NIGENS	5,443	3.45	0	0.00	707	0.43
NSynth	284,070	223.41	13,123	10.78	4,634	3.87
RAVDESS	1,664	1.00	372	0.23	416	0.26
ReaLISED	1,849	0.75	218	0.09	196	0.07
UrbanSound8K	6,981	6.98	816	0.83	837	0.84
Wytham Great Tit Song Dataset	1,975	1.13	254	0.14	225	0.13
Total	1,137,635	1083.05	67,320	57.94	91,955	86.23

B METHOD AND TRAINING DETAILS

B.1 CODEC LATENT PATH

All matched codec-latent systems use one DAC-16k latent interface, and it is the published CodecSep one. CodecSep performs separation on the continuous pre-RVQ DAC encoder output $Z = \mathrm{Enc}(x)$ and decodes the masked estimate directly, with the RVQ absent from the separation path (Banerjee & Arora, 2026). DAC, like SoundStream and EnCodec, pairs a convolutional encoder and decoder with a residual vector quantizer (Zeghidour et al., 2022; Défossez et al., 2023); codec-domain speech enhancement likewise operates on the continuous pre-quantization embedding (Li et al., 2025), and there predicting continuous latents outperforms predicting discrete tokens (Kammoun et al., 2026). Every system here does the same: the separator masks Z, and the masked estimate is decoded by $\mathcal{D}_{\mathrm{fast}}$ without any quantization step. There is no second quantization pass and no straight-through gradient estimator, because no quantizer is ever applied. $\mathcal{D}_{\mathrm{fast}}$ is trained on this continuous latent directly, so it decodes exactly the tensor every separator here emits. Because the latent interface is CodecSep's own, our CodecSep row differs from the published model in its training data, budget, and decoder and in the recipe choices of Appendix C, which is why it is described as retrained rather than reproduced.

For the 16 kHz DAC checkpoint, a 5 s 16 kHz clip produces 250 latent frames with channel dimension 1024. The cache stores this mixture latent plus the output of the L_s-block shared Transformer trunk. The cache size reported in Appendix E.2 includes both tensors in the inference precision used by the benchmark.

B.2 LIGHTWEIGHT DECODER ARCHITECTURE AND TRAINING

$\mathcal{D}_{\mathrm{fast}}$ is a literal port of the WaveNeXt vocoder (Okamoto et al., 2023a) to the continuous DAC-16k latent. The generator is the Vocos ConvNeXt backbone (Siuzdak, 2024) at codec frame rate, followed by the WaveNeXt head, which replaces Vocos's iSTFT synthesis with two linear layers: a per-frame projection to FFT-size-plus-two channels and a bias-free projection to one hop of waveform samples, flattened into the output. The head acts on each frame independently, so it emits exactly one hop (320 samples) per latent frame with no synthesis delay. We use the released EnCodec configuration of a community reimplementation, citing Okamoto et al. (2023a) for the method: backbone width 384, intermediate width 1152, eight ConvNeXt blocks, FFT size 1280, and hop size 320. The frame geometry transfers unchanged, because EnCodec (Défossez et al., 2023) at 24 kHz and 75 Hz and DAC at 16 kHz and 50 Hz both give 320 samples per frame. Two values are adapted to our codec: the input width becomes 1024, the DAC latent width, and the mel loss uses a 16 kHz sample rate with an 8 kHz upper frequency, our Nyquist limit. Bandwidth conditioning is disabled in the generator and both discriminators, since our DAC bitrate is fixed.

The objective is the published one: hinge adversarial and feature-matching losses against the multi-period discriminator of HiFi-GAN (Kong et al., 2020) (periods $\{2, 3, 5, 7, 11\}$) and a multi-resolution discriminator (FFT sizes $\{2048, 1024, 512\}$), plus a mel-spectrogram L_1 term weighted by 45. Optimization uses AdamW with $\beta = (0.8, 0.9)$ and a peak learning rate of 5×10^{-4} with cosine decay to zero, in 32-bit floating point, as published.

Training audio comes only from the separation training partition: all 89 classes, covering speech, music, environmental, and animal sound. Each training item supplies a 1 s crop of an isolated source and a 1 s crop of a mixture built from the same training sources, so the decoder learns both single sounds and the overlapped sounds a separator's masked latent resembles. One step sees 16 such waveforms, the published effective batch. Crops are placed on annotated events, and 90% of digitally silent crops are redrawn. Distractor classes in the training mixtures are sampled in proportion to the square root of their recording count, so that speech, our largest class, does not dominate the mixtures; targets stay class-uniform. The run is 500,000 steps, 8,000,000 s or 2,222.2 hours of waveform audio, with the cosine horizon set to that length (the published recipe runs 10^6 steps); it takes 14.0 hours on one RTX PRO 6000 Blackwell GPU. This budget, half the published step count, was fixed in advance and not tuned, so Table 2 measures the recipe at this budget rather than its limit. The decoder reads the continuous pre-RVQ latent, the same tensor every separator here masks (Appendix B.1), so it is drop-in for all of them. It can only reproduce what the frozen encoder preserved, which is why Section 5.1 compares it with DAC's own reconstruction.

The checkpoint is selected on validation SI-SDR over a validation set fixed before any separator run, then frozen. For this selection only, SI-SDR is maximized over lags within ± 8 samples (0.5 ms); the reported comparison scores each decoder at its fixed delay, 8 samples for DAC's decoder and none for $\mathcal{D}_{\text{fast}}$.

B.3 Qualitative decoder reconstructions

Figure 3 shows 5 s held-out validation clips, the split of Table 2, as heard by DAC's original decoder from its quantized latent and by $\mathcal{D}_{\text{fast}}$ from the continuous latent. The clips were fixed by a rule set before rendering: in the validation stream's fixed order, the first source with at least 4 s of annotated activity for each row (speech, music, bird, dog, cat, another mammal, environmental sound), and the first mixture. Both decoders keep the harmonic and temporal structure of every clip; the per-clip SI-SDR in each panel varies as widely across clips as between decoders, which is why Table 2 reports means and medians over 639 clips.

B.4 Separator architecture and hyperparameters

Table 9 contains the final configuration shared by the matched separator runs. The total Transformer depth, model width, attention configuration, mask parameterization, CLAP encoder, task loss, optimizer family, and training budget are held fixed across CodecSep and every CacheSep split wherever the architecture permits.

Table 9: Separator hyperparameters shared by CodecSep and CacheSep.

Setting	Value
Sample rate / mixture duration	16 kHz / 5 s
Minimum target training clips	20
DAC latent dimension / frame rate	1024 / 50 Hz
Frozen text encoder	CLAP (Wu et al., 2023), HTSAT-base
Total Transformer blocks	16
Cached / query-conditioned blocks	$L_s = 8$ / $L_q = 8$ (ablations: 4/12, 12/4, 16/8; 8/8 with a dense tail)
Model width / attention heads	256 / 8
Dense FFN width / activation	1024 / ReLU
Block ordering	pre-LN (Xiong et al., 2020)
FiLM	additive shift, zero at init; global blocks $2, \ldots, L-1$ and mask head
Sparse experts in each CacheSep tail FFN	12, top-1
Sparse expert width	1024
Router	TF-MoE (time-averaged layer input), 32-bit logits
Output	CodecSep mask head (unbounded), Eq. 6
Primary loss	CodecSep source + mixture SI-SDR, Eq. 7
MoE auxiliary	Switch balance, $\lambda_{\text{bal}} = 10^{-2}$
Optimizer	Adam, lr 1.5×10^{-4}, $\beta = (0.8, 0.99)$, no weight decay
Scheduler	ReduceLROnPlateau, factor 0.5, patience 2, floor 10^{-4}
Gradient clipping	norm 10
Batch size / steps	16 five-second mixtures / 100,000
Training runs per arm	1 (single shared seed)
Checkpoint rule	best CodecSep validation criterion (below)
Precision	bfloat16 mixed precision

The split $L_s/L_q = 8/8$ is fixed before full training; the 4/12, 12/4, 16/8 and dense-tail runs are reported as ablations and are not used to select it. Likewise, the 12-expert count, the full-width top-1 experts, and the router on the time-averaged layer input are inherited from TF-MoE, whose ablation over 3 to 24 experts peaks at 12 (Hu et al., 2026); none is tuned in this paper. Because the router assigns a whole sequence to one expert rather than each frame, inference executes one expert per layer, the property that task-level routing exploits for efficient inference (Kudugunta et al., 2021). CodecSep's validation criterion, which keeps the checkpoint and steps the learning-rate schedule for every arm, is the negated mean of three SI-SDRs: the queried sources after CodecSep's magnitude normalization against their sum, the mixture-consistency term of Eq. 7, and each clean

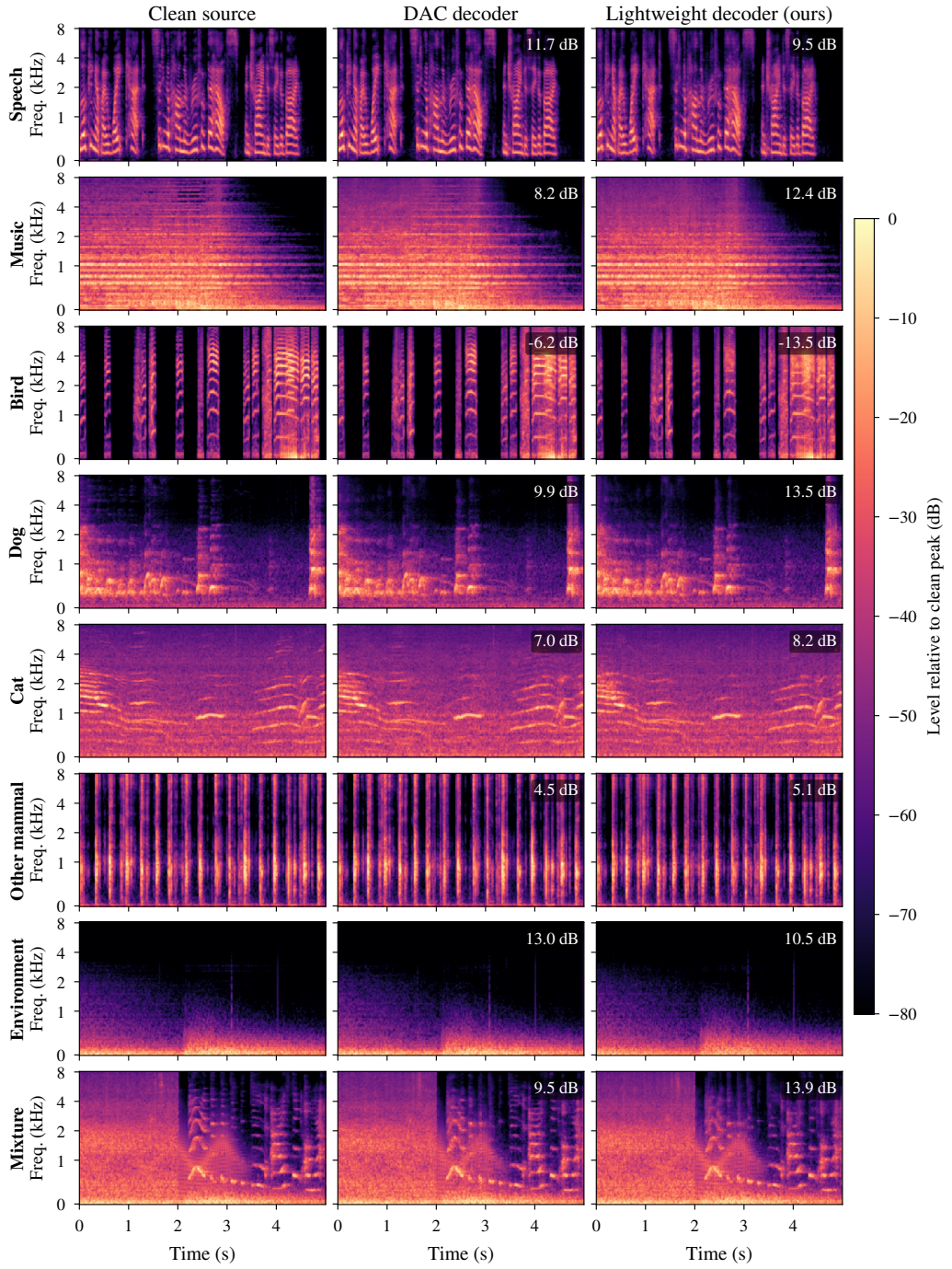


Figure 3: Log-mel spectrograms of held-out clips: clean source, DAC's decoder, and $\mathcal{D}_{\text{fast}}$, each row on the clean clip's level scale. Each decoded panel shows its SI-SDR against the clean source at the decoder's fixed delay.

source passed alone through the separator with its own query. Router logits are computed in 32-bit precision, following Switch Transformer (Fedus et al., 2022). The balance coefficient departs from TF-MoE's 10^{-3}, which TF-MoE uses without reporting expert usage, and follows Switch Transformer's 10^{-2} (Fedus et al., 2022). It was fixed before full training from two 1,000-step runs of the 8/8 configuration that differ only in this coefficient (made with the earlier post-LN, AdamW

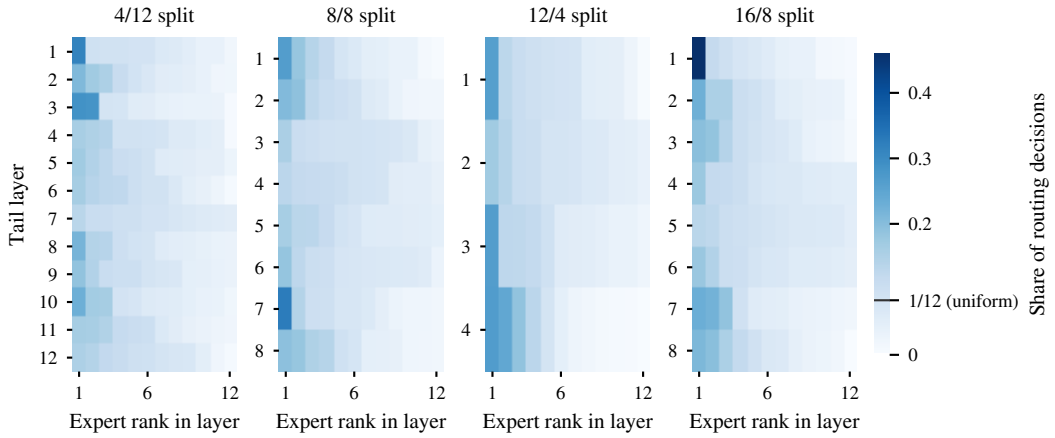


Figure 4: Expert usage per tail layer on the validation split. Each row is one layer's share of routing decisions per expert, sorted from the most to the least used expert, so a uniform layer is a flat row at 1/12 and a collapsed layer puts all its mass in the first column.

Table 10: Expert usage over the tail layers of each variant (range over layers, validation split). Largest share: fraction of routing decisions taken by the most used expert. Normalized entropy: 1 for uniform use, 0 for one expert. Dominant experts: distinct most-used experts across query classes.

System (L_s/L_q)	Expert layers	Largest share	Norm. entropy	Experts $\geq$1%	Dominant experts
CacheSep (4/12)	12	0.13–0.31	0.81–0.98	11–12	8–12
CacheSep (8/8)	8	0.13–0.32	0.85–0.97	11–12	9–12
CacheSep (12/4)	4	0.17–0.27	0.74–0.96	8–12	8–12
CacheSep (16/8)	8	0.13–0.46	0.74–0.98	11–12	5–12

recipe and not repeated), compared on training routing statistics and never on validation or test separation quality. Over training steps 501 to 1,000, the largest share of a batch's routing decisions taken by a single expert, averaged over the eight tail layers, was 0.53 at 10^{-3} and 0.40 at 10^{-2}, against $1/12 \approx 0.083$ for uniform use, with the same training loss (4.79 and 4.78). Appendix B.5 reports expert usage after full training. With $L_s = 0$ and a dense tail, our separator is the retrained CodecSep masker: loading the same weights into both gives outputs equal to numerical precision, both in the released code's post-LN ordering with its FiLM placement and in the recipe used here, a check that is part of our released test suite.

B.5 Expert usage

Figure 4 and Table 10 report how every variant with sparse experts uses them at the checkpoint it keeps, on the validation split.

Each of the 3,071 validation mixture-query pairs gives one top-1 decision per tail layer. No layer collapses onto one expert: in every layer of every variant at least 8 of the 12 experts each take 1% or more of the decisions, the most used expert takes at most 0.46 of them against 0.083 for uniform use, and the normalized routing entropy is at least 0.74. Routing also depends on the query: over the 86 query classes in the validation split, the expert most used for a class takes at least 5 distinct values in every layer.

B.6 Separator training curves

Figure 5 shows how every separator trains under the shared recipe, and Table 11 lists each run's values at the checkpoint it keeps, which is the checkpoint scored on the test set. All quantities come from the runs' own training logs and the validation split; none uses test data. The validation criterion is the checkpoint rule of Appendix B.4 with its sign flipped, so higher is better, and panel (d) shows its separation term alone. The training objective is the running mean of the SI-SDR loss of Eq. 7

over 2,500-step windows, negated, without the expert-balance term. GPU hours convert steps to time with each run's median training speed on the same GPU model, the same rate used for Table 3.

The runs follow closely matched trajectories: the training objectives nearly coincide, and the validation criteria stay within a narrow band throughout training. Plotted against GPU time, the dense 8/8 variant, which trains fastest, reaches each criterion level sooner than CodecSep. Several curves are still rising when the shared 100,000-step budget ends, so all systems are compared at an equal budget rather than at convergence. The learning-rate schedule halves the rate only for runs whose criterion stalls, and Table 11 reports each run's final rate.

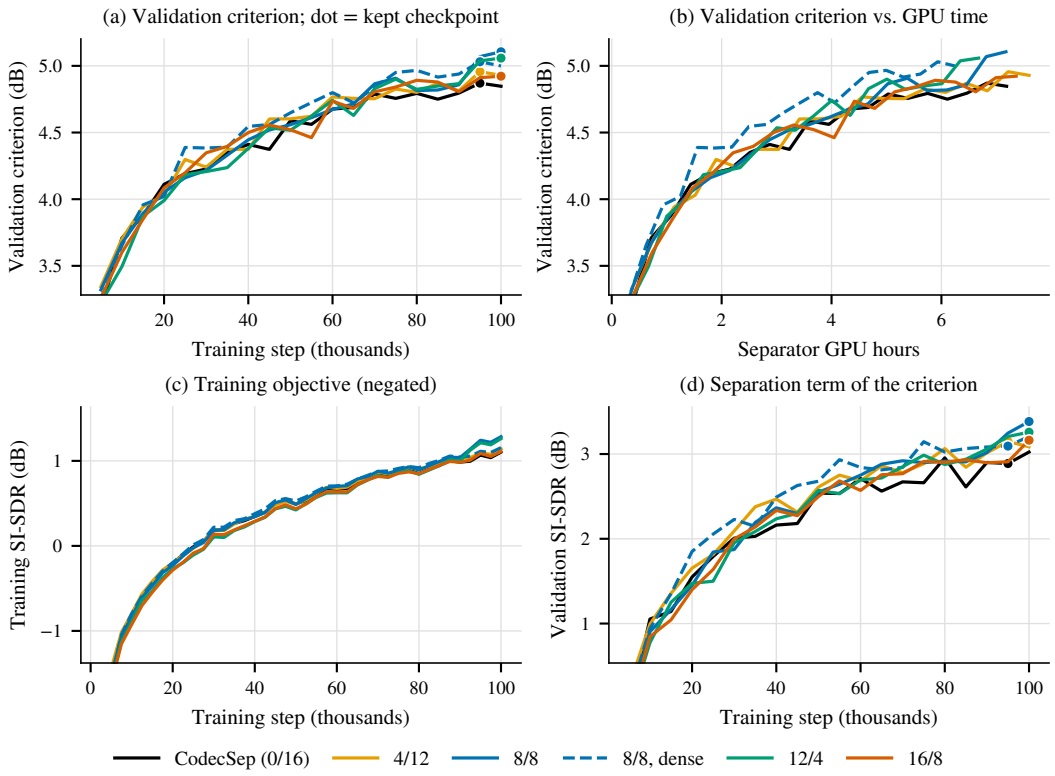


Figure 5: Separator training under the shared recipe, validation split only. (a, b) Validation criterion against training step and GPU hours; dots mark the kept checkpoints. (c) Training objective. (d) Separation term of the criterion.

Table 11: Each separator run at its kept checkpoint: validation SI-SDR terms and criterion, final learning rate, median training speed, and GPU hours for 100,000 steps.

		Validation SI-SDR at kept checkpoint (dB)						
System (L_s/L_q)	Kept step	Sep.	Mix. rec.	Rec.	Criterion	Final lr (10^{-4})	Steps/s	GPU-h
CodecSep (0/16)	95,000	2.89	6.62	5.10	4.87	1.50	3.87	7.2
CacheSep (4/12)	95,000	3.15	6.55	5.17	4.96	1.50	3.66	7.6
CacheSep (8/8)	100,000	3.38	6.64	5.30	5.11	1.00	3.87	7.2
CacheSep (8/8, dense)	95,000	3.09	6.69	5.31	5.03	1.50	4.45	6.2
CacheSep (12/4)	100,000	3.26	6.64	5.28	5.06	1.00	4.16	6.7
CacheSep (16/8)	100,000	3.16	6.49	5.11	4.92	1.50	3.77	7.4

Table 12: Compared systems; all have $L_s + L_q = 16$ except the 16/8 variant.

System	L_s	L_q	Tail FFN	Cached trunk?	Decoder	Role
CodecSep, retrained	0	16	dense	no	$\mathcal{D}_{\text{fast}}$	matched baseline
CacheSep	8	8	top-1 MoE	yes	$\mathcal{D}_{\text{fast}}$	proposed
CacheSep, dense tail	8	8	dense	yes	$\mathcal{D}_{\text{fast}}$	expert ablation
CacheSep variant	4	12	top-1 MoE	yes	$\mathcal{D}_{\text{fast}}$	split ablation
CacheSep variant	12	4	top-1 MoE	yes	$\mathcal{D}_{\text{fast}}$	split ablation
CacheSep variant	16	8	top-1 MoE	yes	$\mathcal{D}_{\text{fast}}$	trunk-depth ablation
AudioSep, released	–	–	–	–	waveform	external reference

C Baselines

C.1 Design of the matched comparison

The setup is built so that CodecSep and CacheSep differ only in where the query enters the network and in the tail feed-forward layers, and so that every shared choice comes from the baseline's recipe rather than being tuned for ours.

- **Recipe.** Every shared setting, from the latent, mask head, CLAP conditioning, and objective to the optimizer, learning rate, schedule, gradient clipping, validation criterion, and checkpoint rule, is CodecSep's, taken from its paper or official code (Table 9). None was tuned for CacheSep. Where CodecSep's paper and released code disagree we follow the paper unless we measured that the code's choice matters, and the one change to both, pre-LN ordering, applies to every system and is justified below by measurements on the baseline.
- **Data.** All systems train from the same seed on the same on-the-fly mixture stream and are tested on the same frozen mixtures and queries.
- **Budget.** All systems run the same number of steps with the same batch and crop length, and each is evaluated from its best checkpoint under the same validation criterion.
- **Decoder.** The decoder is trained and frozen before any separator run, and every codec-latent system decodes through it in training and evaluation, so no system gains from a better decoder.
- **Capacity.** Each query of CacheSep executes blocks of exactly CodecSep's width, so its experts add stored parameters while its active parameters stay at CodecSep's level (Table 3), and the dense 8/8 control tests whether the stored capacity matters.
- **Evaluation.** All systems are scored with the same fixed-delay metrics and the same macro average, and every matched difference is a paired interval over identical targets. The 8/8 split was fixed before any test evaluation, and no configuration is selected on the test set.
- **Cost.** All systems are timed in one harness on one GPU with the same measurement procedure (Appendix E), with prompt embeddings cached for every system, AudioSep included, and operations counted by the same counter.

We retrain CodecSep rather than compare with its published results, because those use other training data, other test mixtures, and DAC's original decoder; a difference against them would mix these factors with the architectural one we study.

C.2 Matched CodecSep retraining

CodecSep is retrained from the authors' official implementation (Banerjee & Arora, 2026), commit `662e910`. We keep its 16-block FiLM-conditioned Transformer masker, frozen CLAP text conditioning, mask head, source SI-SDR term, and decoded-mixture consistency term. The deviations needed for a controlled comparison are our mixtures and 5 s generator, the frozen DAC-16k latent and the lightweight decoder $\mathcal{D}_{\text{fast}}$ shared by all codec-latent systems, and the common training budget. Where the CodecSep paper and its released code disagree, we follow the paper unless we measured that the code's choice matters, and every codec-latent system uses the same recipe (Table 9).

Published description versus released code. The paper applies FiLM to blocks $2, \ldots, L-1$, whereas the code conditions all L; we follow the paper. The paper states that the FiLM shift is zero at initialization, whereas the code initializes the FiLM weights with Xavier and only the biases to zero, so the shift is nonzero for any nonzero text embedding; we zero the weights as well, which makes the paper's statement true. The paper uses Adam, whereas the code uses AdamW with PyTorch's default weight decay; we follow the paper, keeping the code's $\beta = (0.8, 0.99)$ because the paper gives none. The paper mentions Snake activations in the Transformer without saying where; the code's feed-forward sublayers use ReLU, and so do ours. Two choices follow the code or depart from both, for the reasons given below: the mask is unbounded, as in the code, where the paper describes $\mathbf{M}_s \in [0,1]^{d \times T}$; and every block is pre-LN, where the paper and the code are post-LN. The released configuration itself requests pre-LN ordering and a 10,000-step warmup, but the code reads neither value.

Why the mask stays unbounded. A $[0, 1]$ mask is too restrictive for this latent, which is not additive over sources. We measured this with oracle masks on 488 validation pairs, decoding each through $\mathcal{D}_{\text{fast}}$ and scoring it against the codec reference. For 28.1% of latent coordinates the ideal ratio between source and mixture latents is negative, which no $[0, 1]$ mask can express. The ideal ratio clipped to $[0, 1]$ reaches 5.43 dB SI-SDR, clipped only below at zero it reaches 10.07 dB, and clipped to $[-5, 5]$ it reaches 18.90 dB, against -7.39 dB for the unprocessed mixture. The overall scale of the mask is not a concern either, because $\mathcal{D}_{\text{fast}}$ is nearly scale-invariant: decoding the latent scaled by any factor from 0.5 to 35 matches the unscaled decode to at least 34.7 dB SI-SDR.

Why every block is pre-LN. In the post-LN ordering, the masker's output loses its variation over time early in training, so the mask becomes a fixed per-query channel weighting. The CodecSep paper reports masks of this kind as "predominantly channel-wise rather than strongly time-localized". We measured the time variation (standard deviation over frames divided by mean magnitude, averaged over channels) on a fixed validation batch in two short training runs with the released code's settings that differ only in block ordering. After 750 steps it is 0.017 at the last post-LN block (0.63 at initialization) and 0.005 in the mask, against 0.679 (0.94) and 0.193 with pre-LN (Xiong et al., 2020), whose training loss is also lower (2.75 against 3.59). Pre-LN moves the same parameters and FiLM shifts into a different order and adds none, so CodecSep and CacheSep both use it.

C.3 AudioSep

AudioSep uses the released public checkpoint and official inference path (Liu et al., 2025). The same model, retrained on Clotho and FSD50K, is the baseline of the DCASE 2024 language-queried audio source separation task (DCASE 2024 Task 9 organizers, 2024). It is also the external system CodecSep compares against, so both codec-latent systems are placed against the same reference. In CodecSep's experiments, CodecSep attains higher SI-SDR than AudioSep, both released and retrained on CodecSep's training data, with comparable perceptual quality (ViSQOL); most of these comparisons run AudioSep on codec-processed audio (Banerjee & Arora, 2026). It receives the same fixed test mixture and the same canonical text query as the codec-latent systems. Each 16 kHz mixture is resampled to AudioSep's 32 kHz by band-limited Fourier resampling, separated, and resampled back to 16 kHz before scoring, at no synthesis delay. Its CLAP prompt embeddings are cached per class, as for the codec-latent systems, so its timing counts the same work per query. No AudioSep parameter is retrained on our corpus. Because this is not a matched-data comparison, its result is reported separately from the CodecSep architectural contrast.

C.4 Systems not compared

Generative separators synthesize the target rather than estimate it from the mixture. FlowSep samples a VAE latent of the target's mel spectrogram by rectified flow matching and renders it with a vocoder (Yuan et al., 2025), SoloAudio performs latent diffusion with a Transformer backbone (Wang et al., 2025), and SAM Audio samples a DAC-VAE latent with a flow-matching diffusion Transformer of 0.5 to 3 billion parameters (Shi et al., 2025). We do not compare with them for two reasons. First, their outputs are not aligned sample by sample with the target: FlowSep states that sample-level metrics such as SDR do not suit its outputs, and FlowSep and SAM Audio report distributional, embedding-based, learned, and subjective measures instead of SDR, so our waveform

metrics would not measure what these systems are built to do. Second, sampling evaluates the conditioned generator at least once per solver step (10 steps in FlowSep's reported setting and 16 in SAM Audio's), so their per-query cost is set by the sampler, and which of their computation could be shared across queries on one mixture is a different design question from the one studied here. Multi-prompt separators such as TUSS and FasTUSS share computation across a prompt set given in advance (Saijo et al., 2025; Paissan et al., 2025); when all prompts are known together, every system here can batch them instead, and Appendix E.2 reports that regime. Among discriminative systems, earlier extractors select sounds by class label (Ochiai et al., 2020), CLIPSep learns text-queried separation from unlabelled videos through CLIP's joint image and text space (Dong et al., 2023), and OmniSep accepts text, image, and audio queries (Cheng et al., 2025). Their contributions concern the query modality and training signal rather than inference cost, so we use AudioSep, a released text-queried separator trained on large general-audio data and the system CodecSep itself compares against (Banerjee & Arora, 2026), as the single external reference.

C.5 Relation to prior codec-latent separators

CacheSep does not propose a new codec, separation objective, conditioning mechanism, or expert layer. It keeps DAC, CodecSep's mask and SI-SDR objective, CLAP conditioning through FiLM, and TF-MoE's expert layer, and its contribution is where the query enters the network and what this permits at inference. Prior codec-latent separators fall into three groups. CodecFormer separates a fixed number of speakers in codec embeddings and takes no query (Yip et al., 2024). SD-Codec and SUNAC change the codec itself: SD-Codec assigns fixed source types to separate codebooks (Bie et al., 2025), and SUNAC encodes individual sources from the mixture conditioned on source-type prompts, so the sources to extract are chosen when the mixture is encoded (Aihara et al., 2026). CodecSep keeps DAC frozen and accepts an open text query, but its FiLM layers condition every Transformer block except the first and the last, so only the first block's output is independent of the query and the rest of the masker runs again for each query (Banerjee & Arora, 2026).

CacheSep differs from all of them in one structural property. The separator is split at a fixed depth, and every block before the split is prohibited from receiving query information, so its output is a function of the mixture alone (Section 3.3). This yields three properties that no system above provides together with open text queries and a frozen codec.

- **Queries need not be known in advance.** The trunk state is computed once per mixture, before any query arrives, and serves every later query in any order. TUSS, FasTUSS, and SUNAC prepend the prompts to the encoded mixture and process both together in their first Transformer module, so they share computation only within a prompt set supplied together. FasTUSS also analyzes a variant whose attention keeps the mixture and the prompts apart in that module, which makes its mixture features independent of the prompts, but it studies this variant as a conditioning choice, not as a way to reuse computation across queries (Paissan et al., 2025).
- **Each answer is independent of the other queries.** The cached tensors are bit-identical for every query sequence, so the estimate for a query equals the estimate that query would receive alone. In models that process a prompt set jointly, the estimate for one prompt can depend on the other prompts in the set.
- **The codec is unchanged.** CacheSep operates on the latent of an existing general-audio codec without retraining it, unlike SD-Codec and SUNAC, and the same frozen decoder serves any separator on that latent.

The other two elements make this split useful rather than merely possible. The cost model of Section 3.1 shows that the decoder, not the separator, bounds the saving from caching, so with DAC's original decoder caching the trunk saves only 1.3% of a later query's operations; the lightweight decoder raises this to 27.1%. Moving blocks into the trunk also removes query-conditioned depth, and the sparse tail adds query-specific capacity at the active cost of CodecSep's layers. TF-MoE routes each sequence to one expert, and placing its layer only after the query enters means the pooled features it routes on already carry the query, while the trunk stays query-independent and each query executes one expert per layer.

Against CodecSep retrained under the same recipe, every cache split we evaluate keeps SI-SDRi within 0.11 dB and later queries run up to 2.10× faster end to end; the 8/8 reference configuration

needs 1.78× fewer separator operations per later query and runs 1.28× faster (Section 5.2). The design does not claim higher separation quality than CodecSep. It gives little saving when a mixture is queried once, since the first query runs every block, and when all prompts are known together, batching them narrows the difference (Appendix E.2).

D EVALUATION PROTOCOL

The frozen test mixtures are generated once from a seed that does not depend on the training seed. Every trained system therefore sees identical mixture waveforms, target waveforms, source labels, and text prompts. The primary statistic is macro SI-SDRi over the 89 classes, computed against the original raw target waveform.

A fixed synthesis delay, if present, is measured independently for each frozen decoder path before scoring and then corrected identically for every system using that path. No per-example lag search is allowed at test time. For paired intervals, target recordings are resampled within class and all query pairs whose target is a resampled recording enter together; this avoids treating repeated crops of the same held-out recording as independent observations. The bootstrap uses $10{,}000$ resamples.

Appendix F reports per-category and per-class results; they are descriptive and are not used for model selection.

E INFERENCE COST MEASUREMENT

All timing uses one NVIDIA RTX PRO 6000 Blackwell Workstation Edition GPU with PyTorch 2.11.0 and CUDA 12.8, batch size one for sequential-query latency, and the same numerical precision used by the released inference configuration. Each component is warmed up for 20 iterations, synchronized before and after each measurement, and timed over 100 repetitions; we report the mean. Operation counts are computed for the same 5 s input shape used by the latency benchmark. Timing and operation counts use randomly initialized weights, since neither depends on the weight values, so the cost results hold for any training run and run-to-run variance affects only the quality comparisons.

For codec-latent systems, the first-query path contains DAC encoding, the cached trunk when present, one tail pass, and $\mathcal{D}_{\text{fast}}$; RVQ is not on the matched separator path. A later query contains only the query-conditioned tail and $\mathcal{D}_{\text{fast}}$; canonical CLAP embeddings are precomputed for the fixed 89-prompt evaluation vocabulary. We separately time online CLAP text encoding for a raw unseen prompt so that this cost can be added when the deployment cannot precompute prompt embeddings. Peak allocated GPU memory and the size of persistent cached tensors are recorded after warm-up.

E.1 SCOPE OF THE CONTROLLED COMPARISON

The decoder is trained on the same broad domain used to build separation mixtures. That alignment is what the system needs, but it means reconstruction quality outside those sound domains is not established here. The decoder is also an adaptation of a published generator to the DAC latent geometry rather than a new vocoder architecture, and it keeps that generator's published objective unchanged, so it is evidence about the transfer of that recipe rather than about decoder design.

The comparisons test a reference design fixed before training against a matched non-cached baseline, a dense-tail control at the same split, splits that cache less (4/12) and more (12/4) of the same 16 blocks, and a deeper trunk over the same tail (16/8). They therefore do not determine an optimal expert count, routing function, or decoder architecture. The split L_s/L_q and the expert count are fixed before any final test evaluation, so the test set measures the reported comparisons rather than selecting among them.

E.2 FULL REPEATED-QUERY TIMING RESULTS

Table 13 gives the full timing curve in both query regimes, the per-mixture cache, and peak GPU memory. When all prompts are known together, every system may batch them, so the batched rows

show how much of CacheSep's advantage survives when a baseline can also share work across a known prompt set.

The batched rows also explain why AudioSep's single-query latency on our GPU is close to CacheSep's despite its larger operation count (Table 3). AudioSep's batched time grows in proportion to K, so its arithmetic already occupies the GPU at a few queries, and its time per query stays near its single-query time. A single codec-latent query instead leaves the GPU mostly idle, and its latency is set by the fixed overhead of launching many small operations, so its time per query falls by more than an order of magnitude as queries are batched. Latency at one query per call therefore understates the difference in operations; where arithmetic rather than launch overhead limits speed, as for AudioSep here from two queries on, cost follows the operation count.

Table 13: End-to-end latency for one 5 s mixture and K queries. Top: queries arrive one at a time (total ms), the setting caching targets. Bottom: all K queries in one batch (ms per query). Cache: per-mixture state reused by later queries. Peak: GPU memory at the largest K.

System	$K=1$	$K=2$	$K=4$	$K=8$	$K=16$	$K=32$	$K=64$	$K=89$	Cache (MB)	Peak (MB)
Queries one at a time: total ms										
AudioSep, released	6	12	23	46	92	183	366	509	0.00	33667
CodecSep, retrained	10	16	29	56	110	218	434	602	0.98	1873
CacheSep (8/8)	10	15	26	47	89	174	344	474	1.10	1635
CacheSep (8/8, dense tail)	9	12	19	34	64	123	240	332	1.10	1110
CacheSep (4/12)	11	18	33	62	121	238	472	659	1.10	1901
CacheSep (12/4)	10	12	18	31	57	109	212	291	1.10	1370
CacheSep (16/8)	12	17	28	49	91	175	348	476	1.10	1661
All K queries in one batch: ms per query										
AudioSep, released	5.7	4.6	4.4	4.3	4.4	4.5	4.5	4.6		
CodecSep, retrained	13.9	7.0	3.5	1.8	0.9	0.5	0.4	0.4		
CacheSep (8/8)	14.5	7.3	3.6	1.8	0.9	0.5	0.3	0.3		
CacheSep (8/8, dense tail)	12.9	6.5	3.3	1.6	0.8	0.5	0.3	0.2		
CacheSep (4/12)	15.6	8.0	4.0	2.0	1.0	0.5	0.3	0.3		
CacheSep (12/4)	13.4	6.7	3.4	1.7	0.9	0.5	0.3	0.2		
CacheSep (16/8)	16.4	8.3	4.1	2.1	1.1	0.6	0.3	0.3		

F PER-CATEGORY AND PER-CLASS RESULTS

Table 14 breaks SI-SDRi down by category for the reference configuration (8/8), CodecSep, and AudioSep, beside the decoder ceiling. A category score is the macro average over its classes, and its paired interval uses the same stratified bootstrap over target recordings as the headline, so the last row reproduces Table 4. Against CodecSep, the interval lies entirely above zero for 3 and entirely below zero for 0 of the 11 categories on the clean reference, and for 1 and 1 on the codec reference, without correction for multiple comparisons. Table 15 lists every class in the order of Table 5, as point estimates without intervals.

Table 14: SI-SDRi (dB) by category on the test set. Ceiling: $\mathcal{D}_{\text{fast}}$ on the clean target's latent. Δ: paired difference, 95% interval, bold if it excludes zero. AS: AudioSep; CS: CodecSep.

	SI-SDRi, clean reference				Δ = 8/8 − CS	
Category	Ceiling	AS	CS	8/8	Clean reference	Codec reference
Human	10.53	4.42	3.18	3.06	−0.13 [−0.25, +0.00]	**−0.18 [−0.30, −0.05]**
Music	11.80	5.70	4.87	4.97	**+0.10 [+0.01, +0.19]**	+0.02 [−0.07, +0.11]
Machines/tools	4.85	2.31	−0.03	0.19	**+0.23 [+0.15, +0.30]**	**+0.23 [+0.15, +0.31]**
Animal	3.46	2.67	−1.79	−1.81	−0.02 [−0.23, +0.19]	−0.04 [−0.20, +0.12]
Background noise	8.11	−8.13	−0.69	−0.54	**+0.15 [+0.03, +0.27]**	+0.11 [−0.01, +0.24]
Vehicles/road	10.81	4.83	2.08	2.12	+0.04 [−0.06, +0.14]	+0.01 [−0.08, +0.10]
Media	8.41	−1.67	1.21	1.25	+0.04 [−0.12, +0.20]	+0.06 [−0.09, +0.20]
Objects/materials	4.07	1.71	−1.30	−1.23	+0.07 [−0.07, +0.21]	+0.09 [−0.04, +0.23]
Natural environment	7.27	2.26	−0.34	−0.20	+0.13 [−0.05, +0.32]	+0.16 [−0.01, +0.35]
Alerts/electronics	4.10	5.06	−1.64	−1.79	−0.15 [−0.49, +0.16]	−0.07 [−0.37, +0.21]
Miscellaneous	10.20	−3.30	−0.44	−0.53	−0.09 [−0.37, +0.18]	−0.04 [−0.32, +0.22]
All	6.95	2.80	0.78	0.86	**+0.07 [+0.02, +0.13]**	**+0.06 [+0.01, +0.10]**

	SI-SDRi, clean reference				Δ = 8/8 − CS	
Category	Ceiling	AS	CS	8/8	Clean reference	Codec reference

Table 15: SI-SDRi (dB) per class on the test set, point estimates. Recs.: distinct test target recordings. Ceiling as in Table 14. AS: AudioSep; CS: CodecSep.

				Clean ref.			Codec ref.	
Category	Class	Recs.	Ceiling	AS	CS	8/8	CS	8/8
Human	Human speech	100	8.8	5.5	4.9	5.1	6.8	7.0
	Human singing	95	14.7	5.0	3.7	2.8	4.7	3.8
	Human body actions	95	4.8	2.6	0.5	0.8	3.1	3.2
	General human sounds	87	11.6	−0.2	2.0	1.9	4.2	4.0
	Children playing	67	10.3	2.4	2.8	2.7	4.0	4.0
	Crowd/human group	11	12.5	8.5	4.9	4.6	5.7	5.2
	Flatulence	49	10.9	7.1	3.4	3.5	4.4	4.5
Music	Bass	95	6.1	1.2	7.1	7.5	7.5	7.5
	Keyboard	94	12.0	3.5	8.1	8.4	9.2	9.2
	Organ	91	11.3	8.7	6.6	7.1	8.1	8.3
	Guitar	92	7.7	3.8	7.1	7.4	8.1	8.4
	Percussion	96	5.7	5.1	−4.0	−4.2	2.4	2.5
	Musical ensemble	100	14.6	3.9	3.4	3.5	4.0	4.1
	Bowed strings	94	19.5	6.6	8.2	8.2	8.3	8.1
	Reed instruments	83	17.2	7.6	8.7	8.5	9.2	9.0
	Brass	85	9.6	3.9	2.5	2.3	5.7	5.6
	Flute	78	9.2	8.1	6.2	6.6	10.1	10.3
	Wind/brass instruments	84	12.3	10.2	4.6	4.5	5.9	5.9
	Synth lead	94	5.3	4.7	3.4	3.9	7.9	8.1
	Plucked guitar/strings	89	14.0	7.3	4.5	4.6	4.8	4.9
	Keyboard instruments	89	13.6	0.6	0.2	−0.1	0.3	0.1
	Piano	42	19.0	10.1	6.4	6.2	6.2	6.0
Machines/tools	Mechanical fan	94	14.2	0.5	4.1	4.3	4.1	4.3
	Pump	86	9.9	2.8	2.6	2.7	3.4	3.5
	Slider	96	12.7	−1.3	3.5	3.7	3.6	3.9
	Valve	95	5.3	0.2	1.3	1.4	2.5	2.7
	Home appliances	92	6.2	0.9	−0.6	−0.7	2.2	2.2
	Mechanical devices	93	4.3	0.7	−2.9	−3.1	1.6	1.4
	Power tools	93	0.5	5.4	−3.4	−3.5	2.2	2.1
	Mechanisms	79	4.4	0.4	−1.3	−1.0	2.2	2.5
	Lawn mower	70	9.9	4.9	2.5	2.7	3.6	3.8
	Drilling	75	1.8	6.7	−2.0	−2.1	3.2	3.1
	Air conditioner	59	10.9	−2.3	3.2	3.1	3.5	3.3
	Jackhammer	60	2.0	6.9	−1.0	−0.8	3.1	3.3
	Vacuum cleaner	63	2.9	3.6	−0.4	−0.2	3.5	3.6
	Chainsaw	30	2.4	8.4	−1.0	−0.8	2.4	2.5
	Dryer	30	2.0	1.3	−0.2	0.5	4.7	5.3
	Mechanical ticking	72	5.1	5.5	0.9	1.4	4.3	4.7
	Wind turbine	32	−5.6	−8.1	−0.9	−0.7	−1.9	−1.7
	Saw	14	0.2	8.1	−1.0	−0.5	4.2	4.7
	Grinder	42	−4.5	6.9	−6.2	−6.1	1.9	1.9
	Printer	15	2.6	5.6	−2.7	−2.2	2.3	2.7
	Blender	42	−0.7	3.0	−3.4	−3.0	2.3	2.7
	Appliance fan	20	10.3	−3.2	2.9	3.1	3.9	4.5
	Microwave	27	14.7	−3.6	5.3	6.3	6.1	7.2
Animal	Insects	100	−11.9	1.0	−14.8	−15.3	5.7	4.6
	Birds	99	−3.9	5.1	−11.4	−10.4	4.2	4.3
	Dogs	97	13.5	7.1	4.2	4.5	6.4	6.7
	Marmosets	97	11.0	−25.9	8.0	8.4	10.0	10.3
	General animal sounds	97	10.1	1.6	−0.3	−0.5	1.9	1.7
	Cats	90	7.1	−0.3	−0.7	−1.2	2.2	1.8
	Great tits	82	−27.1	12.8	−24.8	−25.8	14.1	14.8

Category	Class	Recs.	Ceiling	Clean ref.			Codec ref.	
				AS	CS	8/8	CS	8/8
	Primates	96	8.6	6.9	15.4	15.6	10.2	10.2
	Frogs	61	−0.8	1.8	−6.7	−7.0	0.5	0.2
	Wild animals	68	6.6	5.9	−0.6	−0.9	5.2	5.6
	Horses	44	9.7	3.5	2.0	2.0	2.9	3.0
	Zebra finches	69	0.8	7.0	−0.3	−0.2	9.6	9.2
	Sheep/goats	43	4.0	7.5	−0.9	−0.3	3.5	3.9
	Pigs	26	4.7	6.7	0.5	0.9	4.7	5.1
	Cows	25	12.5	−0.2	3.6	3.2	4.0	3.5
	Rodents	19	−1.4	−0.2	−9.3	−9.2	1.6	1.2
	Animal growling	11	15.4	4.9	5.7	5.4	5.9	5.6
Background noise	General noise	98	8.1	−8.1	−0.7	−0.5	1.9	2.0
Vehicles/road	Road vehicles	100	10.7	2.4	1.7	2.1	3.2	3.5
	Vehicle engines	95	7.7	4.6	0.6	0.4	2.7	2.5
	Aircraft	90	9.4	4.6	0.3	0.4	2.3	2.5
	Road ambience	95	13.7	3.5	5.0	5.3	5.7	5.9
	Trains	79	9.8	6.6	2.3	2.2	3.6	3.4
	Boats	22	13.4	7.4	2.7	2.3	3.6	3.3
Media	Video-game sound effects	73	7.8	−1.9	−0.2	−0.2	1.3	1.3
	General sound effects	97	10.9	−2.6	1.3	1.4	2.9	3.0
	Television/radio	91	8.3	−2.3	0.1	−0.2	2.4	2.0
	Effects unit	74	11.9	−4.7	2.5	2.6	3.3	3.4
	Spray sound	73	5.3	0.5	−2.6	−2.2	1.8	2.1
	Sine wave	50	2.8	11.6	3.9	3.9	7.3	7.5
	Chorus effect	66	11.8	−12.4	3.4	3.4	4.1	4.0
Objects/materials	Doors/drawers	98	7.5	1.1	0.6	0.8	3.1	3.3
	Fire/explosions	93	7.7	3.6	1.3	1.5	2.7	3.0
	Glass/metal sounds	93	1.2	0.7	−5.3	−5.1	1.4	1.6
	Impacts/crashes	88	3.0	3.1	−0.1	−0.2	3.5	3.4
	Kitchen/dishes/cooking	83	0.7	3.3	−2.7	−2.4	2.9	3.5
	Paper/plastic rustling	80	4.8	1.2	1.2	1.4	4.0	3.9
	Squeaks/creaks	51	5.9	−1.7	−1.4	−1.0	3.1	3.4
	Screeching/friction	58	1.8	2.4	−4.1	−4.7	2.3	1.5
Natural environment	Water	98	4.8	2.3	−1.7	−1.7	1.4	1.4
	Wind/weather	87	9.7	2.2	1.0	1.2	2.2	2.4
Alerts/electronics	Alarm/siren/horn	93	4.1	5.2	−2.8	−3.1	5.0	4.8
	Phone/electronic alert	88	4.1	4.9	−0.5	−0.5	4.6	4.6
Miscellaneous	Miscellaneous sounds	99	10.2	−3.3	−0.4	−0.5	1.5	1.5

G SEPARATION EXAMPLES

Figures 6 and 7 show four mixtures separated by the reference configuration (8/8), CodecSep, and AudioSep. They illustrate the task and are not a basis for comparison, which Tables 4 and 14 provide. Each mixture is new and combines three held-out test sources at the evaluation's loudness targets: an animal (bird, dog, zebra finch, or cat), a human sound (speech, singing, or a crowd), and a music source. The examples are chosen by a rule that treats the three systems alike and aims at typical rather than best cases. For each animal we drew 64 candidate mixtures with a fixed seed and separated each with all three systems, and we show the candidate whose SI-SDRi, averaged over the three systems and the three queries, lies closest to the median of that average, without repeating a human or music class. Every system runs exactly as in its test evaluation, and the same pipeline applied to an existing test mixture reproduces each system's evaluated scores. 8/8 and CodecSep, which share the decoder and the training recipe, produce closely similar outputs, and both keep faint residue of the other sources where the clean target is silent. AudioSep's outputs are often visibly cleaner; AudioSep was trained on 13.02× as much data and with a different model and recipe, and these factors may account for part of the difference. The figures therefore show how each system separates a mixture and are not a controlled comparison with AudioSep.

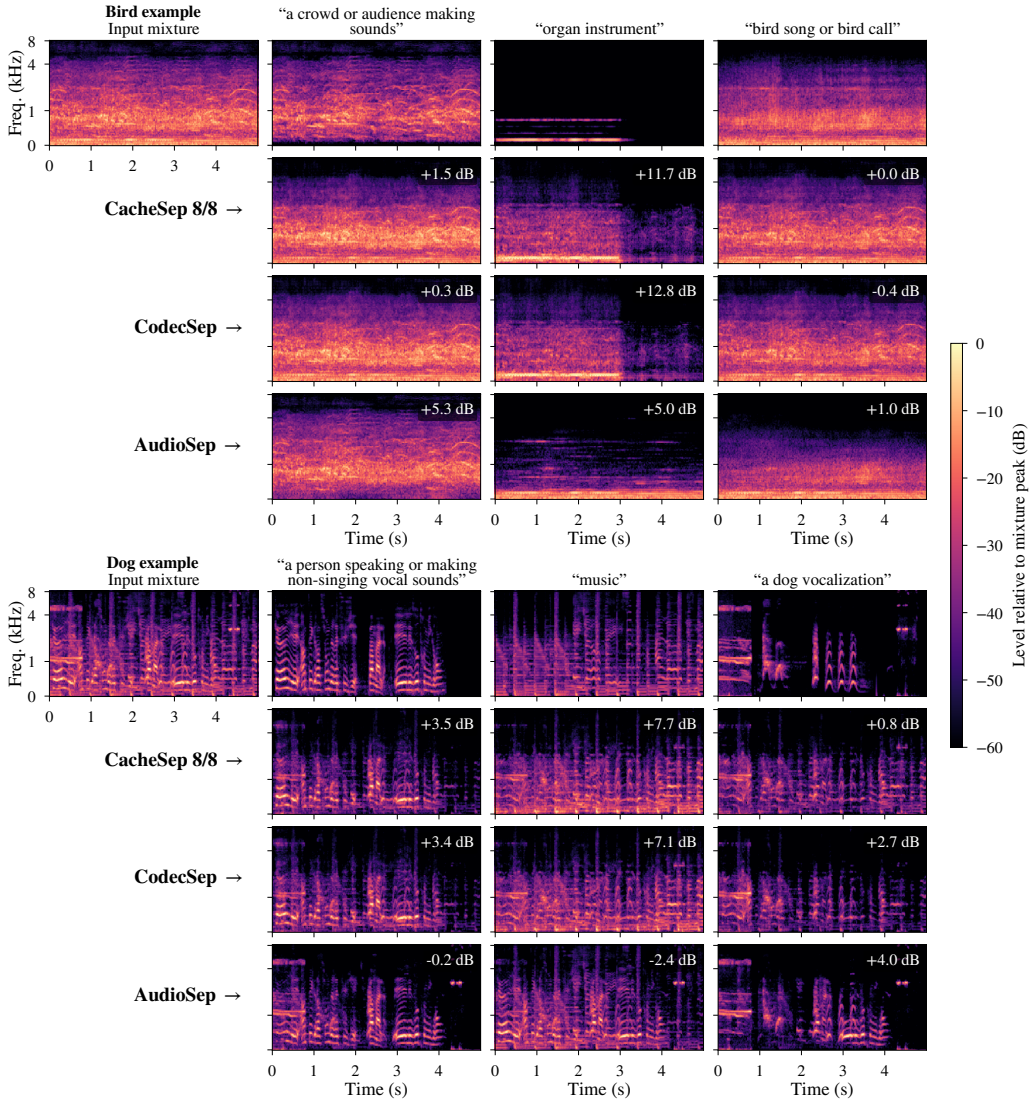


Figure 6: Separation examples with a bird and a dog. In each example, the top row shows the input mixture and, under each query's text, that query's clean target; each row below shows one system's output for the query and its SI-SDRi against the clean target. Log-mel spectrograms on the mixture's level scale.

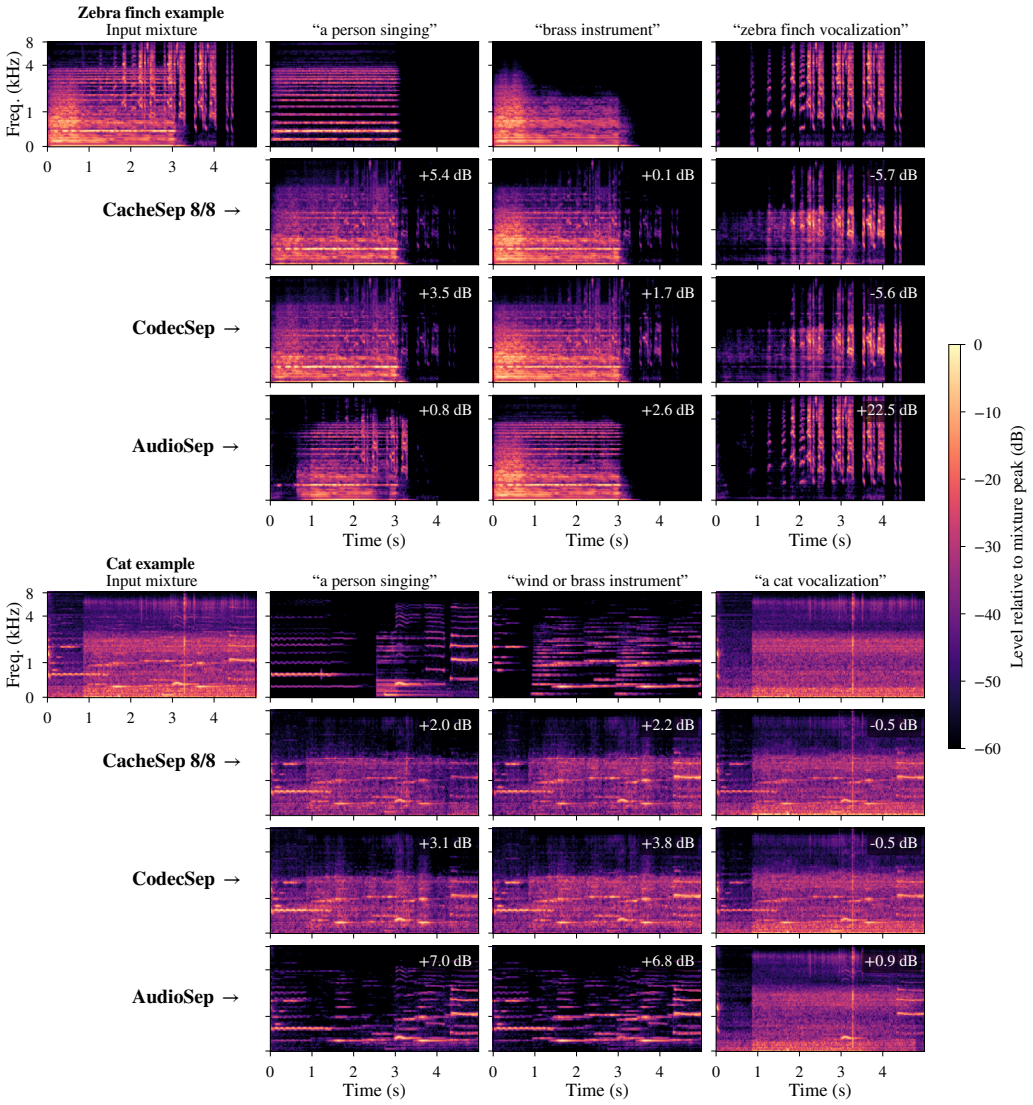


Figure 7: Separation examples with a zebra finch and a cat, laid out as in Figure 6.