MMAudio: Taming Multimodal Joint Training
for High-Quality Video-to-Audio Synthesis

Ho Kei Cheng[1*]   Masato Ishii[2]   Akio Hayakawa[2]   Takashi Shibuya[2]
Alexander Schwing[1]   Yuki Mitsufuji[2,3]
[1]University of Illinois Urbana-Champaign   [2]Sony AI   [3]Sony Group Corporation

{hokeikc2,aschwing}@illinois.edu, {masato.a.ishii,akio.hayakawa,takashi.tak.shibuya,yuhki.mitsufuji}@sony.com

Abstract

*We propose to synthesize high-quality and synchronized audio, given video and optional text conditions, using a novel multimodal joint training framework (**MMAudio**). In contrast to single-modality training conditioned on (limited) video data only, MMAudio is jointly trained with larger-scale, readily available text-audio data to learn to generate semantically aligned high-quality audio samples. Additionally, we improve audio-visual synchrony with a conditional synchronization module that aligns video conditions with audio latents at the frame level. Trained with a flow matching objective, MMAudio achieves new video-to-audio state-of-the-art among public models in terms of audio quality, semantic alignment, and audio-visual synchronization, while having a low inference time (1.23s to generate an 8s clip) and just 157M parameters. MMAudio also achieves surprisingly competitive performance in text-to-audio generation, showing that joint training does not hinder single-modality performance. Code, models, and demo are available at: hkchengrex.github.io/MMAudio.*

1. Introduction

We are interested in *Foley*, *i.e.*, for a given video we want to synthesize ambient sound (*e.g.*, rain, river flow) and sound effects induced by visible events (*e.g.*, a dog barks, a racket hits a tennis ball). Note, Foley does not focus on synthesizing background music or human speech, which is often added in post-processing. Importantly, Foley requires to synthesize convincing high-quality audio that is 1) semantically and 2) temporally aligned to an input video. For semantic alignment, methods need to understand scene contexts and their association with audio – the visual concept of rain should be associated with the sound of splashing raindrops. For temporal alignment, methods need to understand audio-visual synchrony as humans can perceive audio-visual misalignment as
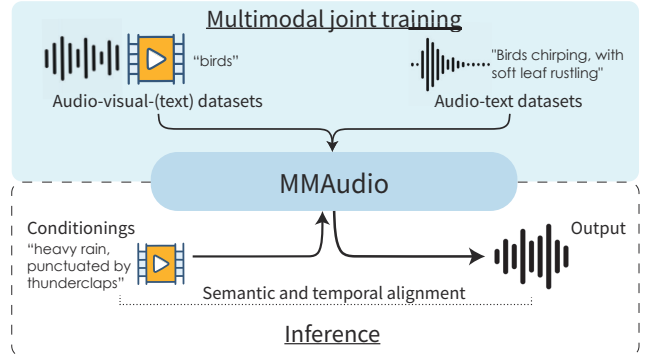
Figure 1. In addition to training on audio-visual-(text) datasets, we perform multimodal joint training with high-quality, abundant audio-text data which enables effective data scaling. At inference, MMAudio generates conditions-aligned audio with video and/or text guidance.

slight as 25 ms [51]. Inspired by the efficacy of training data scaling demonstrated by recent works [8, 21, 53], we pursue a data-driven approach to synthesize high-quality audio that respects these two types of alignment.

Current state-of-the-art video-to-audio methods either train only on audio-visual data [1] from scratch [67] or train new "control modules" [20, 66, 69, 73] for pretrained text-to-audio models on audio-visual data. The former is limited by the amount of available training data: the most commonly used audio-visual dataset VGGSound [1] contains only around 550 hours of videos. Audio-visual data are expensive to curate at a large scale, as in-the-wild videos from the Internet 1) contain music and speech[1], which has limited utility for training a general Foley model, and 2) contain non-diegetic [58] sounds such as background music or sound effects added in post-processing, again unsuitable for a Foley model. The latter, *i.e.*, finetuning pretrained text-to-audio models (with added control modules) on audio-visual data, enables models to benefit from audio-generation knowledge obtained from larger-scale audio-text data (*e.g.*,

---

[1]In AudioSet [9] (collected from YouTube), 82.48% of videos contain either human speech or music according to the provided audio labels.

WavCaps [46] at 7,600 hours). However, adding control modules to pretrained text-to-audio models complicates the network architecture and limits the design space. It is also unclear whether pretrained text-to-audio models have sufficient degrees of freedom to support all video-to-audio scenarios, compared to training from scratch.

To avoid these limitations, we propose a *multimodal joint training* paradigm (Figure 1) that jointly considers video, audio, *and* text in a single multimodal transformer network and masks missing modalities during training. This allows us to train from scratch on both audio-visual and audio-text datasets with a simple end-to-end framework. Jointly training on large multimodal datasets enables a unified semantic space and exposes the model to more data for learning the distribution of natural audio. Empirically, with joint training, we observe a significant relative improvement in audio quality (10% lower Fréchet Distance [23] and 15% higher Inception Score [56]), semantic alignment (4% higher ImageBind [11] score), and temporal alignment (14% better synchronization score).

To further improve temporal alignment, we introduce a conditional synchronization module that uses high frame-rate visual features (extracted from a self-supervised audio-visual desynchronization detector [19]) and operates in the space of scales and biases of adaptive layer normalization (adaLN) layers [50], leading to accurate synchronization (50% relative improvement in synchronization score).

In summary, we first propose MMAudio, a **multimodal joint training** paradigm for video-to-audio. It enables accessible data scaling and cross-modal understanding, significantly improving audio quality and semantic alignment. We also propose a **conditional synchronization module** which enables more precise audio-visual synchrony. We train MMAudio on openly accessible datasets and scale it to two audio sampling rates (16kHz and 44.1kHz) and three model sizes (157M, 621M, 1.03B), with the smallest model already achieving new state-of-the-art performance in video-to-audio synthesis among public models. Surprisingly, our multimodal approach also achieves competitive performance in text-to-audio generation compared to dedicated text-to-audio methods, showing that joint training does not hinder single-modality performance.

2. Related Works

Semantic alignment. Semantic alignment between audio and video is learned by training on paired audio-visual data with a generation objective [2, 6, 18, 44, 49, 60, 66, 71] or a contrastive objective [42]. To further understand audio semantics, we additionally train on paired audio-text data. We argue that the semantic understanding learned from audio-text pairs can be transferred to video-text pairs, as joint training leads to a shared semantic space (similar to ImageBind [11] and LanguageBind [74]) and enables the

network to learn richer semantics from more diverse data.

Temporal alignment. Besides learning temporal alignment directly from audio-visual pairs, some recent works first learn from videos a separate model to predict handcrafted proxy features such as audio onsets [54, 73], energy [17, 20], or root-mean-square of waveforms [4, 34]. We deviate from these handcrafted features and directly learn alignment from the deep feature embeddings of a pretrained self-supervised desynchronization detector Synchformer [19], allowing a more nuanced interpretation of the input signal. A recent work V-AURA [65] also uses Synchformer [19] for synchronization in an autoregressive framework. However, [65] does not perform multimodal training on text and has a short context window (2.56s) while we produce longer-term (8-10s) temporally consistent generations. In terms of improving positional embeddings for temporal alignment, Mei et al. [45] concurrently propose to subsample high-frequency (audio) positional embeddings, while we propose to scale up the frequencies of low-frequency (visual) positional embeddings – the effects are identical when the higher frequency is an integer multiple of the lower one.

Multimodal conditioning. The most common way to support multimodal conditioned generation is to add "control modules" that inject visual features to a pretrained text-to-audio network [13, 17, 20, 34, 47, 54, 73]. However, this increases the number of parameters. Besides, as the text modality is fixed during video-to-audio training, it becomes more challenging to learn a joint semantic space – the video modality needs to bind to the semantics of text instead of both modalities learning to cooperate. In contrast, we train all modalities simultaneously in our multimodal training formulation to learn joint semantics and enable omnidirectional feature sharing among modalities. Alternatively, to align different modalities without training, Seeing-and-Hearing [69] uses a pretrained text-to-audio model and performs gradient ascent on an alignment score (*i.e.*, ImageBind [11]) at test-time. We note that the test time optimization is slow and sometimes results in low-quality and temporally misaligned output. Indeed, our model is faster at test time and more consistently produces synchronized audio. Concurrent to our work, VATT [40] and MultiFoley [3] explore jointly trained multimodal conditioning. VATT [40] uses both video and text to generate audio but always requires video conditions during training. MultiFoley [3] is formulated similarly to ours, but MMAudio uses much higher frame-rate visual features (24 FPS, while MultiFoley uses 8 FPS features), which leads to significantly better audio-visual synchronization.

Multimodal generation. Related to multimodal conditioning, multimodal generation models produce samples composed of multiple modalities (*e.g.*, video *and* audio). Multimodal generation is more challenging and existing approaches [25, 55, 61, 62] are not yet competitive with dedicated video-to-audio models. In this work, we focus on

multimodal conditioned audio generation. We believe our multimodal formulation and architecture serve as a foundation for future work in multimodal generation.

3. MMAudio

3.1. Preliminaries

Conditional flow matching. We use the conditional flow matching objective [37, 63] for generative modeling and refer readers to [63] for details. In short, at test time, to generate a sample, we randomly draw noise x_0 from the standard normal distribution and use an ODE solver to numerically integrate from time $t = 0$ to time $t = 1$ following a learned time-dependent conditional velocity vector field $v_\theta(t, \mathbf{C}, x) : [0, 1] \times \mathbb{R}^C \times \mathbb{R}^d \to \mathbb{R}^d$, where t is the timestep, $\mathbf{C}$ is the condition (*e.g.*, video and text), and x is a point in the vector field. We represent the velocity vector field via a deep net parameterized by θ.

At training time, we find θ by considering the conditional flow matching objective

$$\mathbb{E}_{t,q(x_0),q(x_1,\mathbf{C})} \| v_\theta(t, \mathbf{C}, x_t) - u(x_t | x_0, x_1) \|^2, \quad (1)$$

where $t \in [0, 1]$, $q(x_0)$ is the standard normal distribution, and $q(x_1, \mathbf{C})$ samples from the training data. Further,

$$x_t = tx_1 + (1 - t)x_0 \quad (2)$$

defines a linear interpolation path between noise and data, and

$$u(x_t | x_0, x_1) = x_1 - x_0 \quad (3)$$

denotes its corresponding flow velocity at x_t.

Audio encoding. For computational efficiency, we model the generative process in a latent space, following the common practice [38, 67]. For this, we first transform audio waveforms via Short-Time Fourier Transform (STFT) and extract mel spectrograms [57], which are then encoded by a pretrained variational autoencoder (VAE) [27] into latents x_1. During testing, the generated latents are decoded by the VAE into spectrograms which are then vocoded by a pretrained vocoder [35] into audio waveforms.

3.2. Overview

Following conditional flow matching, at test time, we numerically integrate noise x_0 from $t = 0$ to $t = 1$ following a flow v_θ, which was learned at training time by optimizing Eq. (1). Numerical integration at test time arrives at a latent x_1 that is decoded to audio preferably of high-quality and preferably semantically and temporally aligned to the video and text conditions.

To estimate a flow v_θ for the current latent x, MMAudio operates on the video/text conditions and the flow timestep t. Figure 2 illustrates our network architecture. To combine inputs from different modalities, MMAudio consists of a

series of (N_1) **multimodal transformer** blocks [8] with visual/text/audio branches, followed by a series of (N_2) audio-only transformer blocks [32]. Additionally, for audio-visual synchrony, we devise a **conditional synchronization module** that extracts and integrates into the generation process high frame rate (24 frames per second (fps)) visual features for temporal alignment. Next, we describe both components in detail.

3.3. Multimodal Transformer

Core to our approach is the desire to model the interactions between video, audio, and text modalities. For this purpose, we largely adopt the MM-DiT block design from SD3 [8] and introduce two new components for temporal alignment: aligned RoPE positional embeddings for aligning sequences of different frame rates and 1D convolutional MLPs (ConvMLPs) for capturing local temporal structure. Figure 2 (right) illustrates our block design. Note, we also include a sequence of audio-only single-modality blocks following FLUX [32], implemented by simply removing the streams of the two other modalities (*i.e.*, the joint attention becomes a self-attention). Compared to considering all modalities at every layer, this design allows us to build a deeper network with the same parameter count and compute without sacrificing multimodality. This multimodal architecture allows the model to selectively attend to and focus on different modalities depending on the inputs, thus enabling effective joint training on both audio-visual and audio-text data. Next, we describe the feature representation in our network and then the core components of our block design.

Representations. We represent all the features as one-dimensional tokens. Note, we deliberately do not use any absolute position encoding which allows us to generalize to different durations at test time. Thus, we specify the temporal sequences in terms of frame rates to determine the number of tokens for a given duration. The visual features F_v (one token per frame, at 8 fps) and text features F_t (77 tokens) are extracted from CLIP [53] as 1024d features. The audio latents x are in the VAE latent space (Section 3.1), at 31.25 fps as 20d latents by default. The synchronization features F_{syn} are extracted with Synchformer [19] at 24 fps as 768d features, which we will detail in Section 3.4. Note, except for the text tokens, all other tokens follow the same temporal ordering, albeit at different frame rates. After the initial ConvMLP/MLP layers, all features will be projected to the hidden dimension h.

Joint attention. These tokens from different modalities communicate via joint attention (Figure 2, right). Following [8], we concatenate the query, key, and value representations from the three different modalities and apply scaled dot product attention [64]. The output is split into three modalities, following the input partition. We refer readers to [8] for details. We note that joint attention alone does not
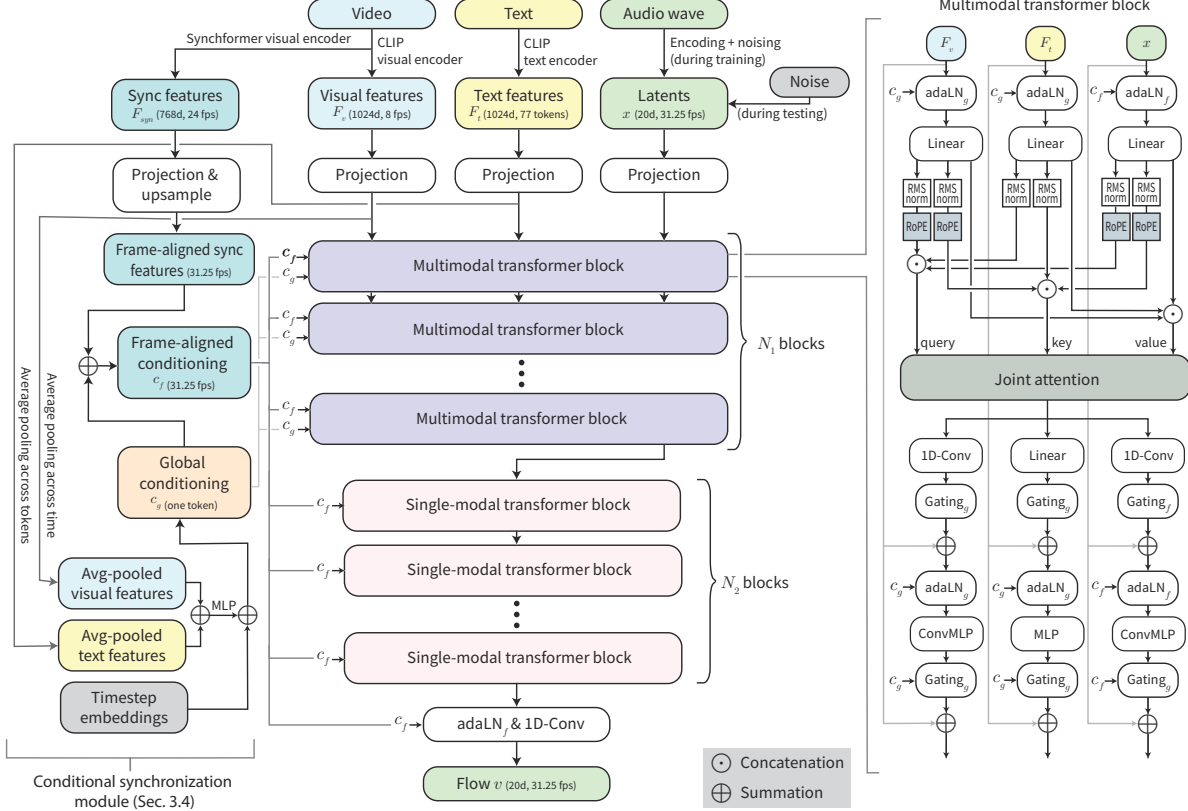
Figure 2. Overview of the MMAudio flow-prediction network. Video conditions, text conditions, and audio latents jointly interact in the multimodal transformer network. A synchronization model (Section 3.4) injects frame-aligned synchronization features for precise audio-visual synchrony.

capture temporal alignment which we will address next.

Aligned RoPE position embedding. For audio-visual synchrony, precise temporal alignment is crucial. As typical in transformers [64], we adopt positional embeddings to inform the attention layers of time. Specifically, we apply RoPE [59] embeddings on the queries and keys in the visual and audio streams before joint attention (Figure 2). Note that we do not apply it to the text stream since it does not follow the temporal order of video or audio. Further, since the frame rates do not align (8 fps for the visual stream, 31.25 fps for the audio stream), we scale the frequencies of the positional embeddings in the visual stream proportionally, *i.e.*, by $31.25/8$. We visualize the default (non-aligned) RoPE and our proposed aligned RoPE in Figure A5. We note that these aligned embeddings are beneficial yet insufficient for good synchrony. Therefore, we introduce an additional synchronization module, which we discuss in Section 3.4.

ConvMLP. To better capture local temporal structure, we use ConvMLPs rather than MLPs in the visual and audio streams. Concretely, our ConvMLP uses 1D convolutions (kernel size $= 3$ and padding $= 1$) rather than linear layers. Again, this change is not made to the text stream since it does not follow the temporal order of video or audio.

Global conditioning. Global conditioning injects global features into the network through scales and biases in adaptive layer normalization layers (adaLN) [50]. First, we compute a global conditioning vector $c_g \in \mathbb{R}^{1 \times h}$ shared across all transformer blocks from the Fourier encoding [64] of the flow timestep, the average-pooled visual features, and the average-pooled text features (Figure 2). Then, each adaLN layer modulates its input $y \in \mathbb{R}^{L \times h}$ (L is the sequence length) with the global condition c_g as follows:

$$\text{adaLN}_g(y, c_g) = \text{LayerNorm}(y) \cdot \mathbf{1}\mathbf{W}_\gamma(c_g) + \mathbf{1}\mathbf{W}_\beta(c_g). \tag{4}$$

Here, $\mathbf{W}_\gamma, \mathbf{W}_\beta$ are MLPs, and $\mathbf{1}$ is a $L \times 1$ all-ones matrix, which "broadcasts" the scales and biases to match the sequence length L – such that the same condition is applied to all tokens in the sequence (hence global). Next, we describe how we design position-dependent conditions for precise audio-visual synchronization.

3.4. Conditional Synchronization Module

We develop a token-level conditioning to further improve audio-visual synchrony. While the visual and audio streams already communicate via cross-modality attention layers, these layers aggregate features via a soft distribution, which we found to hamper precision. To address this issue, we first

4

extract high frame rate (24 fps) features (F_{syn}) from the input video using the visual encoder of Synchformer [19]. We use Synchformer because it is trained in a self-supervised manner to detect temporal misalignment between video and audio data, which we hypothesize will yield visual features relevant to audio events and hence benefit synchronization. We find the frame-aligned conditioning $c_f \in \mathbb{R}^{L \times h}$ via

$$c_f = \text{Upsample}\left(\text{ConvMLP}\left(F_{syn}\right)\right) + \mathbf{1}c_g. \quad (5)$$

Upsample uses nearest neighbor interpolation and matches the frame rate of the synchronization features F_{syn} with that of the audio latent x. This frame-aligned conditioning c_f is injected via the adaLN layers in the audio stream for feature modulation. Similar to Eq. (4), we apply c_f via

$$\text{adaLN}_f(x, c_f) = \text{LayerNorm}(x) \cdot \mathbf{A}_\gamma(c_f) + \mathbf{A}_\beta(c_f), \quad (6)$$

where $\mathbf{A}_\gamma, \mathbf{A}_\beta \in \mathbb{R}^{h \times h}$ are MLPs. Different from Eq. (4), the scales and biases are applied per token without broadcasting, providing fine-grained control.

3.5. Training and Inference

3.5.1. Multimodal Datasets

VGGSound. We train on VGGSound [1] as the only audio-text-visual dataset. It offers around 500 hours of footage. Additionally, VGGSound contains a class label (310 classes in total) for each video and we use the class names as input following ReWaS [20] and VATT [40]. We set aside 2K videos from the training set for validation, resulting in a training set of around 180K 10s videos. We use the first 8s of each video for training.

Audio-text datasets. We use AudioCaps [24] ($\sim$128 hours, manually captioned), Clotho [5] ($\sim$31 hours, manually captioned), and WavCaps [46] ($\sim$7,600 hours, automatically captioned from metadata) as audio-text datasets for training. Since they do not contain the visual modality, we set all the visual features and synchronization features corresponding to these samples as learnable empty tokens $\varnothing_v$ and $\varnothing_{syn}$ respectively. For short audios (<16s), we truncate them to 8s for training, as in VGGSound. For longer audios, we take up to five non-overlapping crops of 8s each. This results in a total of 951K audio clip-text pairs.

Overlaps. We notice a minor (<1% of the test sets) train/test data contamination among these datasets. For a fair comparison, we have removed the test sets of VGGSound and AudioCaps from all training data. We provide more details in Appendix E.

3.5.2. Implementation Details

Model variants. Our default model generates 16kHz audio encoded as 20-dimensional, 31.25fps latents (following Frieren [67]), with $N_1 = 4, N_2 = 8, h = 448$. We refer to this default model as 'S-16kHz'. We additionally train larger models and models with higher audio sampling rates: 'S-44.1kHz', 'M-44.1kHz', and 'L-44.1kHz', detailed in Appendix G.1. The parameter counts and running time of these models are summarized in Table 1. We describe additional implementation details in Appendices F to H.

Classifier-free guidance. To enable classifier-free guidance [14] during inference, we randomly mask away the visual tokens (F_v and F_{syn}) or the text with a 10% probability during training. The masked visual tokens are replaced with learnable tokens ($\varnothing_v$ and $\varnothing_{syn}$), while any masked text is replaced with the empty string $\varnothing_t$.

Inference. By default, we use Euler's method for numerical integration with 25 steps, with a classifier-free guidance strength of 4.5. Both video and text conditions are optional during test-time – we replace the missing modalities with empty tokens $\varnothing_v$, $\varnothing_{syn}$, or $\varnothing_t$. Recall, we deliberately do not use any absolute position encoding and thus can generalize to different durations at test time (*e.g.*, 8s in VGGSound and 10s in AudioCaps in Section 4.2).

4. Experiments

4.1. Metrics

We assess the generation quality in four different dimensions: distribution matching, audio quality, semantic alignment, and temporal alignment.

Distribution matching assesses the similarity in feature distribution between ground-truth audio and generated audio, under some embedding models. Following common practice [18, 67], we compute Fréchet Distance (FD) and Kullback–Leibler (KL) distance. For FD, we adopt PaSST [30] (FD$_{\text{PaSST}}$), PANNs [29] (FD$_{\text{PANNs}}$), and VGGish [9] (FD$_{\text{VGG}}$) as embedding models. Note, PaSST operates at 32kHz, while both PANNs and VGGish operate at 16kHz. Moreover, both PaSST and PANNs produce global features, while VGGish processes non-overlapping 0.96s clips. For the KL distance, we adopt PANNs (KL$_{\text{PANNs}}$) and PaSST (KL$_{\text{PaSST}}$) as classifiers. We follow the implementation of Liu et al. [38].

Audio quality assesses the generation quality without comparing it to the ground truth using the Inception Score [56]. We adopt PANNs as the classifier following Wang et al. [67].

Semantic alignment assesses the semantic similarity between the input video and the generated audio. We use ImageBind [11] following Viertola et al. [65] to extract visual features from the input video and audio features from the generated audio and compute the average cosine similarity as "IB-score".

Temporal alignment assesses audio-visual synchrony with a synchronization score (DeSync). DeSync is predicted by Synchformer [19] as the misalignment (in seconds) between the audio and video. Note that Viertola et al. [65] also use the synchronization score but evaluate on audio (2.56s) that is shorter than the context window (4.8s) of Synchformer.

Method	Params	Distribution matching					Audio quality	Semantic align.	Temporal align.	
		$FD_{PaSST}\downarrow$	$FD_{PANNs}\downarrow$	$FD_{VGG}\downarrow$	$KL_{PANNs}\downarrow$	$KL_{PaSST}\downarrow$	IS↑	IB-score↑	DeSync↓	Time (s)↓
ReWaS [20]*	619M	141.38	17.54	1.79	2.87	2.82	8.51	14.82	1.062	15.97
Seeing&Hearing [69]*	415M	219.01	24.58	5.40	2.26	2.30	8.58	**33.99**	1.204	14.55
V-AURA [65]*◊	695M	218.50	14.80	2.88	2.42	2.07	10.08	27.64	0.654	16.55
VATT [40]†	-	131.88	10.63	2.77	**1.48**	1.41	11.90	25.00	1.195	-
Frieren [67]†◊	159M	106.10	11.45	1.34	2.73	2.86	12.25	22.78	0.851	-
FoleyCrafter [73]*	1.22B	140.09	16.24	2.51	2.30	2.23	15.68	25.68	1.225	1.67
V2A-Mapper [66]†◊	229M	84.57	8.40	0.84	2.69	2.56	12.47	22.58	1.225	-
MMAudio-S-16kHz	157M	70.19	5.22	**0.79**	1.65	1.59	14.44	29.13	0.483	**1.23**
MMAudio-S-44.1kHz	157M	65.25	5.55	1.66	1.67	1.44	**18.02**	32.27	0.444	1.30
MMAudio-M-44.1kHz	621M	61.88	4.74	1.13	1.66	1.41	17.41	32.99	0.443	1.35
MMAudio-L-44.1kHz	1.03B	**60.60**	**4.72**	0.97	1.65	**1.40**	17.40	33.22	**0.442**	1.96

Table 1. Video-to-audio results on the VGGSound test set. Following the common practice [67], the parameter counts exclude pretrained feature extractors (*e.g*., CLIP), latent space encoders/decoders, and vocoders. Time is the total running time using the official code to generate one sample with a batch size of one after warm-up and excludes any disk I/O operations on an H100 GPU. ∗: reproduced using official evaluation code. †: evaluated using generation samples obtained directly from the authors. ◊: does not use text input during testing. Note, Seeing&Hearing [69] directly optimizes ImageBind score during test time, therefore attains the highest IB-score.

Instead, we evaluate on longer (8s) audios by taking two crops (first 4.8s and last 4.8s) and averaging the results. Thus, the scores from Viertola et al. [65] are not directly comparable with ours.

4.2. Main Results

Video-to-audio. Table 1 compares our main results on the VGGSound [1] test set (∼15K videos) with existing state-of-the-art models. We evaluate all generations at 8s following Wang et al. [67] by truncating longer audio to 8 seconds. For V-AURA [65], we use the official autoregression code to generate 8s audio. ReWaS [20] only generates 5s audio thus we evaluate it as-is, by truncating the ground truth also to 5s – we indicate this discrepancy via the gray font in the table. Our smallest model (157M) demonstrates better distribution matching, audio quality, semantic alignment, and temporal alignment than prior methods, while being fast. A notable exception is the IB-score comparison with Seeing-and-Hearing [69] (SAH). We note that SAH directly optimizes the IB score during test time, which we do not perform. Further, our larger models continue to improve in FD_{PaSST} and IB-score, though we observe diminishing returns potentially limited by data quality and the amount of audio-visual data. Note, while our method uses more data for *multimodal joint training*, we *do not use more data overall* than some of the existing methods: Foley-Crafter [73], V2A-Mapper [66], ReWaS [20], and SAH [69] all finetune/incorporate a text-to-audio model that has been trained on audio-text data similar to the one we use. For a fair evaluation, we use the precomputed samples provided by [40, 66, 67], and reproduce the results using the official inference code for [20, 65, 69, 73]. Figure 3 visualizes our results and compares them with prior works. We present the results of a user study and comparison with Movie Gen Audio [52] in Appendices A and B. To address any potential

bias introduced by using the model-based DeSync metric, we additionally assess synchronization via model-free metrics on Greatest Hits [48] in Table 2, detailed in Appendix C.

Method	Acc. ↑	AP ↑	F1↑	DeSync↓
Frieren [67]	0.6949	0.7846	0.6550	0.851
V-AURA [65]	0.5852	0.8567	0.6441	0.654
FoleyCrafter [73]	0.4533	0.6939	0.4319	1.225
Seeing&Hearing [69]	0.1156	0.8342	0.1591	1.204
MMAudio-S-16kHz	**0.7637**	0.9010	**0.7928**	0.483
MMAudio-S-44.1kHz	0.7150	**0.9097**	0.7666	0.444
MMAudio-M-44.1kHz	0.7226	0.9054	0.7620	0.443
MMAudio-L-44.1kHz	0.7158	0.9064	0.7535	**0.442**

Table 2. Onset accuracy, average precision (AP), and F1-score on Greatest Hits, with DeSync on VGGSound for reference.

Text-to-audio. Our multimodal framework can be applied to *text-to-audio* synthesis without fine-tuning. Table 3 compares our method with state-of-the-art text-to-audio models using the AudioCaps [24] test set. For a fair comparison, we follow the evaluation protocol of GenAU [12] to evaluate 10s samples in the AudioCaps [24] test set without using CLAP re-ranking (used by AudioLDM [38]). We transcribe the baselines directly from Haji-Ali et al. [12], who have reproduced those results using officially released checkpoints under the same evaluation protocol. We assess FD_{PANNs}, FD_{VGG}, IS, and CLAP [68]. CLAP measures the semantic alignment between the generated audio and the input caption. While our main focus is on video-to-audio synthesis, MMAudio attains state-of-the-art semantic alignment (CLAP) and audio quality (IS), due to a rich semantic feature space learned from multimodal joint training. We note that we attain a worse FD_{VGG} score compared to recent works. We hypothesize that this is because VGGish processes local features (clips of 0.96s) while our strength lies in generating
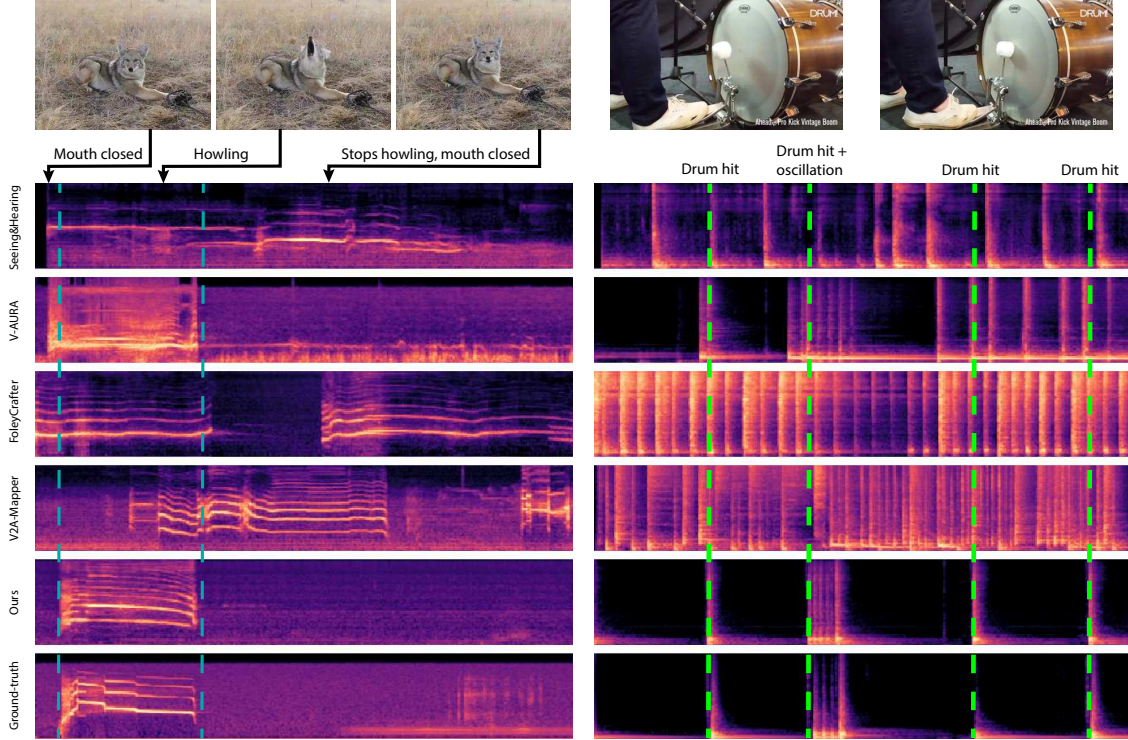
Figure 3. We visualize the spectrograms of generated audio (by prior works and our method) and the ground-truth. Note our method generates the audio effects most closely aligned to the ground-truth, while other methods often generate sounds not explained by the visual input and not present in the ground-truth.

Method	Params	FD$_{PANNs}$↓	FD$_{VGG}$↓	IS↑	CLAP↑
AudioLDM 2-L [39]	712M	32.50	5.11	8.54	0.212
TANGO [10]	866M	26.13	1.87	8.23	0.185
TANGO 2 [43]	866M	19.77	2.74	8.45	0.264
Make-An-Audio [16]	453M	27.93	2.59	7.44	0.207
Make-An-Audio 2 [15]	937M	15.34	1.27	9.58	0.251
GenAU-Large [12]	1.25B	16.51	**1.21**	11.75	0.285
MMAudio-S-16kHz	157M	14.42	2.98	11.36	0.282
MMAudio-S-44.1kHz	157M	15.26	2.74	11.32	0.331
MMAudio-M-44.1kHz	621M	**14.38**	4.07	12.02	**0.351**
MMAudio-L-44.1kHz	1.03B	15.04	4.03	**12.08**	0.348

Table 3. Text-to-audio results on the AudioCaps test set. For a fair comparison, we follow the evaluation protocol of [12] and transcribe all baselines directly from [12], who have reproduced those results using officially released checkpoints under the same evaluation protocol.

Training modalities	FD$_{PaSST}$↓	IS↑	IB-score↑	DeSync↓
AVT+AT	**70.19**	**14.44**	**29.13**	**0.483**
AV+AT	72.77	12.88	28.10	0.502
AVT+A	71.01	14.30	28.72	0.496
AVT	77.38	12.53	27.98	0.562
AV	77.27	12.69	28.10	0.502

Table 4. Results when we vary the training modalities. A: Audio, V: Video, T: Text. In the second and third rows, we mask away the text token in either audio-visual data or audio-text data. In the last two rows, we do not use any audio-text data.

globally and semantically consistent audio.

4.3. Ablations

We base all ablations on the small-16kHz model and evaluate distribution matching (FD$_{PaSST}$), audio quality (IS), semantic alignment (IB-score), and temporal alignment (DeSync) on the VGGSound [1] test set. We highlight our default setting using a blue background.

Cross-modal alignment. To elucidate the benefits of joint multimodal training, we mask away some modalities during training and observe the effects on the results, summarized

in Table 4. We denote the setting as (modalities used for audio-visual-text data + modalities used for audio-text data), where A: Audio, V: Video, T: Text. Our default setting (AVT+AT) means that we train on audio-visual-text data (VGGSound class labels as text input) and audio-text data. We make three observations:

1. Masking away the text modality from either the former (AV+AT) or the latter (AVT+A) leads to worse results. This suggests that having a joint "text feature space" is beneficial for multimodal training.
2. Adding uncaptioned audio data improves results (AVT

7

% audio-text data	$FD_{PaSST}\downarrow$	IS↑	IB-score↑	DeSync↓
100%	**70.19**	14.44	**29.13**	**0.483**
50%	71.03	**14.62**	29.11	0.489
25%	71.67	14.41	28.75	0.505
10%	79.21	13.55	27.47	0.514
None	77.38	12.53	27.98	0.562

Table 5. Results when we vary the amount of multimodal training data. For the first four rows, we sample audio-visual and audio-text data at a 1:1 ratio during training. For the last row, only audio-visual data is used.

    vs. AVT+A). This suggests that our network even benefits from training on unconditional generation data by learning the distribution of natural sounds.

3. When no audio-text data is used, using the simple class labels in VGGSound does not affect results significantly (AVT *vs*. AV). This suggests that training on large multimodal datasets, rather than adding a transformer branch in the network or using the class labels, is key.

Multimodal data. Training on a large collection of multimodal data is crucial. Table 5 shows our model's performance when we vary the amount of audio-text training data. We always sample audio-visual data and audio-text data at the same (roughly 1:1, see Appendix H) ratio except when we use no audio-text data. When we do not use audio-text data, we observe overfitting and stop training early. When more multimodal data is used, distribution matching (FD_{PaSST}), semantic alignment (IB-score), and temporal alignment (DeSync) improve with diminishing returns.

Conditional synchronization module. We compare several different methods for incorporating synchronization features: 1) our default of using the conditional synchronization module (Section 3.4); 2) incorporating the synchronization features into the visual branch of the multimodal transformer. Concretely, we upsample (with nearest neighbor) the CLIP features to 24fps, and then sum the CLIP features and Sync features after linear projections as the final visual feature. We illustrate this architecture in Figure A4; finally, 3) not using synchronization features. Table 6 (top) shows that our synchronization module attains the best temporal alignment. We note the "sum sync with visual" method achieves higher audio quality (IS) – we hypothesize that since upsampling CLIP features increased the number of tokens in the visual stream by three times, the model benefits from using the longer sequence for more fine-grained computations.

RoPE embeddings. We compare our aligned RoPE formulation with 1) not using RoPE embeddings and 2) non-aligned RoPE embeddings, *i.e.*, no frequency scaling in the visual branch. Table 6 (bottom) shows that using aligned RoPE embeddings [59] improves audio-visual synchrony.

ConvMLP. Table 7 (top) summarizes the performance difference of using MLP *vs*. ConvMLP. For the MLP model,

Variant	$FD_{PaSST}\downarrow$	IS↑	IB-score↑	DeSync↓
With sync module	70.19	14.44	29.13	**0.483**
Sum sync with visual	73.59	**16.70**	28.65	0.490
No sync features	**69.33**	15.05	**29.31**	0.973
Aligned RoPE	**70.19**	14.44	29.13	**0.483**
No RoPE	70.24	**14.54**	29.23	0.509
Non-aligned RoPE	70.25	**14.54**	29.25	0.496

Table 6. Results when we use synchronization features or RoPE embeddings differently.

Variant	$FD_{PaSST}\downarrow$	IS↑	IB-score↑	DeSync↓
ConvMLP	**70.19**	**14.44**	**29.13**	**0.483**
MLP	73.84	13.01	28.99	0.533
$N_1 = 4, N_2 = 8$	**70.19**	14.44	29.13	**0.483**
$N_1 = 2, N_2 = 13$	70.33	**15.18**	**29.39**	0.487
$N_1 = 6, N_2 = 3$	72.53	13.75	29.06	0.509

Table 7. Results when we vary the MLP architecture or the ratio between multi-/single-modality transformer blocks.

we increase h from $448 \to 512$, N_1 from $4 \to 6$, and N_2 from $8 \to 10$ to roughly match the number of parameters in the ConvMLP model. The ConvMLP model is better at capturing local temporal structure and thus has a better performance, especially in synchronization.

Ratio between N_1 and N_2. Table 7 (bottom) compares different assignments of the number of multimodal (N_1) and single-modal (N_2) transformer blocks with roughly the same parameter budget. We note our default assignment ($N_1 = 4, N_2 = 8$) performs similarly as using more single-modal blocks ($N_1 = 2, N_2 = 13$) and better than using fewer single-modal blocks ($N_1 = 6, N_2 = 3$). We think this is because using single-modal blocks allows us to build a deeper network with the same parameter count.

4.4. Limitations

Our model generates unintelligible mumbles when prompted to generate human speech (from seeing mouth movement or from text input). We believe human speech is inherently more complex (*e.g.*, with languages, tones, and grammars) and our model aimed at general audio effects (Foley) fails to adequately accommodate.

5. Conclusion

We propose MMAudio, the first multimodal training pipeline that jointly considers audio, video, and text modalities, resulting in effective data scaling and cross-modal semantic alignment. Combined with a conditional synchronization module, our method achieves a new state-of-the-art performance among public models and comparable performance with Movie Gen Audio. We believe a multimodal formulation is key for the synthesis of data in any modality and MMAudio lays the foundation in the audio-video-text space.

References

[1] Honglie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman. Vggsound: A large-scale audio-visual dataset. In *ICASSP*, 2020. 1, 5, 6, 7, 13, 14, 15, 20

[2] Peihao Chen, Yang Zhang, Mingkui Tan, Hongdong Xiao, Deng Huang, and Chuang Gan. Generating visually aligned sound from videos. *TIP*, 2020. 2

[3] Ziyang Chen, Prem Seetharaman, Bryan Russell, Oriol Nieto, David Bourgin, Andrew Owens, and Justin Salamon. Video-guided foley sound generation with multimodal controls. *arXiv*, 2024. 2

[4] Yoonjin Chung, Junwon Lee, and Juhan Nam. T-foley: A controllable waveform-domain diffusion model for temporal-event-guided foley sound synthesis. In *ICASSP*. IEEE, 2024. 2

[5] Konstantinos Drossos, Samuel Lipping, and Tuomas Virtanen. Clotho: An audio captioning dataset. In *ICASSP*. IEEE, 2020. 5, 15

[6] Yuexi Du, Ziyang Chen, Justin Salamon, Bryan Russell, and Andrew Owens. Conditional generation of audio from video via foley analogies. In *CVPR*, 2023. 2, 14

[7] Benjamin Elizalde, Soham Deshmukh, and Huaming Wang. Natural language supervision for general-purpose audio representations. In *ICASSP*, 2024. 13

[8] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024. 1, 3

[9] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *ICASSP*. IEEE, 2017. 1, 5, 15, 16

[10] Deepanway Ghosal, Navonil Majumder, Ambuj Mehrish, and Soujanya Poria. Text-to-audio generation using instruction tuned llm and latent diffusion model. *arXiv preprint arXiv:2304.13731*, 2023. 7

[11] Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. Imagebind: One embedding space to bind them all. In *CVPR*, 2023. 2, 5, 13

[12] Moayed Haji-Ali, Willi Menapace, Aliaksandr Siarohin, Guha Balakrishnan, Sergey Tulyakov, and Vicente Ordonez. Taming data and transformers for audio generation. *arXiv preprint arXiv:2406.19388*, 2024. 6, 7

[13] Moayed Haji-Ali, Willi Menapace, Aliaksandr Siarohin, Ivan Skorokhodov, Alper Canberk, Kwot Sin Lee, Vicente Ordonez, and Sergey Tulyakov. Av-link: Temporally-aligned diffusion features for cross-modal audio-video generation. *arXiv*, 2024. 2

[14] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 5

[15] Jiawei Huang, Yi Ren, Rongjie Huang, Dongchao Yang, Zhenhui Ye, Chen Zhang, Jinglin Liu, Xiang Yin, Zejun Ma, and Zhou Zhao. Make-an-audio 2: Temporal-enhanced text-to-audio generation. *arXiv preprint arXiv:2305.18474*, 2023. 7, 16

[16] Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. In *ICML*, 2023. 7

[17] Zhiqi Huang, Dan Luo, Jun Wang, Huan Liao, Zhiheng Li, and Zhiyong Wu. Rhythmic foley: A framework for seamless audio-visual alignment in video-to-audio synthesis. *arXiv preprint arXiv:2409.08628*, 2024. 2

[18] Vladimir Iashin and Esa Rahtu. Taming visually guided sound generation. In *BMVC*, 2021. 2, 5

[19] Vladimir Iashin, Weidi Xie, Esa Rahtu, and Andrew Zisserman. Synchformer: Efficient synchronization from sparse cues. In *ICASSP*. IEEE, 2024. 2, 3, 5, 13, 17

[20] Yujin Jeong, Yunji Kim, Sanghyuk Chun, and Jiyoung Lee. Read, watch and scream! sound generation from text and video. *arXiv preprint arXiv:2407.05551*, 2024. 1, 2, 5, 6

[21] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. 1

[22] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *CVPR*, 2024. 16, 18

[23] Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fréchet audio distance: A metric for evaluating music enhancement algorithms. In *Interspeech*, 2018. 2

[24] Chris Dongjoo Kim, Byeongchang Kim, Hyunmin Lee, and Gunhee Kim. Audiocaps: Generating captions for audios in the wild. In *NAACL-HLT*, 2019. 5, 6, 15

[25] Gwanghyun Kim, Alonso Martinez, Yu-Chuan Su, Brendan Jou, José Lezama, Agrim Gupta, Lijun Yu, Lu Jiang, Aren Jansen, Jacob Walker, et al. A versatile diffusion transformer with mixture of noise levels for audiovisual generation. In *NeurIPS*, 2024. 2

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 18

[27] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 3, 16

[28] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *NeurIPS*, 2017. 17

[29] Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D Plumbley. Panns: Large-scale pretrained audio neural networks for audio pattern recognition. *TASLP*, 2020. 5

[30] Khaled Koutini, Jan Schlüter, Hamid Eghbal-Zadeh, and Gerhard Widmer. Efficient training of audio transformers with patchout. In *Interspeech*, 2022. 5

[31] Etienne Labb, Thomas Pellegrini, and Julien Pinquier. Conette: An efficient audio captioning system leveraging multiple datasets with task embedding. *TASLP*, 2024. 15

[32] Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2024. 3

[33] LAION-AI. Audio dataset project. https://github.com/LAION-AI/audio-dataset, 2024. 15, 16

[34] Junwon Lee, Jaekwon Im, Dabin Kim, and Juhan Nam. Video-foley: Two-stage video-to-sound generation via temporal event condition for foley sound. *arXiv preprint arXiv:2408.11915*, 2024. 2

[35] Sang-gil Lee, Wei Ping, Boris Ginsburg, Bryan Catanzaro, and Sungroh Yoon. Bigvgan: A universal neural vocoder with large-scale training. In *ICLR*, 2023. 3, 16

[36] Rensis Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 1932. 13

[37] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. 3

[38] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. In *ICML*, 2023. 3, 5, 6

[39] Haohe Liu, Yi Yuan, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Qiao Tian, Yuping Wang, Wenwu Wang, Yuxuan Wang, and Mark D Plumbley. Audioldm 2: Learning holistic audio generation with self-supervised pretraining. *TASLP*, 2024. 7

[40] Xiulong Liu, Kun Su, and Eli Shlizerman. Tell what you hear from what you see – video to audio generation through text. In *NeurIPS*, 2024. 2, 5, 6, 13

[41] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *ICLR*, 2019. 18

[42] Simian Luo, Chuanhao Yan, Chenxu Hu, and Hang Zhao. Diff-foley: Synchronized video-to-audio synthesis with latent diffusion models. In *NeurIPS*, 2024. 2

[43] Navonil Majumder, Chia-Yu Hung, Deepanway Ghosal, Wei-Ning Hsu, Rada Mihalcea, and Soujanya Poria. Tango 2: Aligning diffusion-based text-to-audio generations through direct preference optimization. In *ACM MM*, 2024. 7

[44] Xinhao Mei, Varun Nagaraja, Gael Le Lan, Zhaoheng Ni, Ernie Chang, Yangyang Shi, and Vikas Chandra. Foleygen: Visually-guided audio generation. *arXiv preprint arXiv:2309.10537*, 2023. 2

[45] Xinhao Mei, Gael Le Lan, Haohe Liu, Zhaoheng Ni, Varun K Nagaraja, Anurag Kumar, Yangyang Shi, and Vikas Chandra. Towards temporally synchronized visually indicated sounds through scale-adapted positional embeddings. In *Audio Imagination: NeurIPS 2024 Workshop AI-Driven Speech, Music, and Sound Generation*, 2024. 2

[46] 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. *TASLP*, 2024. 2, 5, 15

[47] Shentong Mo, Jing Shi, and Yapeng Tian. Text-to-audio generation synchronized with videos. *arXiv preprint arXiv:2403.07938*, 2024. 2

[48] Andrew Owens, Phillip Isola, Josh McDermott, Antonio Torralba, Edward H Adelson, and William T Freeman. Visually indicated sounds. In *CVPR*, 2016. 6, 14, 22

[49] Santiago Pascual, Chunghsin Yeh, Ioannis Tsiamas, and Joan Serrà. Masked generative video-to-audio transformers with enhanced synchronicity. In *ECCV*, 2024. 2

[50] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018. 2, 4

[51] Karin Petrini, Sofia Dahl, Davide Rocchesso, Carl Haakon Waadeland, Federico Avanzini, Aina Puce, and Frank E Pollick. Multisensory integration of drumming actions: musical expertise affects perceived audiovisual asynchrony. *Experimental brain research*, 198:339–352, 2009. 1

[52] Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024. 6, 13, 14

[53] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICLR*, 2021. 1, 3

[54] Yong Ren, Chenxing Li, Manjie Xu, Wei Liang, Yu Gu, Rilin Chen, and Dong Yu. Sta-v2a: Video-to-audio generation with semantic and temporal alignment. *arXiv preprint arXiv:2409.08601*, 2024. 2

[55] Ludan Ruan, Y Ma, Huan Yang, Huiguo He, Bei Liu, Jianlong Fu, Nicholas Jing Yuan, Qin Jin, and Baining Guo. Mm-diffusion: Learning multi-modal diffusion models for joint audio and video generation. 2023 ieee. In *CVPR*, 2023. 2

[56] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NeurIPS*, 2016. 2, 5

[57] Stanley Smith Stevens, John Volkmann, and Edwin Broomell Newman. A scale for the measurement of the psychological magnitude pitch. *The journal of the acoustical society of america*, 1937. 3, 16

[58] Robynn J Stilwell. The fantastical gap between diegetic and nondiegetic, 2007. 1

[59] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 2024. 4, 8, 18

[60] Kun Su, Kaizhi Qian, Eli Shlizerman, Antonio Torralba, and Chuang Gan. Physics-driven diffusion models for impact sound synthesis from videos. In *CVPR*, 2023. 2

[61] Zineng Tang, Ziyi Yang, Mahmoud Khademi, Yang Liu, Chenguang Zhu, and Mohit Bansal. Codi-2: In-context interleaved and interactive any-to-any generation. In *CVPR*, 2024. 2

[62] Zineng Tang, Ziyi Yang, Chenguang Zhu, Michael Zeng, and Mohit Bansal. Any-to-any generation via composable diffusion. In *NeurIPS*, 2024. 2

[63] Alexander Tong, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Kilian Fatras, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based

generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023. 3

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 3, 4

[65] Ilpo Viertola, Vladimir Iashin, and Esa Rahtu. Temporally aligned audio for video with autoregression. *arXiv preprint arXiv:2409.13689*, 2024. 2, 5, 6, 13

[66] Heng Wang, Jianbo Ma, Santiago Pascual, Richard Cartwright, and Weidong Cai. V2a-mapper: A lightweight solution for vision-to-audio generation by connecting foundation models. In *AAAI*, 2024. 1, 2, 6, 13

[67] Yongqi Wang, Wenxiang Guo, Rongjie Huang, Jiawei Huang, Zehan Wang, Fuming You, Ruiqi Li, and Zhou Zhao. Frieren: Efficient video-to-audio generation with rectified flow matching. In *NeurIPS*, 2024. 1, 3, 5, 6, 16

[68] 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 *ICASSP*, 2023. 6, 13

[69] Yazhou Xing, Yingqing He, Zeyue Tian, Xintao Wang, and Qifeng Chen. Seeing and hearing: Open-domain visual-audio generation with diffusion latent aligners. In *CVPR*, 2024. 1, 2, 6, 13

[70] Jinlong Xue, Yayue Deng, Yingming Gao, and Ya Li. Auffusion: Leveraging the power of diffusion and large language models for text-to-audio generation. *arXiv preprint arXiv:2401.01044*, 2024. 15

[71] Qi Yang, Binjie Mao, Zili Wang, Xing Nie, Pengfei Gao, Ying Guo, Cheng Zhen, Pengfei Yan, and Shiming Xiang. Draw an audio: Leveraging multi-instruction for video-to-audio synthesis. *arXiv preprint arXiv:2409.06135*, 2024. 2

[72] Lin Zhang, Shentong Mo, Yijing Zhang, and Pedro Morgado. Audio-synchronized visual animation. *ECCV*, 2024. 15

[73] Yiming Zhang, Yicheng Gu, Yanhong Zeng, Zhening Xing, Yuancheng Wang, Zhizheng Wu, and Kai Chen. Foleycrafter: Bring silent videos to life with lifelike and synchronized sounds. *arXiv preprint arXiv:2407.01494*, 2024. 1, 2, 6, 15

[74] Bin Zhu, Bin Lin, Munan Ning, Yang Yan, Jiaxi Cui, HongFa Wang, Yatian Pang, Wenhao Jiang, Junwu Zhang, Zongwei Li, et al. Languagebind: Extending video-language pretraining to n-modality by language-based semantic alignment. In *ICLR*, 2024. 2

Table of Contents

A. User Study

In addition to the objective metrics presented in Table 1, we have also performed a user study for subjective evaluation on the VGGSound [1] test set. For comparisons, we have selected our best model (MMAudio-L-44.1kHz) and four best baselines:

1. Seeing and Hearing [69], as it has the highest ImageBind (*i.e.*, best semantic alignment with videos) score, besides ours.
2. V-AURA [65], as it has the lowest DeSync (*i.e.*, best temporal alignment) with videos, besides ours.
3. VATT [40], as it has the lowest Kullback–Leibler divergence (*i.e.*, KL_{PANNs} and KL_{PaSST}), besides ours.
4. V2A-Mapper [66], as it has the lowest Fréchet distances (*i.e.*, FD_{PaSST}, FD_{PANNs}, and FD_{VGG}), besides ours.

We sample eight videos from the VGGSound [1] test set, after excluding videos that are of low-resolution (below 360p) or that contain human speech. In total, each participant evaluates 40 videos (8 videos × 5 methods). We group the samples for the same video, and randomly shuffle the ordering in each group to avoid bias. We ask each participant to rate the generation in three aspects using the Likert scale [36] (1-5; strongly disagree, disagree, neutral, agree, strongly agree) providing the following instructions:

(a) The audio is of **high quality**.
   Explanation: An audio is low-quality if it is noisy, unclear, or muffled. In this aspect, ignore visual information and focus on the audio.
(b) The audio is **semantically aligned** with the video.
   Explanation: An audio is semantically misaligned with the video if the audio effects are unlikely to occur in the scenario depicted by the video, *e.g.*, the sound of an explosion in a library.
(c) The audio is **temporally aligned** with the video.
   Explanation: An audio is temporally misaligned with the video if the audio sounds delayed/advanced compared to the video, or when audio events happen at the wrong time (*e.g.*, in the video, the drummer hits the drum twice and stops; but in the audio, the sound of the drum keeps occurring).

In total, we have collected 920 responses in each of these aspects from 23 participants. Table A1 summarizes the results from the user study. MMAudio receives significantly higher ratings in all three aspects from the users, which aligns with the objective metrics presented in Table 1 of the main paper.

Method	Audio quality↑	Semantic alignment↑	Temporal alignment↑
Seeing&Hearing [69]	2.65±1.05	3.10±1.24	1.85±0.99
V-AURA [65]	3.59±1.02	3.70±1.17	3.65±1.16
VATT [40]	2.66±0.99	3.32±1.17	2.04±1.07
V2A-Mapper [66]	3.00±0.95	3.28±1.27	2.03±1.11
MMAudio-L-44.1kHz	**4.14±0.77**	**4.52±0.74**	**4.46±0.80**

Table A1. Average ratings for each method from the user study. We show mean±std in each aspect.

B. Comparisons with Movie Gen Audio

Recently, Movie Gen Audio [52] has been introduced for generating sound effects and music for input videos. While Movie Gen Audio's technical details are sparse, it represents the industry's current state-of-the-art video-to-audio synthesis algorithm. Its 13-billion parameters model has been trained on non-publicly accessible data that is $> 100\times$ larger than ours. Nevertheless, we compare MMAudio to Movie Gen Audio [52] to benchmark the differences between public and private models.

At the time of writing, the only accessible outputs from Movie Gen Audio are 527[2] generations in the "Movie Gen Audio Bench" dataset. All the videos from Movie Gen Audio Bench are generated by MovieGen [52], which we note is different from the distribution of real-world videos (*e.g.*, over-smoothed textures, slow motions). Since these are synthetic videos, there is no corresponding ground-truth audio. We run our best model MMAudio-L-44.1kHz on these videos and the corresponding audio prompts (which Movie Gen Audio also uses) and compare our generations with Movie Gen Audio.

Since there is no ground truth audio, among the standard metrics that we have used in the main paper, we can only evaluate Inception Score (IS, audio quality), IB-score (ImageBind [11] similarly, semantic alignment between video and audio), DeSync (misalignment predicted by SynchFormer [19] between video and audio), and CLAP [7, 68] (alignment between text and

---

[2] While the MovieGen technical report mentioned 538 samples, only 527 were released at the time of writing.

audio). Additionally, we have conducted a user study following the protocol of Appendix A, and have excluded audios with very low volume (cannot be heard clearly at a normal volume) generated by Movie Gen Audio to prevent bias. We sampled a total of 5 videos and received 230 responses in each of the aspects from 23 participants.

Table A2 summarizes our results. In subjective metrics, MMAudio is comparable to Movie Gen Audio – slightly worse in semantic alignment and slightly better in temporal alignment. In objective metrics, we observe the same trend – MMAudio and Movie Gen Audio obtain the same audio quality (IS) score, Movie Gen Audio has a better semantic alignment (IB-score and CLAP), and MMAudio has a better video-audio synchrony (DeSync).

			Subjective metrics			Objective metrics			
Method	Param	Training data	Audio qual.↑	Semantic align.↑	Temporal align.↑	IS↑	IB-score↑	CLAP↑	DeSync↓
Movie Gen Audio [52]	13B	$\mathcal{O}(1,000,000)$h	**3.93±0.92**	**4.36±0.74**	3.52±1.21	**8.40**	**36.26**	**0.4409**	1.006
MMAudio-L-44.1kHz	1.03B	∼ 8,200h	**3.93±0.89**	4.26±0.71	**3.62±1.03**	**8.40**	27.01	0.4324	**0.771**

Table A2. Comparisons between Movie Gen Audio and MMAudio in both subjective metrics (from user study) and objective metrics. For the subjective metrics, we show mean±std.

Further, in terms of IB-score, we find that MMAudio struggles more in some videos, while Movie Gen Audio delivers more consistent results. We plot the sorted IB-score comparing MMAudio and Movie Gen Audio in Figure A1 (left). Movie Gen Audio consistently performs better in the low-performance regime, but the gap narrows in the high-performance region. We believe this is due to our limited training data, which is unable to adequately cover the data in Movie Gen Audio Bench and thus falls short in unfamiliar video types. Note, our only video-audio dataset for training is VGGSound [1] which contains videos for 310 classes. We hypothesize that collecting open-world data beyond these classes can effectively reduce this performance gap. The same phenomenon occurs at a much smaller scale for the CLAP score, which might be because we use more audio-text data. Figure A2 shows examples where we obtain a substantially higher/lower IB-score on videos with concepts well/not well covered by the training data.
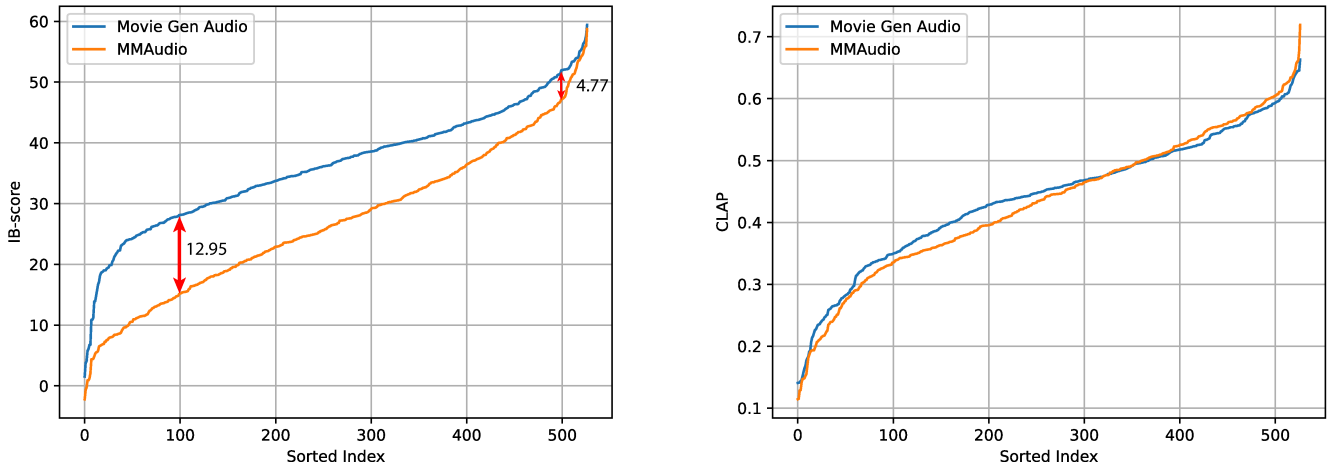

Figure A1. Sorted MMAudio and Movie Gen Audio performance scores in Movie Gen Audio Bench.

C. Evaluation on the Greatest Hits Dataset

To address any potential bias by using the model-based DeSync metric, we conduct an additional experiment to assess temporal alignment by comparing the onsets of generated audio with ground-truth labels. Concretely, we use the *Greatest Hits* [48] test set (244 videos) which contains videos with distinct and labeled sound events (*a drumstick hitting <object>*). Notably, neither our models nor the baselines have been trained on this dataset. We test both our method and baselines (using available code) on each video's first 8 seconds (due to the models' constraints): we extract onsets from the generated audio following [6] and compare them with the labeled sound events. We assess performance using accuracy, average precision (AP), and F1-score. We provide the results in Table 2 and visualize the spectrograms in Figure A8. MMAudio achieves significantly better performance in these *model-free* metrics. Note that a high AP (not accuracy or F1) can be achieved by generating very few onsets (*e.g.*, silent/noise), which is the case in Seeing&Hearing.

14

Audio prompt: rhythmic splashing and lapping of water
IB-score (Movie Gen Audio): 42.74
IB-score (MMAudio, ours): 53.95

Audio prompt: creamy sound of mashed potatoes being scooped
IB-score (Movie Gen Audio): 30.94
IB-score (MMAudio, ours): 10.52

Figure A2. Examples of videos in Movie Gen Audio Bench that are well/not well covered by our training data. Left: with a familiar concept in our training data (516 swimming videos in the VGGSound training set), MMAudio achieves a higher IB-score. Right: with an unfamiliar concept (there are no videos about mashed potatoes in VGGSound [1], according to the provided labels), MMAudio attains a significantly lower IB-score.

D. Ablations on Filling in Missing Modalities

Among our training data, VGGSound is the only tri-modal (with class names as text) dataset while all others are audio-text. For other data, we replace missing visual modalities (CLIP and Sync features) with end-to-end learnable embeddings ($\varnothing_v$ and $\varnothing_{syn}$) and missing text modalities with the empty string ($\varnothing_t$). We believe other methods to fill in missing modalities would be similarly effective since the deep net likely adapts. Indeed, replacing the missing modalities with either all learnable embeddings or zeros yields no significant difference (Table A3). Note, we also drop modalities randomly during training to enable classifier-free guidance, which enhances the model's robustness to missing modalities.

Method	$FD_{PaSST}\downarrow$	$IS\uparrow$	IB-score$\uparrow$	DeSync$\downarrow$
Ours	70.19	14.44	29.13	0.483
With all learnable	70.13	14.63	29.23	0.494
With zeros	69.91	14.60	29.22	0.496

Table A3. Comparisons of different methods to fill in missing modalities. As expected, there is no significant difference as the deep net learns to adapt.

E. Details on Data Overlaps

We note that there are training and testing data overlaps among commonly used datasets for video-to-audio generation. For example, AudioSet [9] is commonly used to train VAE encoders/decoders but it contains test set data from VGGSound [1] and AudioCaps [24]. Additionally, AudioCaps is often used to train text-to-audio models [70], which is then used as the backbone for video-to-audio models which evaluate on VGGSound [1] – however, part of the VGGSound test set overlaps with the AudioCaps training set. Moreover, AVSync15 [72], which is sometimes used jointly with VGGSound for training/evaluating video-to-audio algorithms [73], contains severe cross-contamination with VGGSound. This results in biased evaluations in both VGGSound and AVSync15. To our best knowledge, this data contamination is not yet addressed in the video-to-audio community. We thank Labb et al. [31] for raising this issue in the audio captioning field, which has helped us identify this problem.

Table A4 summarizes the observed overlaps. The overlaps with WavCaps [46] and Freesound [33] have been included as part of their release, which we do not repeat in our table.

We have carefully removed from our training data (AudioSet [9], AudioCaps [24], Clotho [5], Freesound [33], WavCaps [46], and VGGSound [1]) anything that overlaps with any of the test sets (VGGSound and AudioCaps). Additionally, we have also removed from our training data the test set of Clotho [5]. Since most baselines have been trained on VGGSound, we elect not to evaluate on AVSync15.

Test sets (number of samples)	Training sets			
	AudioSet	AudioCaps	VGGSound	AVSync15
AudioCaps (975)	580 (59.5%)	-	147 (15.1%)	-
VGGSound (15,496)	132 (0.9%)	13 (0.1%)	-	59 (0.4%)
AVSync-15 (150)	-	-	144 (96.0%)	-

Table A4. Overlaps between training and test sets of different datasets. The percentage denotes the proportion of overlapping data in the entire test set. "-" means that we did not compute this data (we do not train or test on AVSync15).

F. Details on the Audio Latents

As mentioned in the main paper, we obtain the audio latents by first transforming audio waveforms with the short-time Fourier transform (STFT) and extracting the magnitude component as mel spectrograms [57]. Then, spectrograms are encoded into latents by a pretrained variational autoencoder (VAE) [27]. During testing, the generated latents are decoded by the VAE into spectrograms, which are then vocoded by a pretrained vocoder [35] into audio waveforms. Table A5 tabulates our STFT parameters and latent information.

For the VAE, we follow the 1D convolutional network design of Make-An-Audio 2 [15] with a downsampling factor of 2 and trained with reconstruction, adversarial, and Kullback–Leibler divergence (KL) objectives. We note that the default setting leads to extreme values in the latent at the end of every sequence ($\pm 10\sigma$ away). To tackle this problem, we have applied the magnitude-preserving network design from EDM2 [22], by replacing the convolutional, normalization, addition, and concatenation layers with magnitude-preserving equivalents. While this change removes the extreme values, it leads to no significant empirical performance difference. We train the 16kHz model on AudioSet [9], following Make-An-Audio 2 [15]. For the 44.1kHz model, we increase the hidden dimension from 384 to 512 and train it on AudioSet [9] and Freesound [33] to accommodate the increased reconstruction difficulty due to a higher sampling rate.

For vocoders, we use the BigVGAN [35] trained by Make-An-Audio 2 [15] in our 16kHz model. For our 44.1kHz model, we use BigVGAN-v2 [35] (the `bigvgan_v2_44khz_128band_512x` checkpoint).

Model variants	Latent frame rate	# latent channels	# mel bins	# FFTs	Hop size	Window size	Window function
16kHz	31.25	20	80	1024	256	1024	Hann
44.1kHz	43.07	40	128	2048	512	2048	Hann

Table A5. Short-time Fourier transform (STFT) parameters and latent information.

G. Network Details

G.1. Model Variants

Our default model generates 16kHz audio encoded as 20-dimensional, 31.25fps latents (following Frieren [67]), with $N_1 = 4, N_2 = 8, h = 448$. We refer to this default model as 'S-16kHz'. To faithfully capture higher frequencies, we also train a 44.1kHz model ('S-44.1kHz') that generates 40-dimensional, 43.07fps latents while all other settings are identical to the default. To scale up the high-frequency model, we first double the hidden dimension to match the doubled latent dimension, *i.e.*, we use $N_1 = 4, N_2 = 8, h = 896$ and refer to this model using 'M-44.1kHz'. Finally, we scale the number of layers, *i.e.*, $N_1 = 7, N_2 = 14, h = 896$ and refer to this model via 'L-44.1kHz'. These model variants are summarized in Table A6.

G.2. Projection Layers

We use projection layers to project input text, visual, and audio features to the hidden dimension h and for initial aggregation of the temporal context.

Text feature projection. We use a linear layer that projects to h, followed by an MLP.

Clip feature projection. We use a linear layer that projects to h, followed by a ConvMLP with a kernel size of 3 and a padding of 1.

Model variants	Params	# multimodal blocks N_1	# single-modal blocks N_2	Hidden dim h	Latent dim	Time (s)
S-16kHz	157M	4	8	448	20	1.23
S-44.1kHz	157M	4	8	448	40	1.30
M-44.1kHz	621M	4	8	896	40	1.35
L-44.1kHz	1.03B	7	14	896	40	1.96

Table A6. Summary for different MMAudio model variants. Time is the total running time to generate one sample with a batch size of one after warm-up and excludes any disk I/O operations on an H100 GPU.

Sync feature projection. We use a 1D convolutional layer with a kernel size of 7 and a padding of 3 that projects to h, an SELU [28] activation layer, followed by a ConvMLP with a kernel size of 3 and a padding of 1.

Audio feature projection. We use a 1D convolutional layer with a kernel size of 7 and a padding of 3 that projects to h, an SELU [28] activation layer, followed by a ConvMLP with a kernel size of 7 and a padding of 3.

G.3. Gating

The gating layers are similar to the adaptive normalization layers (adaLN). Each global gating layer modulates its input $y \in \mathbb{R}^{L \times h}$ (L is the sequence length) with the global condition c_g as follows:

$$\text{Gating}_g(y, c_g) = y \cdot \mathbf{1} \mathbf{W}_g(c_g). \tag{A1}$$

Here, $\mathbf{W}_g \in \mathbb{R}^{h \times h}$ is an MLP, and $\mathbf{1}$ is a $L \times 1$ all-ones matrix, which "broadcasts" the scales to match the sequence length L – such that the same condition is applied to all tokens in the sequence (hence global).

Similarly, for per-token gating layers, the frame-aligned conditioning c_f is injected into the audio stream for precise feature modulation via

$$\text{Gating}_f(y, c_f) = y \cdot \mathbf{W}_f(c_f), \tag{A2}$$

where $\mathbf{W}_f \in \mathbb{R}^{h \times h}$ is an MLP. Different from Equation (A1), the scales are applied per token without broadcasting,

G.4. Details on Synchronization Features

We use the visual encoder of Synchformer [19] to extract synchronization features. We use the pretrained audio-visual synchronization model trained on AudioSet, provided by Iashin et al. [19]. As input, we obtain frames at 25 fps. Synchformer partitions these frames into overlapping clips of 16 frames with stride 8 and produces features of length 8 for each clip. Thus, for a video of length T_{sec} seconds, the sequence length of the synchronization features is

$$L_{\text{sync}} = 8 \left(\left\lfloor \frac{25 T_{\text{sec}} - 16}{8} \right\rfloor + 1 \right). \tag{A3}$$

The corresponding feature fps is

$$\text{FPS}_{\text{sync}} = \frac{L_{\text{text}}}{T_{\text{sec}}}. \tag{A4}$$

In this paper, we experimented with $T_{\text{sec}} = 8$ and $T_{\text{sec}} = 10$. In both cases, FPS_{sync} is exactly 24. Additionally, we introduce a learnable positional embedding of length 8 (matching the number of features in each clip processed by Synchformer) that is added to the Synchformer features, as illustrated in Figure A3.

G.5. Illustration of the "sum sync with visual" Ablation

Figure A4 illustrates the network architecture for the "sum sync with visual" ablation in the "conditional synchronization module" paragraph. The visual features are upsampled using the nearest neighbor to match the frame rate of the synchronization features. This architecture has a worse FD_{PaSST}, IB-score, synchronization (DeSync) but a better inception score (IS), which we hypothesize is due to the increased number of visual tokens in the upsampling step, leading to finer-grained computations.
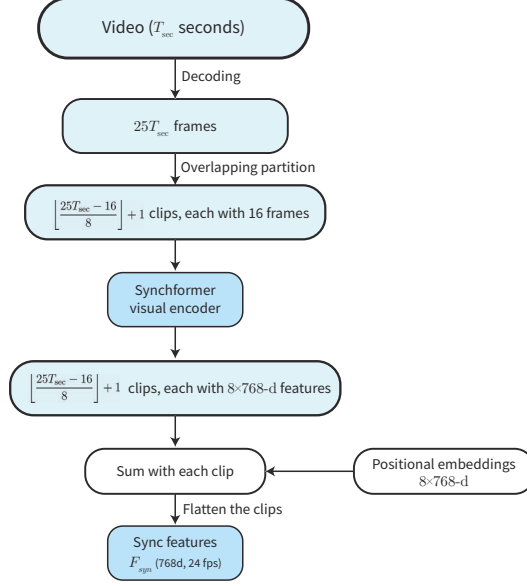
Figure A3. Synchformer feature extraction.

G.6. Visualization of Aligned RoPE

To visualize the effects of using aligned RoPE [59], we compare the dot-product affinity of two sequences $\mathbf{1}^{250 \times C}$ and $\mathbf{1}^{64 \times C}$ when RoPE is applied. Here, 250 represents the audio sequence length (31.25 fps for 8 seconds), 64 represents the visual sequence length (8 fps for 8 seconds), and $C = 64$ is the channel size. Concretely, we visualize

$$\text{RoPE}_{\text{default}}(\mathbf{1}^{250 \times C}) \left(\text{RoPE}_{\text{default}}(\mathbf{1}^{64 \times C})\right)^T, \tag{A5}$$

and,

$$\text{RoPE}_{\text{aligned}}(\mathbf{1}^{250 \times C}) \left(\text{RoPE}_{\text{aligned}}(\mathbf{1}^{64 \times C})\right)^T, \tag{A6}$$

in Figure A5. Temporal alignment is attained when we use aligned RoPE.

H. Training Details

Training setup. Unless otherwise specified, we used the same set of hyperparameters for all model sizes. To train the models, we use the base learning rate of 1e-4, with a linear warm-up schedule of 1K steps, for 300K iterations, and with a batch size of 512. We use the AdamW optimizer [26, 41] with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and a weight decay of 1e-6. If the default $\beta_2 = 0.999$ was used instead, we notice occasional training collapse (to NaN). For learning rate scheduling, we reduce the learning rate to 1e-5 after 80% of the training steps, and once again to 1e-6 after 90% of the training steps. For model exponential moving average (EMA), we use the post-hoc EMA [22] formulation with a relative width $\sigma_{\text{rel}} = 0.05$ for all models. For training efficiency, we use bf16 mixed precision training, and all the audio latents and visual embeddings are precomputed offline and loaded during training. Table A7 summarizes the training resources we used for each model size.

Model	Number of GPUs used	Number of hours to train	Total GPU-hours
MMAudio-S-16kHz	2	22	44
MMAudio-S-44.1kHz	2	26	52
MMAudio-M-44.1kHz	8	21	168
MMAudio-L-44.1kHz	8	38	304

Table A7. The amount of training resources used for each model size. H100 GPUs are used in all settings.
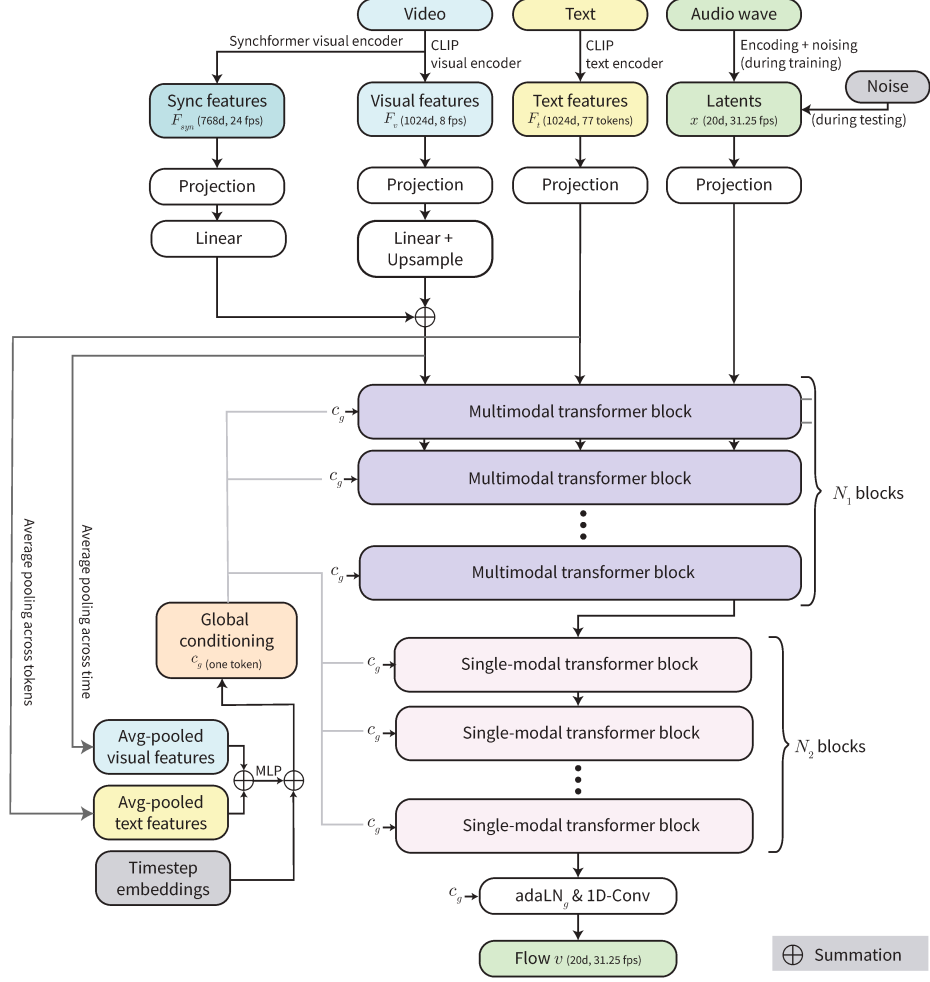
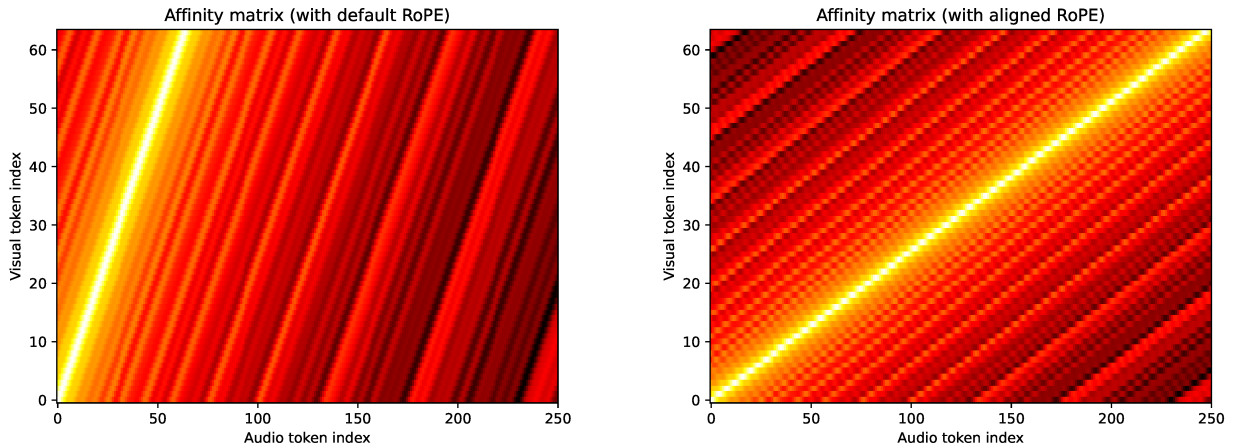Figure A4. Illustration of the "sum sync with visual" ablation.

Figure A5. Affinity visualizations between two sequences with different frame rates when default/aligned RoPE embeddings are used. **Left**: with default RoPE, the sequences are not aligned. **Right**: with our proposed aligned RoPE, we attain temporal alignment.

Balancing multimodal training data. Since we have significantly more audio-text training data (951K) than audio-text-visual data (180K), we balance the dataset by duplicating the audio-text-visual samples before random shuffling in each epoch.

By default, we apply a 5X duplication for a rough 1:1 data sampling ratio. For the "medium" and "large" models, we reduce the duplication ratio to 3X to mitigate overfitting.

Duplicated videos. We observe VGGSound dataset [1] contains duplicated videos, likely due to multiple uploads of the same video to YouTube under different video IDs. For instance, videos `4PjEi5fFD6A` (in training set) and `FhaYvI1yrUM` (in test set) are the same video.[3] In Appendix E, we remove train-test sets overlaps by comparing the video IDs, though this method does not eliminate repeated uploads. Since prior works have been trained on the same dataset, our training scheme remains a fair comparison.

I. Additional Visualizations

We provide generated samples and comparisons with state-of-the-art methods on our project page https://hkchengrex.com/MMAudio/video_main.html. Below, we provide additional spectrogram visualizations comparing our method with prior works in Figures A6 to A8.
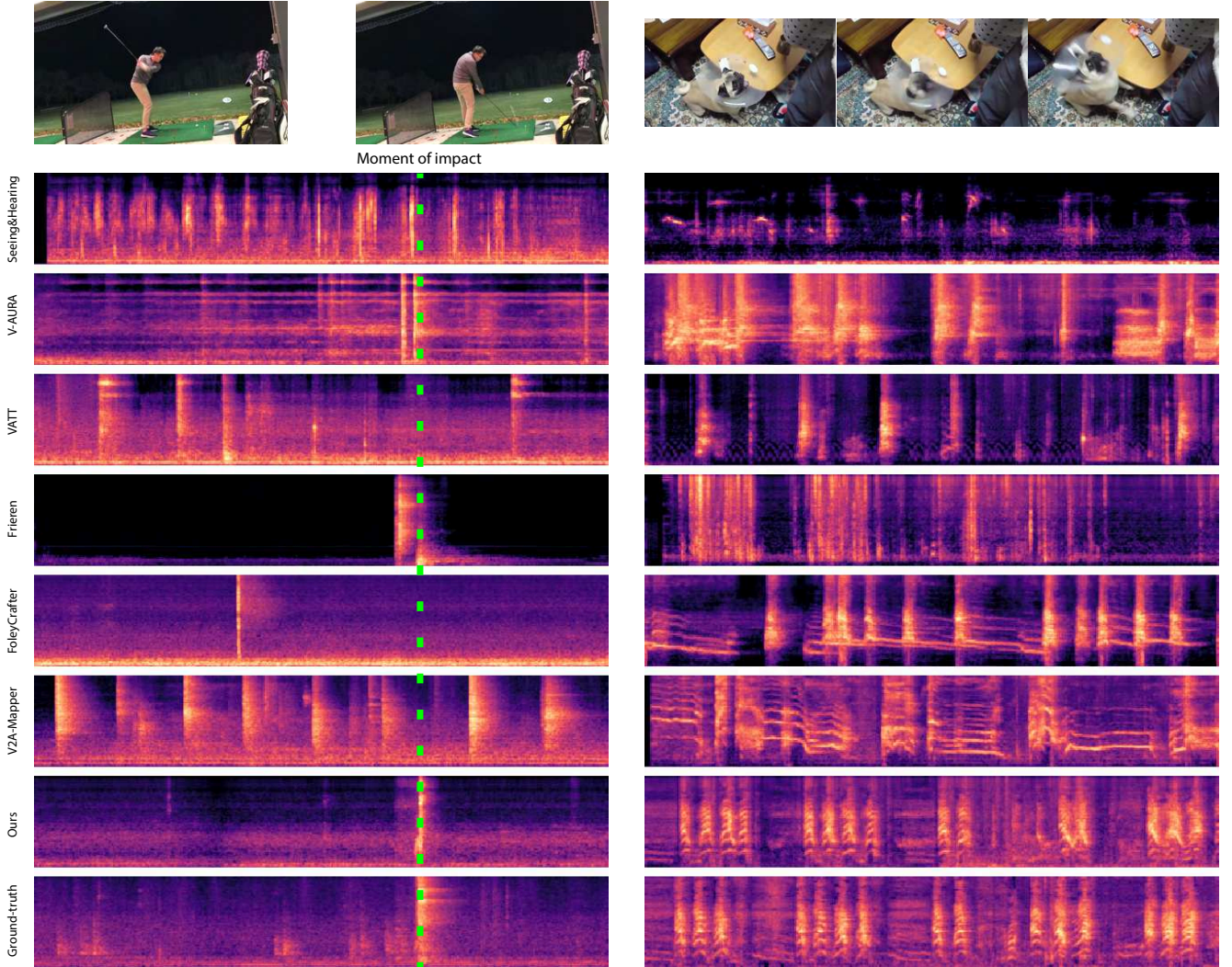

Figure A6. Left: our method can precisely capture the distinct audio event of striking a golf ball. Right: a dog barks in successive bursts. Our generation does not line up with the ground-truth as precisely due to the ambiguous nature of video-to-audio generation, but does capture the rapid bursts.

---

[3]Other uploads of this video that are *not* part of the VGGSound dataset include `1MQkMdlBezY` and `vHmRikW9axQ`.
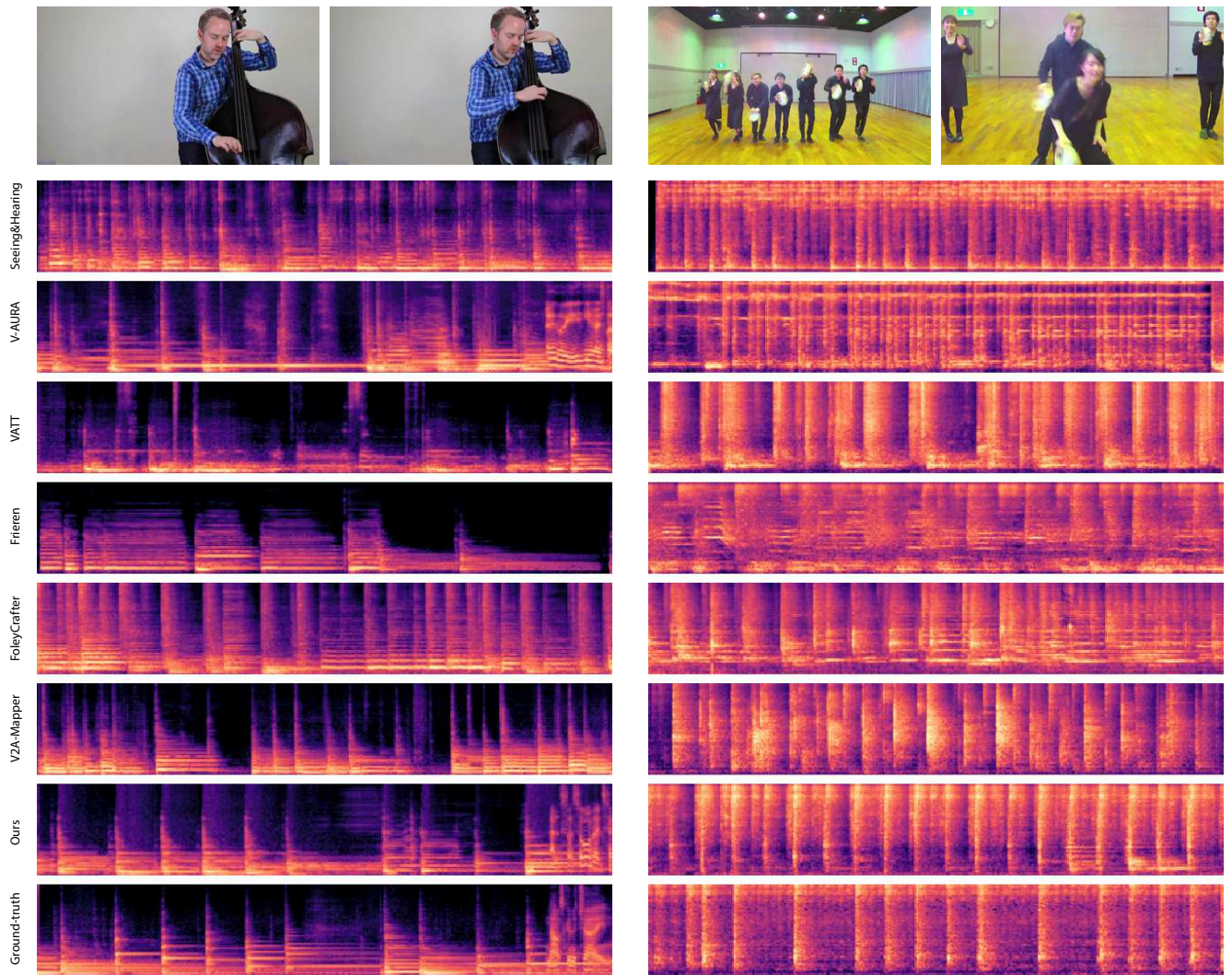
Figure A7. Left: when visible audio events (*e.g.*, when a string is played) can be clearly seen, MMAudio captures them much more precisely than existing methods. Right: in a complex scenario, MMAudio does not always generate audio aligned to the ground-truth (as common in the generative setting) but the generation is often still plausible.
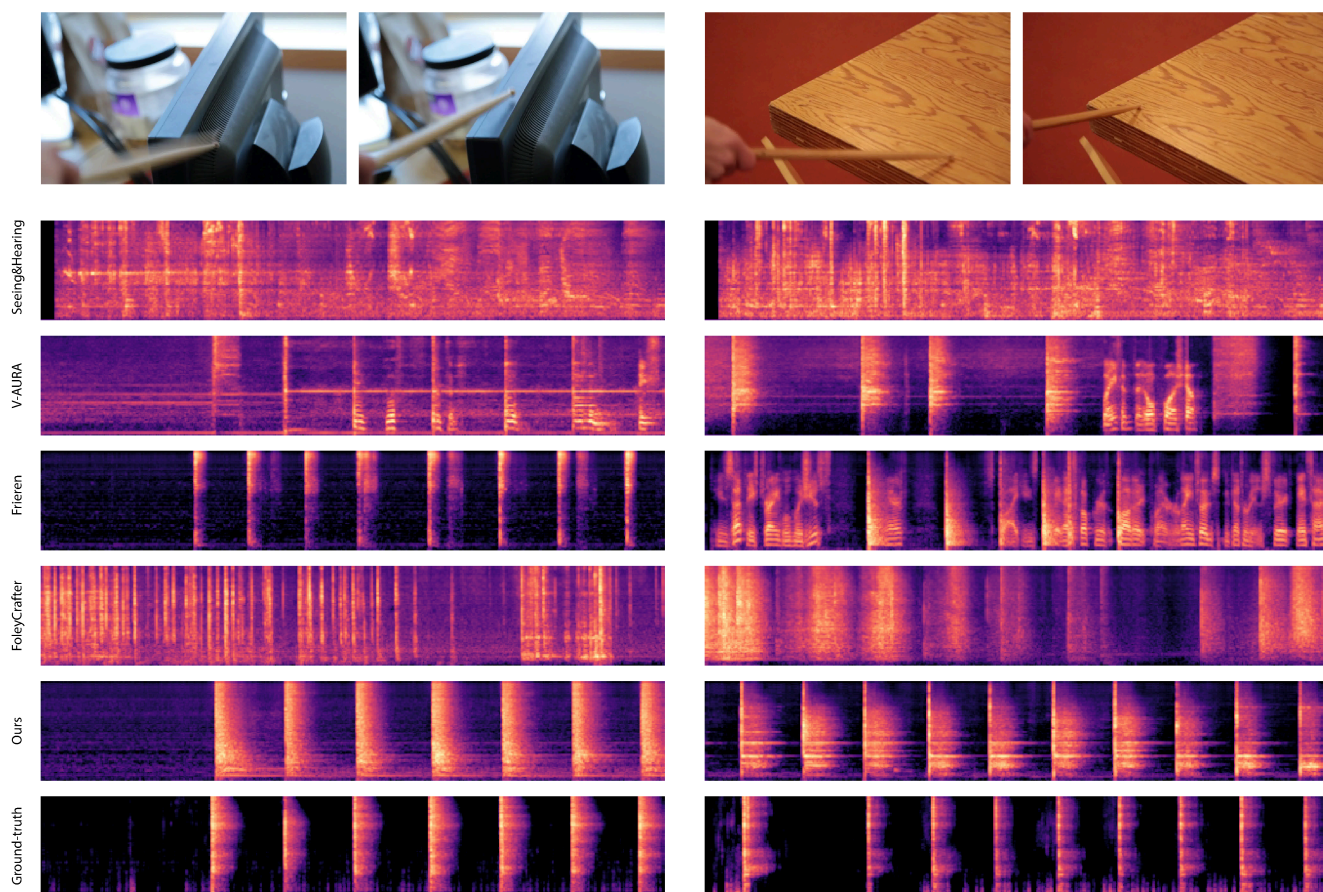
Figure A8. Comparisons of prior works with MMAudio on the Greatest Hits [48] dataset.