Supplementary Material

A. Experimental Details

A.1. Implementation Details

Input log-magnitude spectrograms are computed from 16 kHz audio using a 1024-point FFT, a 64 ms window, and a 256-sample hop size, corresponding to 8.192-second clips. For the visual modality, video streams are sampled at 4 FPS for the scene stream ($\boldsymbol{v}^s$) and 25 FPS for the facial stream ($\boldsymbol{v}^f$). All frames are resized to 224×224 pixels before being fed into their respective encoders.

The facial encoder is adopted from AVDiffuSS [37], pretrained on VoxCeleb2 [11] for the audio-visual speech separation task. The scene encoder follows the design of CAVP [41], pretrained on the VGGSound dataset via contrastive audio-video alignment. Both visual encoders are frozen during training, while the fusion module is jointly optimized with the vector field estimator $\boldsymbol{u}_\theta$ to obtain a unified multimodal representation $\boldsymbol{c}^V$ from the two visual inputs.

A.2. Baseline Adaptation for CASS

DAVIS-Flow [29] originally processes a single visual input and predicts only the corresponding target source. To adapt it to the CASS setting, we use the model to predict the dialogue (DX) and effects (FX) components, and treat the residual (mixture minus predicted DX and FX) as the music (MX) estimate.

B. Pseudo Code for Training and Inference

We provide the pseudo-code for training and sampling processes of AV-CASS in Alg. 1 and Alg. 2 in this technical appendix. CAT$(\cdot)$ in the provided algorithms indicates channel-wise concatenation operation.

C. Statistics of the Training Data

We synthesize 10k training samples, each 60 seconds long, following the pipeline described in *Training Data Construction Pipeline* in the main paper. Fig. 7 presents the statistics of the training data from our training data construction pipeline. The loudness distribution of mixture audio shows an average loudness of –27 LKFS with a true peak value of –2 dBFS, ensuring that the synthesized cinematic audio closely resembles the loudness characteristics of real-world film audio [60]. We also report the number of segments per stem type contained in a 60-second mixture, along with the duration distribution of individual segments for each stem.

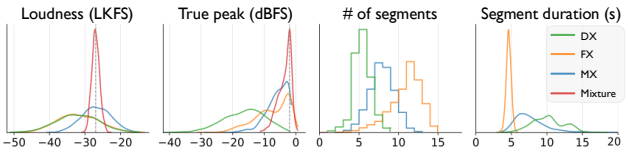

Figure 7. Distributions of the synthesized training data for loudness in LKFS, true peak in dBFS, number of segments for each stem, and the duration of each segment in seconds.

D. Details of MOS evaluation

We conducted a Mean Opinion Score (MOS) test as a user study to evaluate the perceptual quality of audio separated by various methods. This assessment was performed on both real-world audio-visual samples from movies and synthetic samples from the AVDnR dataset in separate studies. The primary motivation for including MOS on real-world data is the absence of ground-truth (GT) references, which makes objective evaluation nearly impossible. In contrast, the AVDnR evaluation helps reveal how well objective metrics align with human judgment, as it includes clean, separated source tracks. For real-world samples, we compare with the existing CASS methods, while for AVDnR, we also include the music source separation models, *i.e.*, HDemucs, HTDemucs, and MSDM.

D.1. Sample Selection

For real-world samples, we selected 30 audio-visual segments from the Condensed Movies dataset [4]. Each segment was 4 to 10 seconds long and was chosen to contain all three target separation stems: DX (dialogue), FX (sound effects), and MX (music).

For AVDnR, we select 30 5-second-long audio clips from synthetic AVDnR test set. Since ground-truth stems are available, we show the GT audio for the specified stem prior to evaluating each sample. This allows listeners to score each sample based on the comparison with the given GT.

D.2. Test Procedure

All audio clips were volume-normalized to minimize loudness bias. Participants listened to samples in randomized order, and each clip was evaluated only once. The instructions shown to the participants before the start of the test are displayed on the left side of Fig. 8 for real-world samples, and the left side of Fig. 9 for AVDnR samples. After carefully reading the instructions, the participants start the evaluation.

For real-world samples, as shown on the right side of

```
# --- Setup ---
# VFE_model (u_theta): torch.nn.Module
# optimizer: torch.optim.Optimizer
# criterion: torch.nn.MSELoss (L2 loss)
# D: torch.utils.data.DataLoader
# visual_fusion: Function/Module for C_V = F_theta(E^f(v_f) || E^s(v_s))

VFE_model.train()

for s_A, v_f, v_s, s_DX, s_FX, s_MX in D:
    # 1. Compute fused visual condition vector c_V
    c_V = visual_fusion(v_f, v_s)

    # 2. Sample time step t (according to t = 1/(1 + exp(-s)), s ~ N(0, 1))
    s = torch.randn_like(s_DX)
    t = torch.sigmoid(s)

    # 3. Sample initial noise x_0 and concatenate target sources x_1
    x_0 = torch.randn_like(s_DX)
    x_1 = torch.cat([s_DX, s_FX, s_MX], dim=1) # Target clean sources

    # 4. Compute the point at time step t: x_t = (1-t)x_0 + t*x_1
    # Adjust t's shape for proper broadcasting across audio dimensions
    x_t = (1 - t) * x_0 + t * x_1

    # 5. Model Input and Prediction
    input_cat = torch.cat([x_t, s_A], dim=1)
    u_pred = VFE_model(input_cat, t, c_V)

    # 6. Compute Target Vector Field: u_target = x_1 - x_0
    u_target = x_1 - x_0

    # 7. MSE flow-matching loss: L = ||u_pred - u_target||^2_2
    loss = criterion(u_pred, u_target)

    # 8. Update Weights
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

Algorithm 1. PyTorch-style pseudocode for AV-CASS training.

Fig. 8, each participant first watches the original video on the top of the scoring page, which helps participants to understand the overall audio scene and the different layers of sound present. Then, there is a text note "TARGET TRACK" which is the target sound type that the participant should focus on in the current evaluating sample. Finally, the separation results from the three CASS models are presented, and each separated audio sample is synchronized with the video and placed in a randomized order. Participants then evaluate each sample based on the quality of the separated track for the given target. This setup allows participants to assess how well each model separates the target stem in the context of the original audio-visual scene.

For AVDnR samples, as shown on the left side of Fig. 9, participants first listen to the mixture audio, the GT audio for the target track, and then evaluate the separation outputs from six different models. Note that we also put the GT sample into the six evaluation samples to get the score of the GT sample, which can serve as a perceptual upper bound when evaluating model performance. Therefore, as shown on the right side of Fig. 9, participants are asked to score a total of seven samples. Although the GT is provided before each evaluation, it is not identified during the scoring of the seven samples in random order to avoid bias in the evaluation.

D.3. Participants and Scoring

The real-world and AVDnR samples were evaluated by 27 and 15 participants, respectively. Based on their responses, average MOS scores with 95% confidence intervals are reported. The results for the real-world and AVDnR evaluations are shown in Tab. 1 and Tab. 4 of the main paper, where our method is preferred by listeners. The MOS for the ground-truth tracks of AVDnR was 4.5, reflecting the general scoring tendency of the participant group. This value serves as a perceptual upper bound for interpreting

```python
# --- Requirements ---
# VFE_model: Trained torch.nn.Module (u_theta)
# s_A: Mixture spectrogram
# v_f, v_s: Visual frames
# N: Sampling time steps (e.g., N=100)

@torch.no_grad()
def inference_av_cass(VFE_model, s_A, v_f, v_s, N):
    VFE_model.eval()

    # 1. Compute step size and condition vector
    eta = 1.0 / N
    c_V = visual_fusion(v_f, v_s)

    # 2. Initialize x_t (x_0) with Gaussian noise
    batch_size, channels_A, T, F = s_A.shape
    required_channels = 3 * channels_A # Assuming 3 sources: DX, FX, MX
    x_t = torch.randn(batch_size, required_channels, T, F, device=s_A.device)

    # 3. Euler Integration Loop (n = 1 to N)
    for n in range(1, N + 1):
        t = n / N # Normalized time step

        # Compute VFE direction: u = u_theta(CAT(x_t, s_A), t, c_V)
        input_cat = torch.cat([x_t, s_A], dim=1)
        u = VFE_model(input_cat, t_tensor, c_V)

        # Euler Step: x_{t+eta} = x_t + eta * u
        x_t = x_t + (eta * u)

    # 4. Return Separated Sources
    # The final x_t is the estimate of x_1 (CAT(s_DX, s_FX, s_MX))
    pred_s_DX, pred_s_FX, pred_s_MX = torch.chunk(x_t, chunks=3, dim=1)

    return pred_s_DX, pred_s_FX, pred_s_MX
```

Algorithm 2. PyTorch-style pseudocode for AV-CASS inference.

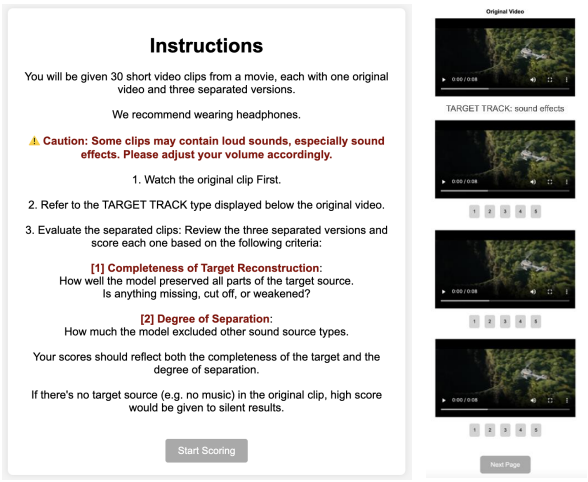

Figure 8. Instructions and scoring interface for the MOS test on real-world movie clips.

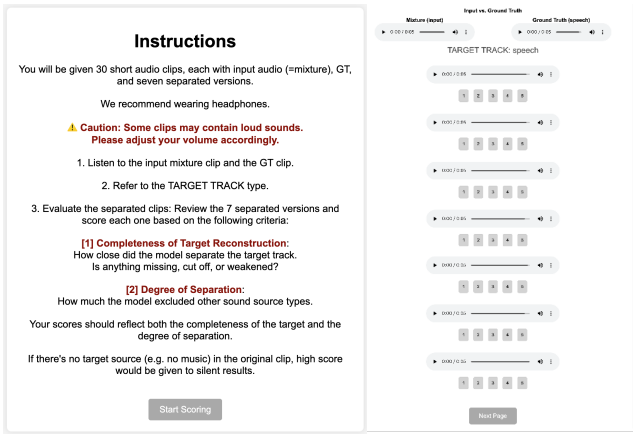

Figure 9. Instructions and scoring interface for the MOS test on AVDnR dataset.

the MOS of model outputs on the AVDnR dataset. There is no perceptual upper bound for the real-world experiment, as ground-truth references are not available for real-world samples.

E. Metrics

E.1. Standard Quality Metrics

We evaluate separation quality using standard objective and perceptual metrics, including FAD, KL divergence, SI-SDRi, and PESQ. These metrics capture signal fidelity, perceptual quality, and distributional similarity to reference audio. We evaluate FAD and KL divergence using the official evaluation code* provided by AudioLDM [40].

Fréchet Audio Distance (FAD). Fréchet Audio Distance (FAD) [32] is a metric to measure the distance between the generated outputs and the ground truth, similar to Fréchet Inception Distance (FID) [25] in the image domain. We use the VGGish model pretrained on YouTube-100M to extract embeddings from both the separated and ground truth audio clips of AVDnR. FAD is then computed as the Fréchet distance between the two sets of embeddings. A lower FAD score indicates that the generated audio is more plausible and perceptually closer to real-world clean recordings.

Kullback–Leibler (KL) divergence. Kullback-Leibler (KL) divergence measures how much the predicted distribution is different from a true distribution. The KL divergence computes a pairwise KL divergence between the extracted feature from the separated audio and ground truth audio, and reports the average across all the evaluation set. Specifically, we use the PANNs [34] model pretrained on AudioSet for large-scale audio classification to extract class probability distributions from both the generated and reference audio. A lower KL divergence suggests that the model output shares similar semantic content with the ground truth, implying better preservation of the original sound concepts.

Scale-Invariant Signal-to-Distortion Ratio improvement (SI-SDRi). SI-SDRi measures the quality improvement of a separated signal compared to the input signal while considering the scale-invariance of audio signals. A higher SI-SDRi indicates better separation performance, with 0 dB signifying no improvement over the input.

The scale-invariant signal-to-distortion ratio (SI-SDR) [36] is defined as:

$$\text{SI-SDR}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = 10\log_{10}\frac{\|\alpha\boldsymbol{x}\|^2 + \epsilon}{\|\alpha\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2 + \epsilon}, \tag{9}$$

where $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ denote the ground-truth and estimated sources, respectively. The optimal scaling factor α for the target signal is defined as:

$$\alpha = \frac{\boldsymbol{x}^\top \hat{\boldsymbol{x}} + \epsilon}{\|\boldsymbol{x}\|^2 + \epsilon}, \tag{10}$$

with $\epsilon = 9.76562 \times 10^{-4}$ to ensure numerical stability. Given the ground-truth audio signal $\boldsymbol{x}$, input mixture audio $\boldsymbol{y}$, and the estimated source $\hat{\boldsymbol{x}}$, SI-SDRi is calculated as the improvement over the mixture baseline:

$$\text{SI-SDRi} = \text{SI-SDR}(\boldsymbol{x}, \hat{\boldsymbol{x}}) - \text{SI-SDR}(\boldsymbol{x}, \boldsymbol{y}). \tag{11}$$

Perceptual Evaluation of Speech Quality (PESQ). We use PESQ [50] to evaluate the perceptual quality of the separated speech track, which is widely used in speech enhancement tasks. PESQ uses psychoacoustic modeling to measure perceived distortions between the ground truth and predicted signals. The score ranges from –0.5 to 4.5, where higher is better.

E.2. Wrong Placement Ratio (WPR)

In this paper, we introduce a new metric, Wrong Placement Ratio (WPR), to assess the presence of incorrectly placed sounds in each separated stem. This metric does not require ground-truth reference separated tracks. It uses a pretrained frame-wise sound event detection (SED) model, PANNs [34], to detect segments of residual or misplaced components of other stems in each target stem. The SED operates with a frame resolution of 10 ms and produces per-frame activation probabilities for 527 sound event categories. These fine-grained classes are manually grouped into three broad categories, *i.e.* speech, sound effects, and music, based on their semantic labels and auditory characteristics. The grouping taxonomy can be found on the attached "class_labels_with_main_class.csv".

For each separated track, we apply the SED model and convert the resulting probability matrix into a binary matrix $P \in \{0,1\}^{T\times C}$, where T is the number of frames and $C = 3$ is the number of merged classes. An element $P_{t,c} = 1$ indicates that class c is predicted to be active at frame t. A fixed threshold of 0.25 is applied to the SED probabilities to obtain the binary decisions.

We define WPR for each track to quantify the proportion of frames contaminated by non-target sound classes, *i.e.*, for the separated speech track, we measure how many sound effects and music segments were wrongly placed there. To ensure the metric captures substantive misplacements and excludes transient artifacts, we only count non-target activations that persist for a minimum length threshold τ. This threshold is empirically set to 50 frames (0.5 seconds). Let target_i denote the expected class label for separated track i, where $i \in 1, 2, 3$ corresponds to speech, sound effects, and music, respectively. We define a thresholded binary prediction matrix $\hat{P}$ where $\hat{P}_{t,c} = 1$ only if the prediction for class c at time t belongs to a contiguous block of at least τ activated frames. We calculate the WPR based on non-silent frames, which are frames with at least one class activated

*https://github.com/haoheliu/audioldm_eval

Method	A-V	P/G	FAD (↓)				KL (↓)				SI-SDRi [dB] (↑)				WPR [%] (↓)				PESQ (↑)
			Avg.	DX	FX	MX	Avg.	DX	FX	MX	Avg.	DX	FX	MX	Avg.	DX	FX	MX	DX
Hybrid Demucs [14]	✗	P	2.05	1.66	1.34	3.16	1.03	0.99	0.95	1.14	13.57	12.75	13.59	14.38	5.24	0.12	12.67	2.93	2.16
HT Demucs [52]	✗	P	2.08	1.97	1.62	2.65	1.06	1.09	**0.92**	1.17	13.41	12.68	13.33	14.21	9.23	0.15	19.97	7.57	2.06
MRX [48]	✗	P	3.47	2.67	2.02	4.01	1.67	2.00	1.02	1.97	10.60	11.23	11.34	12.31	14.91	2.18	37.60	4.94	1.89
BandIt [59]	✗	P	2.15	2.38	2.96	1.11	1.14	0.82	1.19	1.41	**14.40**	**13.33**	**14.39**	**15.48**	4.65	0.04	12.46	1.46	2.15
MSDM [44]	✗	P	2.90	3.84	3.03	3.56	1.70	2.47	0.97	1.66	11.63	11.48	9.38	10.94	5.65	1.65	11.49	3.82	2.12
DAVIS-Flow [29]	✓	G	5.94	6.72	3.75	7.35	1.64	1.50	1.48	1.94	9.25	9.44	10.96	7.35	12.14	0.22	**3.66**	32.55	1.94
AV-CASS (Ours)	✓	G	**0.85**	**0.67**	**0.89**	**0.98**	**0.93**	**0.64**	1.05	**1.09**	12.32	10.93	12.33	13.71	**1.84**	**0.03**	4.29	**1.21**	**2.26**
Ours (Audio-only)	✗	G	1.63	2.13	1.61	1.15	1.15	0.71	1.37	1.38	12.23	10.94	12.24	13.52	2.01	0.03	4.65	1.34	2.08

Table 8. *Detailed objective metric results on AVDnR for each stem.* All models are trained on our training data. A-V refers to audio-visual, P refers to predictive model, and G refers to generative model.

in the original prediction matrix P. The WPR for separated track i is computed as follows:

$$\text{WPR}_i = \frac{1}{T_i} \sum_{t=1}^{T_i} 1 \left[\sum_{c \neq \text{target}_i} \hat{P}_{t,c} > 0 \right] \tag{12}$$

where T_i is the sum of non-silent frames in track i, and the indicator function returns 1 if any non-target class is active in frame t. We exclude silent frames from this calculation because they trivially yield a WPR of zero, which does not reflect correct separation behavior when the target class is missing entirely. This ensures that the metric captures actual contamination in tracks with meaningful content.

F. Per-Track Objective Metrics on AVDnR

Tab. 8 presents detailed metric scores for each stem on the AVDnR dataset, along with the average values across all stems. Our model consistently outperforms other methods across all stems in terms of FAD and achieves the highest PESQ score for the DX stem. It also achieves the best KL divergence scores for the DX and MX stems and performs comparably to the top method on the FX stem. While our model shows lower performance on the SI-SDRi metric, this is expected given the nature of generative models, which often prioritize perceptual quality over waveform-level fidelity, as discussed in the main paper.

For the per-track WPR shown in Tab. 8, AV-CASS achieves the lowest average leakage (1.84%) and sets new best results on DX (0.03%) and MX (1.21%). The only stem where AV-CASS is not the top performer is FX, where DAVIS-Flow [29] achieves the lowest WPR (3.66%), reflecting its strong reliance on visual object cues for transient effects. However, DAVIS-Flow performs significantly worse on DX and MX, with large leakages of 0.22% (DX) and 32.55% (MX), highlighting its difficulty in handling dialogue and music where audio-visual associations are weaker or more complex. In contrast, AV-CASS delivers consistently low leakage across all stems, demonstrating robust and well-balanced separation quality.

G. Per-Track WPR Results on DnRv2 and DnRv3

Method	DnRv2 [48]			DnRv3 [60]		
	DX	FX	MX	DX	FX	MX
Hybrid Demucs [14]	0.26	10.94	3.13	0.19	7.75	3.39
HT Demucs [52]	0.26	21.45	5.71	0.20	19.67	5.61
MSDM [44]	2.42	10.20	3.38	0.18	8.86	4.41
MRX [48]	2.00	41.50	6.19	3.25	39.73	6.94
BandIt [59]	0.24	7.36	3.40	**0.09**	6.56	3.95
Ours (Audio-Only)	**0.12**	**3.78**	**2.48**	0.14	**2.46**	**3.13**

Table 9. *Wrong Placement Ratio (WPR [%]).* Ratio of residual sounds from other tracks within each track type for the DnRv2 and DnRv3 test sets. Lower is better.

We presented per-track WPR results on real-world samples and the AVDnR dataset in the main paper. As part of the main paper, we also showed quantitative results on the DnRv2 and DnRv3 datasets using audio-only CASS methods in Tab. 5 of the main paper, but only with average WPR scores. Here in the appendix, we provide the per-track WPR results for DnRv2 and DnRv3 in Tab. 9. Our model consistently outperforms others, showing particularly large improvements on the FX stem. While it slightly underperforms on the DX track of DnRv3, the performance gap is minimal (0.05%). These results demonstrate our model's flexibility and generalizability in audio-only setup, achieving accurate separation with minimal cross-track confusion.

H. Qualitative Results on AVDnR

For qualitative assessment, we present spectrograms from our model and existing CASS methods in Fig. 10. Unlike previous audio-only approaches, our model consistently isolates sounds with minimal misclassification. Notably, in the orange-boxed area, AV-CASS accurately reconstructs speech even under challenging conditions with significant overlap from music and sound effects, leveraging facial cues for improved separation. Additionally, the vacuum cleaning sound, indicated by the pink arrow on the FX track and associated with a visible vacuum cleaner in $\boldsymbol{v}^s$, is correctly separated as FX by our model. In contrast, other methods incorrectly place it on music or speech tracks.

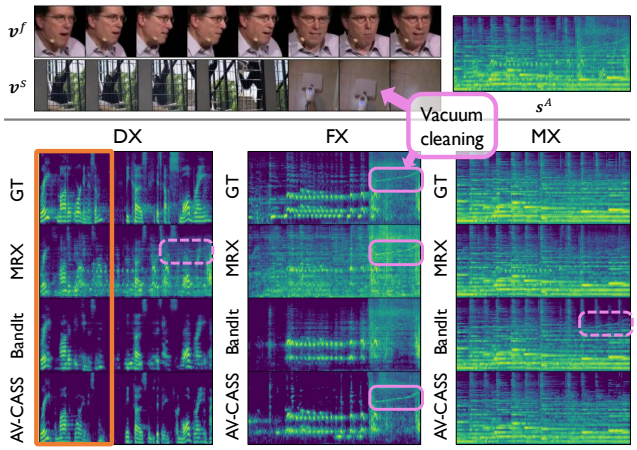

Figure 10. *Comparison of spectrograms from MRX, BandIt, and AV-CASS (ours) on AVDnR.* Facial frames $\boldsymbol{v}^f$, scene frames $\boldsymbol{v}^s$, and the input audio spectrogram $\boldsymbol{s}^A$ shown at the top. Ground truth audio spectrogram is on top of the models' results for each stem. The orange box in the DX stem highlights that our model reconstructs speech most accurately compared to others. Pink solid boxes indicate the presence of vacuum cleaning in $\boldsymbol{v}^s$ and the spectrograms of the FX stem, while dotted boxes indicate incorrectly placed vacuum cleaning sounds.

Furthermore, we present additional comparisons on the AVDnR dataset in Fig. 11 through Fig. 15. These figures compare our audio-visual model with existing music source separation and CASS models, including Hybrid Demucs (HDemucs) [14], HT Demucs [52], MSDM [44], MRX [48], and BandIt [59]. The visualizations highlight the effectiveness of our approach in accurately separating audio components into each corresponding track.

Similar to Sec. 5.2.2 in the main paper, we include additional results comparing our audio-only and audio-visual models in Fig. 16 to Fig. 17, demonstrating the effectiveness of visual cues in improving separation quality.

I. Failure Case Analysis

Among the real-world sample results, we observed several failure cases for our model.

Difficulty with non-verbal vocalizations. First, our model showed difficulty in handling non-verbal vocalizations, such as screaming and laughter. Since both our training data and previous DnR datasets included a limited amount of data for screaming or laughter within the DX stem, the model struggles to consistently separate these sounds into the designated speech track. This issue arises from the ambiguity of the class boundary and the lack of such data in the training set. Given the data-centric nature of the problem, other existing CASS models also share this limitation. This specific issue has recently been addressed in audio-only CASS by introducing a new, dedicated dataset [23]. We plan to resolve this on the audio-visual dataset side, or by pretraining the model on these audio-only datasets to provide the necessary guidance.

Generative artifacts from loud sound effects. Second, when encountering a loud sound effect that occupies a wide range of frequency bands (*e.g.* a loud engine sound), our model sometimes generates speech-like artifacts even in the absence of actual speech. This is likely caused by the inherent characteristics of generative models which may overgeneralize complex noise patterns as vocal cues. In future work, we will investigate methods to explicitly penalize such generative artifacts during training and inference.

Confusion between music and synchronized SFXs. The final failure case observed occurs specifically in contexts like musical films or movie trailers. When the music's rhythm and the SFX timing are edited in perfect synchronization, the model shows a tendency to confuse the music and SFX. It frequently leads to imperfect separation, where the output streams are not cleanly isolated and contain components of the other sound source.

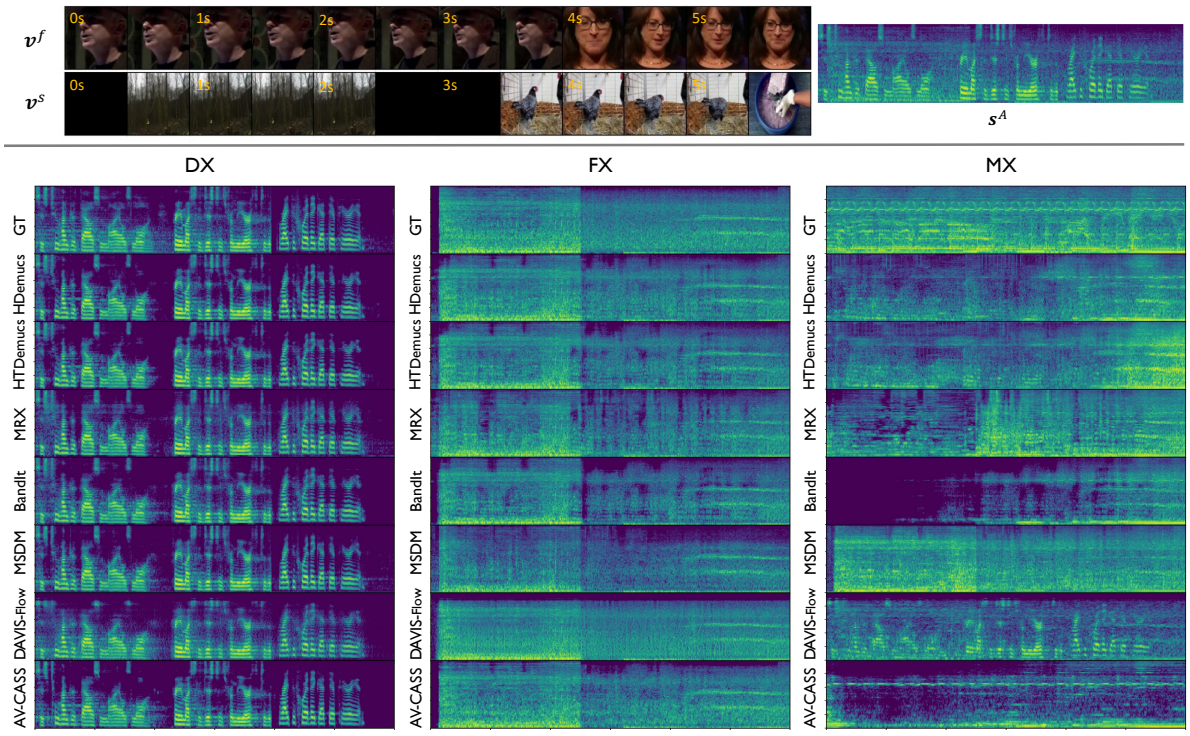

Figure 11. Comparison of the separation results of our model and other methods on AVDnR dataset.

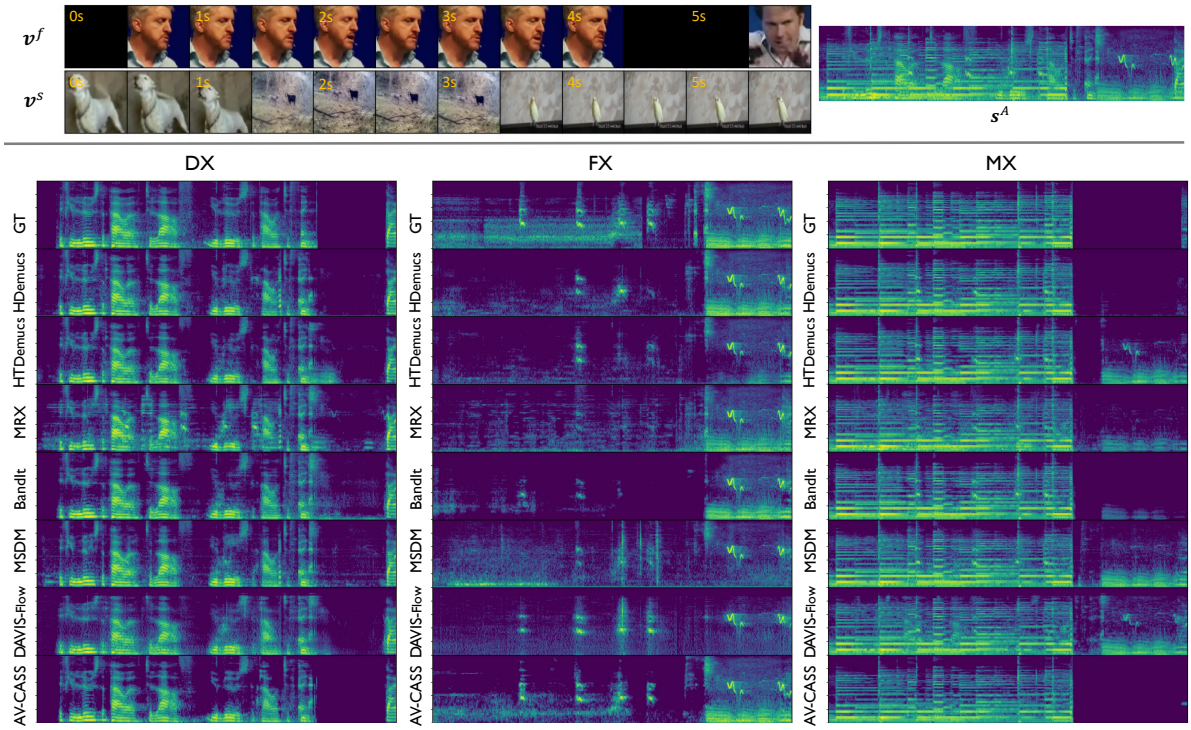

Figure 12. Comparison of the separation results of our model and other methods on AVDnR dataset.

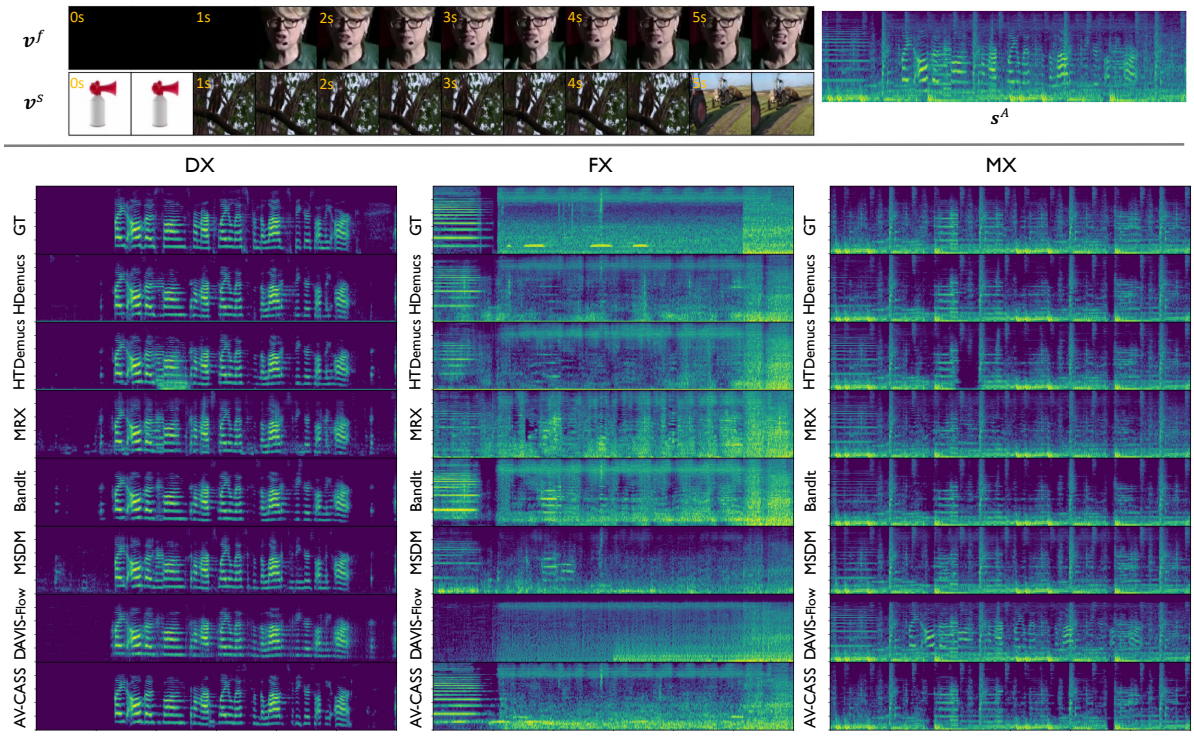

Figure 13. Comparison of the separation results of our model and other methods on AVDnR dataset.

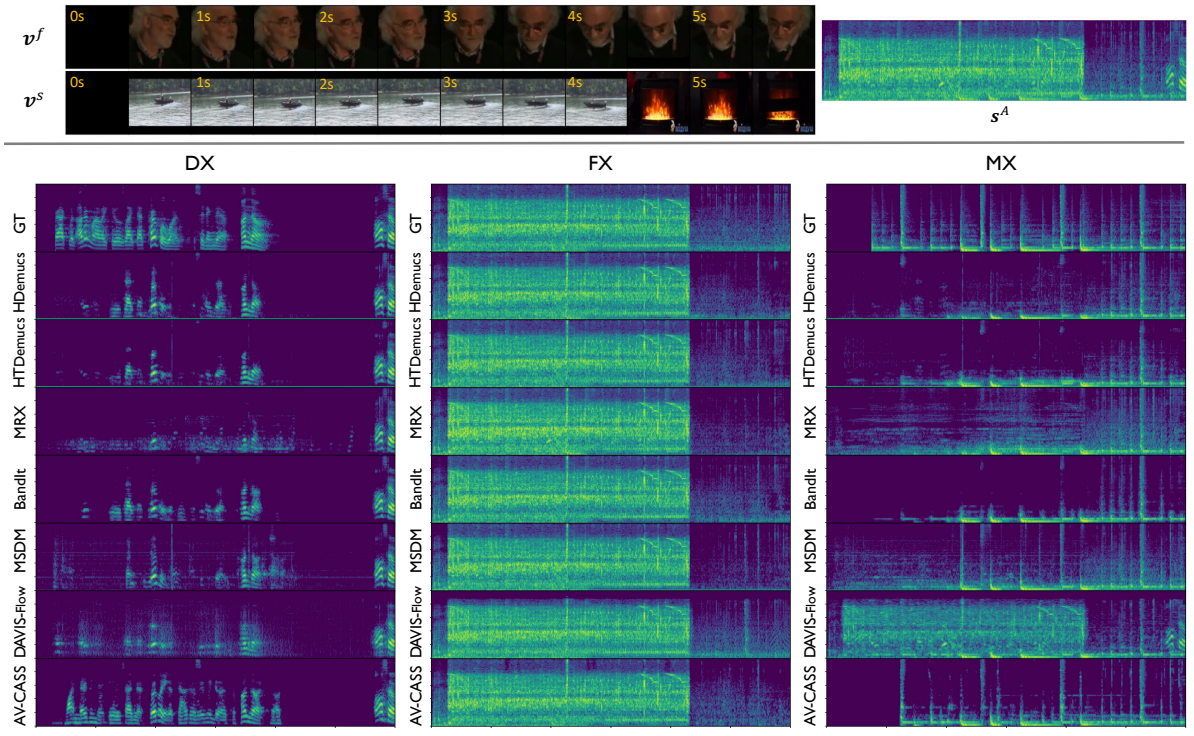

Figure 14. Comparison of the separation results of our model and other methods on AVDnR dataset.

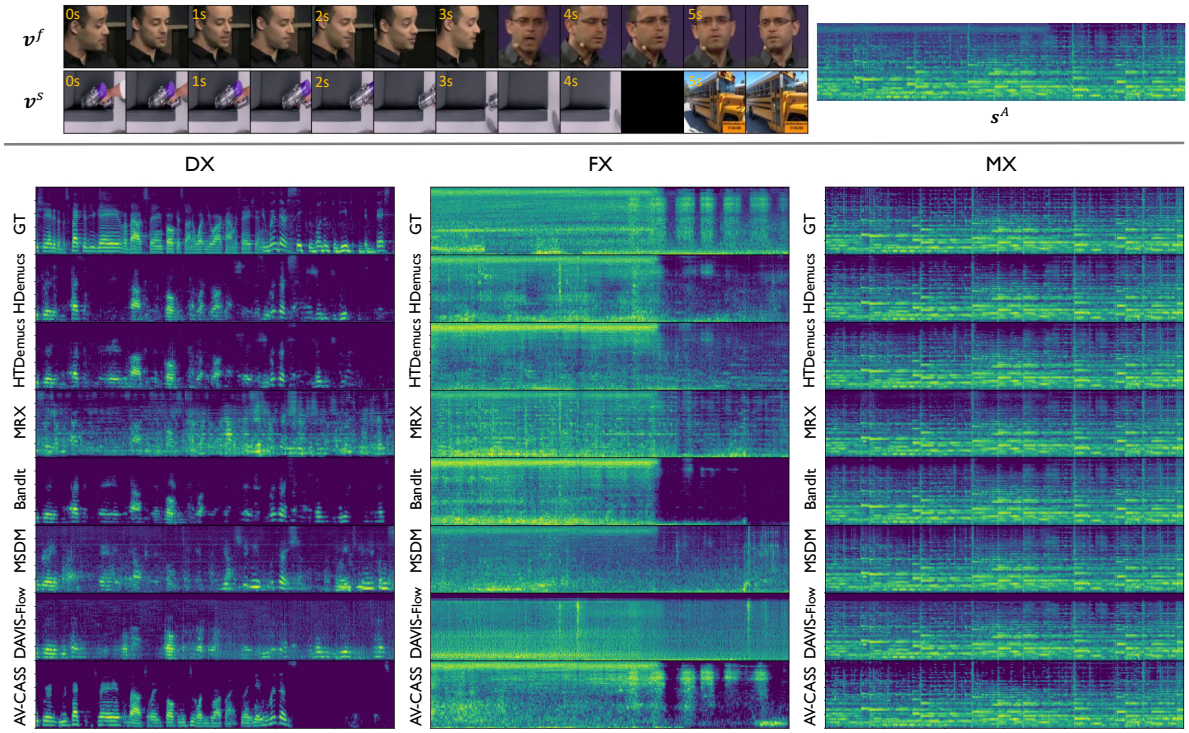

Figure 15. Comparison of the separation results of our model and other methods on AVDnR dataset.

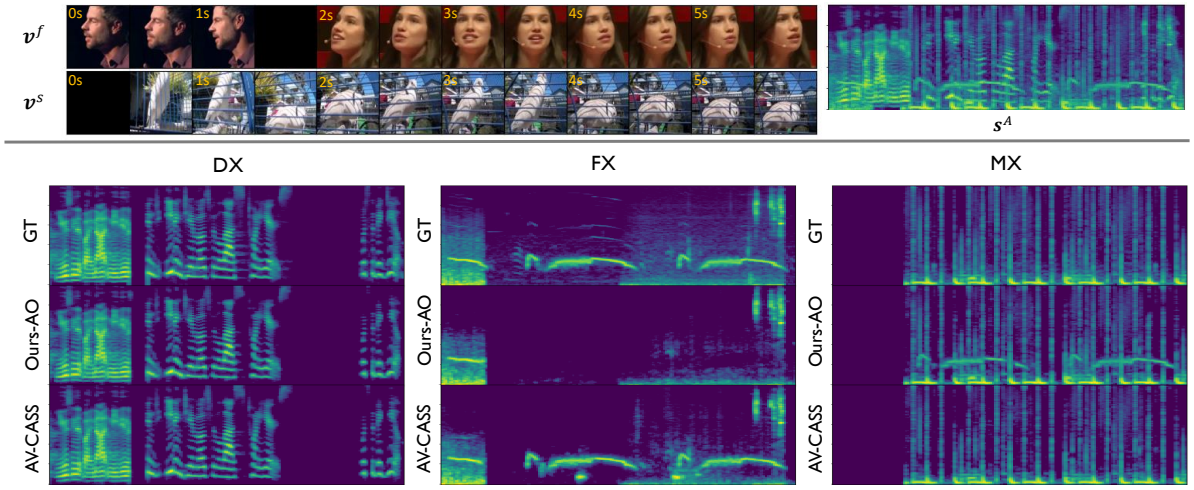

Figure 16. Spectrogram comparison of separated outputs from our audio-only model (Ours-AO), audio-visual model (AV-CASS), and ground truth (GT) on AVDnR.

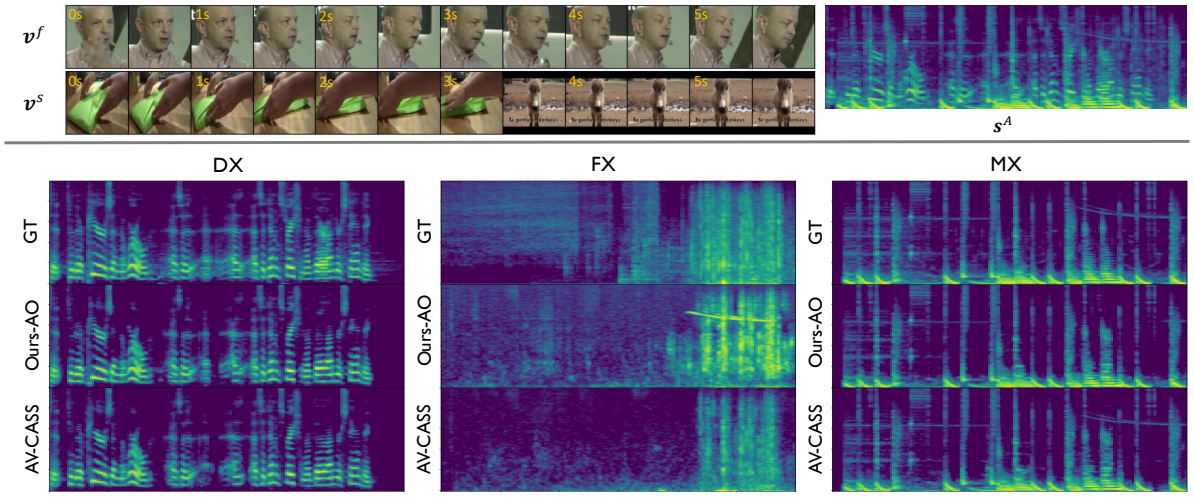

Figure 17. Spectrogram comparison of separated outputs from our audio-only model (Ours-AO), audio-visual model (AV-CASS), and ground truth (GT) on AVDnR.