WavFlow: Audio Generation in Waveform Space

Feiyan Zhou[1,2] **Luyuan Wang**[1] **Shoufa Chen**[1,*] **Zhe Wang**[1]
Zhiheng Liu[1] **Yuren Cong**[1] **Xiaohui Zhang**[1] **Fanny Yang**[1] **Belinda Zeng**[1]

[1]Meta AI [2]Northeastern University

Abstract

Modern audio generation predominantly relies on latent-space compression, introducing additional complexity and potential information loss. In this work, we challenge this paradigm with WavFlow, a framework that generates high-fidelity audio directly in raw waveform space without intermediate representations. To overcome the inherent difficulties of modeling high-dimensional and low-energy signals, we reshape audio into 2D token grids through waveform patchify and introduce amplitude lifting to align signal scales, enabling stable optimization via direct x-prediction in flow matching. To capture complex semantic alignment and temporal synchronization, we leverage an automated data pipeline to curate 5 M high-quality video-text-audio triplets, allowing the model to learn fine-grained acoustic patterns from scratch. Experimental results show that WavFlow achieves competitive results on the video-to-audio benchmark VGGSound (FD_{PaSST} 59.98, IS_{PANNs} 17.40, DeSync 0.44) and the text-to-audio benchmark AudioCaps (FD_{PANNs} 10.63, IS_{PANNs} 12.62), matching or exceeding the performance of established latent-based methods. Our work demonstrates that such intermediate compression is not a prerequisite for high-quality synthesis, offering a simpler and more scalable alternative for multimodal audio generation. The code is available at https://facebookresearch.github.io/WavFlow/.

1 Introduction

Video-to-audio synthesis, often referred to as Foley-style generation [7, 64, 49, 57], aims to produce environmental and event-based soundscapes temporally and semantically aligned with the visual content. Recent state-of-the-art methods [45, 40, 64, 59, 7, 49, 36, 37, 57, 8, 51] have made rapid progress by adopting a common latent-space recipe: raw signals are first mapped into a compressed representation by a pretrained tokenizer or VAE [9, 63, 31, 13, 26, 32], then a multimodal diffusion or flow-matching transformer [18, 47, 34, 11, 43] learns their conditional distribution given visual and text features. Finally, a decoder reconstructs the waveform from these generated latents, as shown in Figure 1 (top). This paradigm has become the dominant framework for audio generation tasks.

While effective, this approach leaves a foundational question open: is latent-space compression truly necessary for audio generation? Relying on a separate, pretrained tokenizer not only increases pipeline complexity but also constrains the final synthesis quality to the reconstruction fidelity. This motivates us to investigate direct raw-waveform generation as a way to achieve strict temporal and semantic alignment while bypassing the intermediate compression layer.

Doing so, however, is non-trivial, as raw audio differs from latent representations in three fundamental ways. First, raw waveforms are extremely high-dimensional, leading to long sequences that are computationally challenging to model directly. Second, waveform amplitudes exhibit a high dynamic range while heavily concentrating near zero, yielding a poor signal-to-noise ratio during training

*Corresponding author: shoufachen@meta.com

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

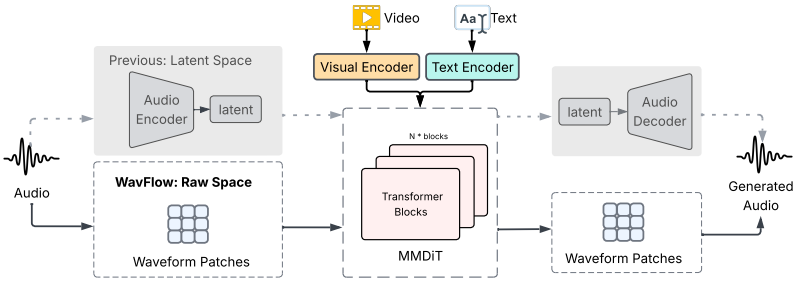


Figure 1: **Standard Latent-Space vs. WavFlow.** WavFlow eliminates the encoding-decoding steps by processing waveform patches directly in the raw space.

that makes the flow-matching objective difficult to optimize in raw space. Third, paired video-audio datasets remain relatively scarce. Even the widely-used VGGSound [3] contains only ∼200K samples (500 hours), a scale insufficient for models that operate directly on raw waveforms, which must learn complex acoustic structures, temporal dynamics, and precise cross-modal alignments end-to-end without the inductive bias provided by encoded audio priors.

In this work, we introduce WavFlow, a generative framework that performs Foley-style audio synthesis directly in raw waveform space. The architecture is deliberately simple: as illustrated in Figure 1 (bottom), we employ *waveform patchify* to reshape high-dimensional 1D waveforms into 2D token grids, and adopt x-prediction [33] under conditional flow matching as a more stable training target for raw signals. To bridge the signal intensity mismatch between raw waveforms and the unit-variance Gaussian prior, we incorporate RMS normalization and amplitude scaling, lifting the signal into a range conducive to generative modeling. Finally, to address the data scarcity in raw-space learning, we develop an automated curation pipeline to filter a large-scale media data for audio quality and event diversity, yielding approximately 5 M high-quality video-text-audio pairs [45]. We train WavFlow on this curated dataset to achieve robust video-conditioned generation and extend the model to text-only audio generation by simply zeroing out the visual conditions.

We evaluate WavFlow on the standard VT2A (VGGSound) and T2A (AudioCaps [25]) benchmarks. On VGGSound, WavFlow achieves state-of-the-art FD_{PaSST} (55.82 at 44.1 kHz and 59.98 at 16 kHz) while demonstrating competitive performance in DeSync (0.44) and IS_{PANNs} (17.40) compared to latent-based models [59, 56, 49, 7]. These results validate that raw-waveform synthesis can match or even exceed the precision and fidelity of latent-space paradigms. Furthermore, on AudioCaps, our model attains the best FD_{PANNs} (10.63) and IS_{PANNs} (12.62) reported to date, rivaling dedicated T2A systems.

In summary, our contributions to direct raw-space audio generation are three-fold: **(i) Streamlined Framework:** we introduce WavFlow, a simplified architecture that synthesizes high-fidelity audio directly in the waveform space through *waveform patchify*, x-prediction flow matching, and specialized signal preprocessing, effectively eliminating the need for audio tokenizers; **(ii) Large-scale Data Curation:** we identify that direct waveform modeling is exceptionally sensitive to data quality and scale, and thus develop an automated pipeline to harvest high-quality, large-scale supervision consisting of multi-modal VT2A samples; **(iii) Empirical Validation:** we achieve highly competitive results on the VGGSound (VT2A) and AudioCaps (T2A) benchmarks, demonstrating that end-to-end waveform generation reaches performance on par with established latent-based methods in acoustic richness, fidelity, and synchronization.

2 Related Work

2.1 Latent-Space Audio Generation

The landscape of latent-space audio generation is characterized by two main paradigms: *continuous latent modeling* and *discrete codec-based synthesis*. Models such as AudioLDM [35, 38], TANGO [15], and MMAudio [7] operate on continuous manifolds learned by audio VAEs [13].

These frameworks prioritize spectral reconstruction and often incorporate adversarial discriminators from vocoders like HiFi-GAN [26] or BigVGAN [32] to refine the decoded waveforms. Conversely, systems like AudioGen [30] and V-AURA [54] leverage discrete neural audio codecs [9, 31], where generative modeling is performed over quantized tokens.

While this paradigm bypasses the high-dimensionality of raw audio, it imposes a rigid performance ceiling: the output quality is strictly upper-bounded by the reconstruction fidelity of the pretrained backbone. Critical details, such as high-frequency transients and fine-grained phase information, are often compromised during latent bottlenecking and remain irrecoverable through post-processing. This inherent lossiness motivates the exploration of modeling the audio distribution directly in its native, uncompressed space.

2.2 Raw-Space Generative Modeling

Early raw-waveform generation methods primarily focused on neural vocoding conditioned on mel-spectrograms, such as WaveRNN [24] and WaveGrad [4], while models such as WaveNet [53] and DiffWave [28] also explored unconditional or class-conditional generation. However, the extremely long sequences of raw audio make direct modeling computationally challenging due to the cost of sample-level generation and the difficulty of capturing long-range dependencies. Video-to-audio generation further requires not only high-quality audio synthesis but also semantic consistency and temporal synchronization with visual and text features. Consequently, existing multimodal audio generation methods predominantly rely on compressed representations, such as mel-spectrograms or compressed latents, to shorten audio sequences and make long-range cross-modal alignment more tractable.

In the image domain, while early CNNs relied on specialized noise schedules [6, 19], Transformers often suffer from catastrophic degradation in high-dimensional raw space [33]. To mitigate this, frameworks like SiD2 [20], PixelFlow [5], and PixNerd [58] resort to hierarchical designs or specialized heads. Most recently, JiT [33] succeeds by revisiting the manifold hypothesis [2, 55]: since clean data lies on a low-dimensional manifold while noise or velocity spans the entire high-dimensional space, x-prediction is fundamentally easier to learn than noise or v-prediction. This allows the network to focus on recovering the low-dimensional data structure rather than modeling full-space noise. These advances in vision suggest that the raw-space paradigm, if properly adapted, can overcome the scalability issues previously encountered in audio modeling.

2.3 Multimodal DiT for Audio Generation

Video-to-audio (VT2A) generation requires precise temporal synchronization and semantic consistency, leading recent systems to adopt Multimodal Diffusion Transformers (MMDiT) [11] for joint modeling of audio, video, and text. The evolution of these architectures reflects a progression from efficient latent-space synthesis in Frieren [59] to the unified joint-attention paradigm of MMAudio [7], which significantly improved cross-modal alignment. More recently, industrial-scale models [49, 57] have pushed performance limits by scaling architectures to dozens of layers and training on massive datasets, such as the 100k hours of video-text-audio samples used in HunyuanVideo-Foley, while utilizing universal latent codecs and enhanced visual modules for high-fidelity synthesis.

Despite these advancements, existing systems remain confined to the compressed latent space. Our work overcomes this by adopting an MMDiT-based architecture that eliminates the latent stage entirely, enabling high-fidelity synthesis directly on raw waveforms.

3 Method

The architecture of WavFlow is built on a MultiModal Diffusion Transformer (MMDiT) [11, 7] backbone, without relying on an audio encoder or decoder. Given a multimodal conditioning signal c (video and text), the model employs conditional flow matching to generate the raw waveform $x \in \mathbb{R}^T$ directly in observation space. To handle the high dimensionality of raw audio, we apply *waveform patchify* to reshape the signal into patches for transformer processing and adopt an x*-prediction* strategy to promote stable training.

3.1 Flow Matching in Waveform Space

Conditional Flow Matching. We formulate waveform generation using conditional flow matching [34, 39, 1]. Let $x_0 \sim \mathcal{N}(0, I)$ denote Gaussian noise and x_1 denote a clean waveform. A continuous interpolation between noise and data is defined as:

$$x_t = (1-t)x_0 + tx_1, \quad t \in [0,1], \tag{1}$$

with the corresponding target velocity $v^*(x_t, t) = x_1 - x_0$. The goal is to learn a velocity field $v_\theta(x_t, t, c)$ that transports noise to data along this path. While latent-space methods model these flows in a compressed representation, we perform this mapping directly in the waveform space, solving the ODE $\frac{dx_t}{dt} = v_\theta(x_t, t, c)$ during inference to obtain the final waveform.

Prediction Parameterization and Loss. We adopt x-prediction [33, 48], where network predicts the clean signal:

$$\hat{x}_1 = f_\theta(x_t, t, c). \tag{2}$$

The velocity is recovered as $v_\theta = (\hat{x}_1 - x_t)/(1-t)$. Our default configuration optimizes this x-prediction through a v-loss:

$$\mathcal{L} = \mathbb{E}_{x_0, x_1, t}\left[\left\| \frac{\hat{x}_1 - x_t}{1-t} - \frac{x_1 - x_t}{1-t} \right\|_2^2\right]. \tag{3}$$

This combination ensures that while the network focuses on recovering the data manifold [2], the objective remains anchored to the flow-matching velocity field. We validate this design choice through ablation experiments in Section 4.5.

3.2 Model Architecture

Audio Preprocessing. Raw waveforms typically exhibit a sharp, zero-centered distribution with low energy (average RMS often below 0.2), making them easily masked by noise during training. To mitigate this, we apply *amplitude lifting* by combining RMS normalization and global scaling. Specifically, after converting audio $x \in \mathbb{R}^T$ to mono, the lifted waveform x_{lift} is computed as:

$$x_{lift} = s_a \cdot \text{clamp}\left(\frac{r_\star}{rms(x)}x, -1, 1\right), \tag{4}$$

where we empirically set $r_\star = 0.33$ and $s_a = 3.0$ to align the signal scale with the Gaussian noise prior. During inference, the output is rescaled by $1/s_a$ and normalized to -23 LUFS [12] to ensure perceptually comfortable playback. A visualization of this shift is provided in Appendix B.

Waveform Patchify. After preprocessing, raw audio is reshaped into a $C \times D$ grid via *waveform patchify* (Figure 2), where each row serves as a token analogous to the image patchify in ViT [10]. The patch dimension D represents the samples per token, defining its temporal granularity. This involves a fundamental trade-off: while smaller D eases the learning of intricate acoustic details, it increases computational complexity ($O(C^2)$); conversely, larger D improves efficiency but increases per-token information density. Ablation studies (Section 4.5) reveal that increasing the data scale effectively compensates for this modeling difficulty, allowing the network to extract sufficient information even from wider patches.

Our investigations identify D=200 as the saturation point where performance stabilizes. At 16 kHz, this yields C=640 tokens for an 8 s clip, resulting in a 12.5 ms granularity—well below the ∼25 ms human auditory resolution threshold [44]. To maintain architectural consistency, we keep D=200 for 44.1 kHz signals (C=1,764), resulting in an even finer temporal granularity that further reinforces the model's high-fidelity representation. After generation, the grid is reshaped back to a 1D waveform (*waveform unpatchify*). This process is entirely parameter-free and lossless, requiring no learned decoders or neural vocoders.

Multimodal DiT Architecture. As shown in Figure 2, we adopt the Multimodal Diffusion Transformer (MMDiT) [11, 7] as our backbone, consisting of L_{joint} joint blocks for multimodal fusion followed by L_{fused} audio-only blocks for waveform refinement. Three input streams enter the joint attention sequence: audio waveform tokens, visual features from a frozen CLIP [46] encoder, and

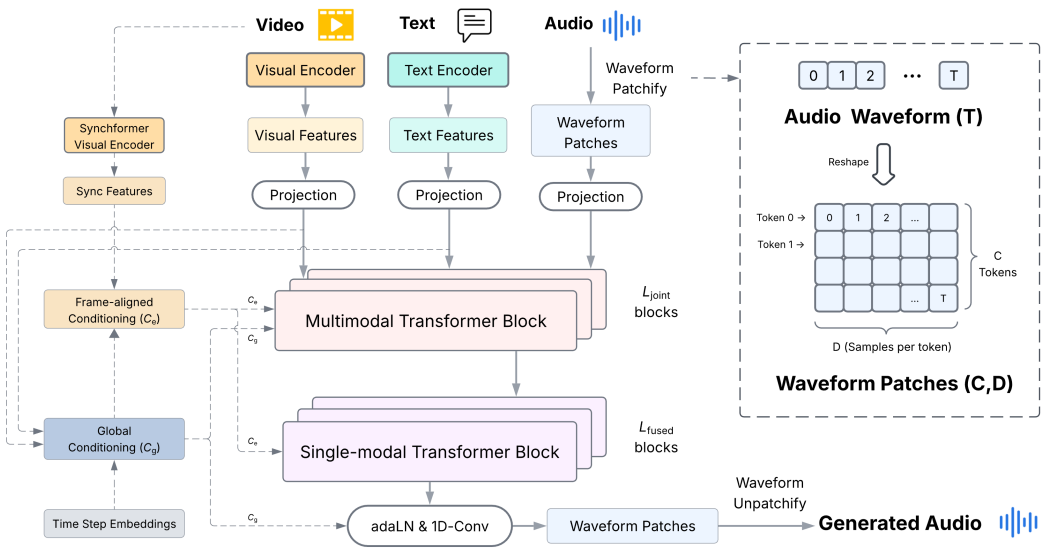


Figure 2: **The WavFlow Architecture.** Raw audio is represented as a 2D patch grid and processed through a series of joint and fused transformer blocks. The model leverages multimodal conditioning ($\mathbf{c}_g$ and $\mathbf{c}_e$) for precise semantic and temporal control during the flow-matching process. *Note:* Solid arrows denote the main feature flow, while dashed arrows indicate auxiliary conditioning paths, including synchronization feature extraction and conditioning injection.

text embeddings from a CLIP text encoder. Audio waveform tokens and visual CLIP features are projected into a shared hidden dimension d via convolutional input blocks, whereas text features use a linear projection.

The model employs dual-level conditioning to capture semantic ("what") and temporal ("when") cues. A global condition $\mathbf{c}_g$ is formed by summing mean-pooled visual and text features with the flow-matching timestep embedding, providing semantic guidance. To capture precise temporal cues, a frozen Synchformer [23] extracts synchronization features from video, augmented with learnable per-segment positional embeddings. A frame-aligned condition $\mathbf{c}_e \in \mathbb{R}^{C\times d}$ is obtained by adding $\mathbf{c}_g$ to these synchronization features (upsampled to length C via nearest interpolation), providing frame-level alignment. These conditions are injected into the transformer blocks through AdaLN modulation [43], ensuring robust audio-visual correlation and semantic grounding in the raw waveform space.

Following the transformer blocks, a final output block projects the features from d back to D samples per token via AdaLN and a 1D convolution (kernel size 7). The resulting grid is then reconstructed into a 1D waveform via *waveform unpatchify*. We instantiate two variants based on scale: **WavFlow-M** (L_{joint}=4, L_{fused}=8, ∼624M parameters) and **WavFlow-L** (L_{joint}=7, L_{fused}=14, ∼1.03B parameters), both sharing hidden dimension d=896 and 14 attention heads.

Positional Encoding. We apply RoPE [50] on the queries and keys to inject relative position information into joint attention; text tokens are excluded since captions encode unordered semantics rather than temporal structure. Because the audio and visual CLIP streams run at different frame rates, applying identical base frequencies would map equivalent moments in the two streams to mismatched rotary angles. We therefore multiply the visual stream's RoPE base frequency by the audio-to-visual ratio C/N_{clip} (e.g., 10× for C=640, N_{clip}=64), so that tokens at the same relative temporal position receive matching rotary phases.

3.3 Training and Inference

Classifier-Free Guidance. During training, we independently replace the visual conditioning (visual CLIP and Synchformer features jointly) and text features with learned null embeddings with a 10% probability. This strategy not only enables classifier-free guidance during inference but

also allows WavFlow to support both video-to-audio (VT2A) and text-to-audio (T2A) generation within the same architecture. For T2A generation, we replace the visual conditions with learned null embeddings, reducing the conditioning signal to text alone without any architectural modification.

Inference. Generation begins with Gaussian noise sampled in the waveform token space. We solve the learned ODE using an Euler solver with classifier-free guidance (CFG):

$$\hat{f}_{\text{CFG}} = f_\theta(x_t, t, \varnothing) + w\left(f_\theta(x_t, t, c) - f_\theta(x_t, t, \varnothing)\right), \tag{5}$$

where w is the guidance scale and $\varnothing$ denotes the null conditions. After the integration, the generated token grid is converted back to a 1D raw waveform via *waveform unpatchify*, requiring no additional learned decoder.

4 Experiments

4.1 Dataset

Training generative models directly in raw waveform space imposes significant demands on data scale and quality. In latent-space methods, a pretrained audio encoder leverages extensive audio data (e.g., $\sim$20 K hours in [13]) to encode rich acoustic priors, effectively mapping complex waveforms into a compressed manifold that simplifies generative learning. By contrast, waveform-space models must learn intricate acoustic patterns and cross-modal dependencies from scratch, necessitating access to large-scale, high-fidelity audio-visual datasets.

However, public VT2A resources like VGGSound [3] are limited ($\sim$500 hours) compared to T2A datasets like WavCaps ($\sim$7 K hours) [42]. While augmenting sparse VT2A data with T2A samples [7], we find this strategy ineffective in the raw waveform domain. Our ablation study (Appendix C) confirms that this fails to improve and even degrades performance; for instance, adding T2A data results in a DeSync of 0.57, worse than the 0.52 achieved with VGGSound alone. This underscores that audio-only signals cannot substitute for the grounded, high-temporal-resolution supervision required for end-to-end synthesis. Consequently, we curate a large-scale proprietary media dataset, primarily sourced from MovieGen [45], via an automated unified pipeline, constructing robust training sets for both tasks.

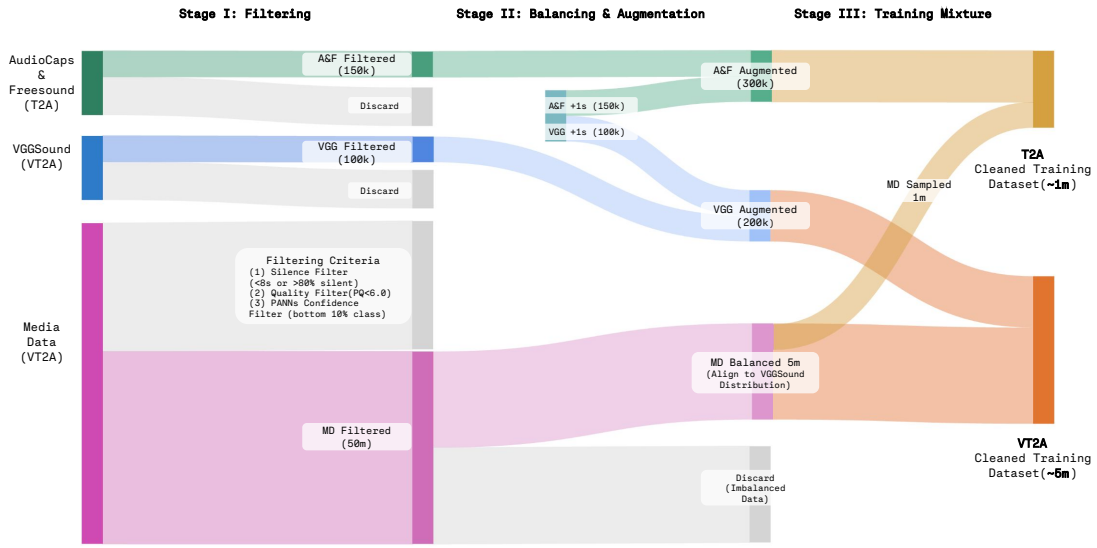


Figure 3: Overview of multi-stage data curation from raw datasets to final training mixtures.

Data Curation Pipeline. As illustrated in Figure 3, our pipeline unifies VT2A and T2A samples through three stages. For open-source data, we utilize VGGSound [3] alongside AudioCaps [25] and

Freesound [14]. Initially, we apply multi-stage filtering across all sources: extracting 8 s segments and discarding samples with $> 80\%$ silence, low aesthetic scores (PQ < 6.0 via audiobox-aesthetics [52]), or low classification confidence (bottom 10% via PANNs [27]). This process yields roughly 50 M filtered media clips, 100 K VGGSound samples, and 150 K high-quality T2A samples.

Subsequently, we balance and augment the filtered data. The curated media clips are category-aligned with VGGSound to form a balanced pool of 5 M samples. For the smaller VGGSound and public T2A sets, we apply temporal augmentation by extracting two overlapping 8 s chunks starting at 0 s and 1 s, respectively, to double their size to 200 K and 300 K. Ultimately, these sources are merged into our final mixtures: a **VT2A set** combining the 5 M balanced media pool with augmented VGGSound, and a **T2A set** mixing the 300 K augmented public T2A samples with 1 M clips randomly sampled from the same high-quality media corpus. This ensures a consistent data distribution across both tasks.

4.2 Experimental Setup

Training. Training configurations are detailed in Table 7. Models are trained on 8 s clips using flow-matching with x-prediction and v-loss, sampling timesteps from a logit-normal distribution [11]. Raw audio is tokenized via *waveform patchify* (D=200), resulting in sequence lengths of 640 tokens for 16 kHz and 1,764 tokens for 44.1 kHz. We instantiate two 16 kHz variants (**WavFlow-M-16kHz**, **WavFlow-L-16kHz**) and one 44.1 kHz model (**WavFlow-L-44.1kHz**).

The 16 kHz models are trained from scratch for 400 epochs, a convergence point identified by monitoring validation metrics (see Figure 5). Our primary VT2A model utilizes the $\sim$5M mixture with a global batch size of 10,752, while the T2A model uses the $\sim$1M mixture with a batch size of 8,192. For high-fidelity synthesis, the 44.1 kHz Large model is fine-tuned (SFT) [62] from the converged 16 kHz checkpoint. All stages use the AdamW optimizer with a constant learning rate of 1×10^{-4} (1×10^{-5} for SFT) and a 20-epoch linear warmup.

Evaluation Metrics. We evaluate VT2A on the VGGSound test set (15 K videos) and T2A on AudioCaps (4.8 K samples). Inference uses an ODE solver with 50 steps and a CFG scale of 4.5 (see Appendix F). Performance is assessed across three dimensions: **(i) Acoustic Quality:** using Fréchet Distance (FD), KL divergence, and Inception Score (IS) via features from PANNs [27] and PaSST [29] classifiers; **(ii) Semantic Alignment:** employing ImageBind (IB) [16] for audio-visual correspondence and CLAP [60] for text-audio consistency; **(iii) Temporal Synchronization:** utilizing the DeSync metric [23] to quantify timing accuracy between visual events and audio onsets.

4.3 Comparison with State-of-the-Art Methods

Video-to-Audio Generation. Results on the VGGSound test set are summarized in Table 1. Even at 16 kHz with the medium backbone, WavFlow outperforms recent latent-based systems—including Frieren [59], V2A-Mapper [56] and HunyuanVideo-Foley [49]—across all acoustic quality and synchronization metrics (e.g., FD_{PANNs}: 6.37, IS_{PANNs}: 17.24, and DeSync: 0.47), approaching the state-of-the-art MMAudio-L-44.1kHz [7] baseline even without a pretrained codec.

Scaling to WavFlow-L-16k yields consistent improvements, matching MMAudio on the two perceptual indicators most critical for audio-visual generation: IS_{PANNs} (**17.40**) and DeSync (**0.44**). This demonstrates that raw-waveform modeling can attain the same level of temporal precision and acoustic richness as latent-space methods. Furthermore, WavFlow-L-44k, fine-tuned from the 16 kHz Large checkpoint, pushes distributional fidelity even further. It achieves the **best FD_{PaSST} (55.82)** across all compared methods and the lowest FD_{PANNs} (5.25) among our variants, while maintaining high synchronization (DeSync: 0.46). Collectively, these results confirm that with high-quality, large-scale data, end-to-end waveform synthesis can eliminate the need for intermediate latent representations without sacrificing performance.

Text-to-Audio Generation. To further validate the versatility of our VAE-free approach, we evaluate WavFlow-M-16kHz on the AudioCaps text-to-audio benchmark. As shown in Table 2, our system achieves competitive acoustic quality across all compared methods. Specifically, it attains the lowest FD_{PANNs} (**10.63**) and the highest IS_{PANNs} (**12.62**), outperforming dedicated latent-space models like TANGO 2 [41] and the previous state-of-the-art MMAudio. These results demonstrate that the intricate acoustic patterns learned directly in raw waveform space generalize effectively

Table 1: Comparison with state-of-the-art methods on VGGSound-Test for video-to-audio generation. Lower is better for FD, KL and DeSync; higher is better for IS, IB and CLAP. The best and second-best results are highlighted in bold and underlined, respectively.

Method	FD_{PANNs} ↓	FD_{PaSST} ↓	KL_{PANNs} ↓	IS_{PANNs} ↑	IB↑	DeSync↓	CLAP↑	Params
Frieren [59]†	11.45	106.10	2.73	12.25	0.23	0.85	0.11	159M
V2A-Mapper [56]†	8.40	84.57	2.69	12.47	0.23	1.23	0.11	229M
HunyuanVideo-Foley [49]*	10.53	97.85	2.02	14.99	<u>0.32</u>	0.54	**0.23**	–
MMAudio-L-44.1kHz [7]†	**4.72**	60.60	**1.65**	**17.40**	**0.33**	**0.44**	<u>0.22</u>	1.03B
WavFlow-M-16kHz	6.37	62.64	1.68	<u>17.24</u>	0.30	0.47	0.21	624M
WavFlow-L-16kHz	5.86	<u>59.98</u>	<u>1.66</u>	**17.40**	0.31	**0.44**	<u>0.22</u>	1.03B
WavFlow-L-44.1kHz	<u>5.25</u>	**55.82**	1.73	15.05	0.31	<u>0.46</u>	0.19	1.03B

All methods are evaluated on the same VGGSound test split from the MMAudio [7] benchmark, utilizing original videos and native class labels as captions to ensure a fair comparison. Due to the difference in semantic granularity (sparse labels vs. dense captions), we exclude direct comparisons with models relying on LLM-refined captions. †: results taken from the MMAudio paper. *: reproduced by using their open-source checkpoints on the same test set.

across different input modalities. More importantly, this cross-task success reinforces our primary conclusion: high-fidelity synthesis can be achieved without intermediate latent representations or task-specific vocoders.

Table 2: Performance comparison for text-to-audio (T2A) generation on the AudioCaps test set.

Method	Params	FD_{PANNs} ↓	FD_{VGG} ↓	IS_{PANNs} ↑	CLAP↑
AudioLDM 2-L [38]	712M	32.50	5.11	8.54	0.21
TANGO [15]	866M	26.13	1.87	8.23	0.19
TANGO 2 [41]	866M	19.77	2.74	8.45	0.26
Make-An-Audio [22]	453M	27.93	2.59	7.44	0.21
Make-An-Audio 2 [21]	937M	15.34	<u>1.27</u>	9.58	0.25
GenAU-Large [17]	1.25B	16.51	**1.21**	11.75	<u>0.29</u>
MMAudio-L-44.1kHz [7]	1.03B	<u>15.04</u>	4.03	<u>12.08</u>	**0.35**
WavFlow-M-16kHz	624M	**10.63**	1.58	**12.62**	0.24

Baseline results are quoted from [7] for a fair comparison, as we adopt the same evaluation protocol and test splits.

4.4 Controlled Comparison with Latent-Space Generation

To further evaluate the effectiveness of raw-waveform generation, we conduct a controlled comparison between WavFlow and the latent-space model MMAudio [7] across three data scales: VGGSound (200K samples, 500 hours), and 1M (2.2K hours) and 3M (6.7K hours) subsets of our VT2A dataset. Both models are trained on identical data at each scale.

We adopt the Medium 16 kHz configuration for both models, with comparable generator parameters (624M for WavFlow and 622M for MMAudio), as shown in Table 3. MMAudio additionally relies on a pretrained audio VAE (trained on approximately 19.5K hours of audio) and a BigVGAN vocoder. We keep these components frozen and retrain only its generator, while WavFlow is trained from scratch without pretrained audio codecs.

Both models share the same optimizer, learning rate, and training strategy, with nearly identical global batch sizes at each data scale (Figure 5). We use EMA checkpoints for evaluation and stop training once validation metrics plateau. MMAudio converges faster, benefiting from its compressed latent representations and pretrained audio priors. At the 3M scale, training WavFlow requires approximately 6,200 GPU-hours, compared to 1,980 GPU-hours for MMAudio, excluding the pretraining costs of its VAE and vocoder.

Results and Analysis. As shown in Table 3, MMAudio outperforms WavFlow on most metrics at smaller data scales, potentially benefiting from its pretrained VAE. However, as the training data scales from 1M to 3M, MMAudio largely saturates on several metrics, whereas WavFlow continues to improve. At 3M samples, WavFlow surpasses MMAudio in FD_{PANNs} (6.08 vs. 6.26), matches its ImageBind score (0.30), and remains competitive on the remaining metrics. These results

Table 3: **Controlled comparison between MMAudio-M-16kHz and WavFlow-M-16kHz** across three matched generator-training data scales on VGGSound-Test. Lower is better for FD, KL, and DeSync; higher is better for IS, IB, and CLAP. Bold indicates the better result at each scale.

Data Scale	Model	FD_{PaSST}↓	FD_{PANNs}↓	KL_{PANNs}↓	IS_{PANNs}↑	IB↑	DeSync↓	CLAP↑	Codec Params	Gen. Params
200K (VGGSound)	MMAudio	**65.87**	**6.77**	**1.66**	11.74	**0.27**	**0.52**	**0.18**	284M	622M
	WavFlow	90.24	8.63	1.83	**13.45**	0.25	0.59	**0.18**	–	624M
1M	MMAudio	**56.61**	**6.12**	**1.61**	**16.35**	**0.30**	**0.48**	**0.21**	284M	622M
	WavFlow	63.05	6.21	1.76	15.58	0.28	0.50	0.20	–	624M
3M	MMAudio	**56.60**	6.26	**1.60**	**17.32**	**0.30**	**0.47**	**0.22**	284M	622M
	WavFlow	60.75	**6.08**	1.74	16.56	**0.30**	0.48	0.20	–	624M

Codec parameters include all pretrained audio VAE/vocoder components outside the generator. MMAudio uses a 172M-parameter VAE and a 112M-parameter BigVGAN-v2-16k vocoder. WavFlow requires neither component.

suggest that raw-waveform generation can scale effectively with training data and achieve competitive performance against latent-space methods without relying on pretrained audio codecs.

4.5 Ablation Studies

To further investigate the key design choices of WavFlow, we conduct ablation studies on waveform patchify granularity, prediction target, and audio preprocessing. Unless otherwise specified, all experiments use WavFlow-M-16k trained on the 1M subset of our VT2A dataset, following the default configurations in Appendix A.

Patchify Granularity Analysis. Figure 4 and Table 4 reveal an interplay between the patch dimension D and data scale. Reducing D refines the temporal resolution of raw waveform tokens by increasing the token count C. In the low-data regime (200K VGGSound), this finer granularity is the most effective lever for quality: shrinking D from 512 to 200 drives a dramatic improvement in FD_{PaSST} (136.45 → 90.24) and DeSync (0.66 → 0.59). This suggests that high-resolution tokenization is essential for capturing intricate acoustic structures when training samples are sparse.

Conversely, increasing data scale can partially compensate for the modeling difficulty of a larger D. For a fixed D=512, expanding the dataset from 200K to 1M significantly improves FD_{PaSST} (136.45 → 81.81). However, at 3M samples, this coarse configuration suffers from a *capacity bottleneck*, where performance degrades to 89.51, indicating that large D values eventually fail to represent the complex information inherent in larger datasets. In contrast, with sufficient granularity ($D \leq 256$), increasing data scale consistently improves results. Notably, at 3M samples, the performance gain from further reducing D from 200 to 160 is marginal (60.75 vs. 59.43), identifying D=200 as the *saturation point* where computational efficiency and synthesis quality are optimally balanced. Based on these findings, we adopt D=200 (C=640) as our default configuration.

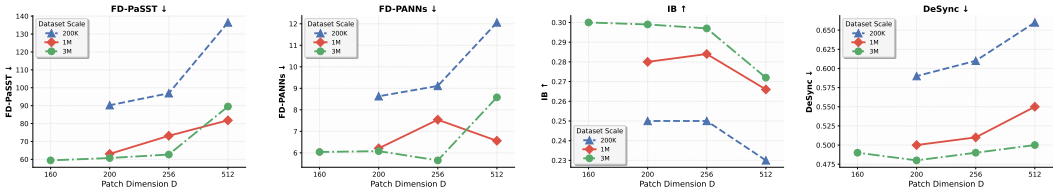


Figure 4: Patchify granularity vs. data scale across key metrics. Note the performance degradation at D=512 when scaling to 3M, indicating a capacity bottleneck for low-granularity tokens.

Prediction Target and Loss Formulation. Table 5 compares prediction targets (x vs. v) and loss formulations. The results show that x-prediction consistently outperforms v-prediction across most metrics, likely because it provides a more structured objective that better conforms to the manifold hypothesis [2, 33]. It is worth noting that both targets remain viable for direct raw-waveform generation within our framework. This universal effectiveness can be attributed to the low per-token dimensionality (D=200) relative to the model's hidden dimension (d=896), which provides sufficient capacity to represent either target without a latent bottleneck.

Among the x-prediction variants, x-pred with v-loss achieves the best FD_{PaSST} (63.05) and IS_{PANNs} (15.58), whereas x-pred with x-loss shows slight advantages in FD_{PANNs} (4.86). Given that v-

loss superiorly balances generative diversity (IS_{PANNs}) and high-frequency fidelity—as reflected by FD_{PaSST}, which leverages the 32 kHz-aware PaSST to capture a wider frequency range than PANNs (operates on 16 kHz) —we adopt x-prediction with v-loss as our default.

Table 4: Impact of patchify granularity ($C \times D$) on waveform synthesis across varying data scales. Results demonstrate the trade-off between temporal resolution and data diversity.

Data Scale	$C \times D$	FD_{PaSST} ↓	FD_{PANNs} ↓	KL_{PANNs} ↓	IS_{PANNs} ↑	IB↑	DeSync↓
200K (VGGSound)	250 × 512	136.45	12.06	1.88	10.86	0.23	0.66
	500 × 256	96.91	9.11	1.84	12.56	0.25	0.61
	640 × 200	90.24	8.63	1.83	13.45	0.25	0.59
1M	250 × 512	81.81	6.56	1.78	13.66	0.27	0.55
	500 × 256	73.16	7.54	1.91	16.18	0.28	0.51
	640 × 200	63.05	6.21	1.76	15.58	0.28	0.50
3M	250 × 512	89.51	8.58	1.96	15.60	0.27	0.50
	500 × 256	62.69	5.65	1.68	15.80	0.30	0.49
	640 × 200	60.75	6.08	1.74	16.56	0.30	0.48
	800 × 160	59.43	6.04	1.73	16.43	0.30	0.49

Table 5: Ablation on prediction target (x vs. v) and loss formulation (x vs. v).

Setting	FD_{PaSST} ↓	FD_{PANNs} ↓	KL_{PANNs} ↓	IS_{PANNs} ↑	IB↑	DeSync↓
v-pred + v-loss	77.19	6.38	1.75	13.48	0.27	0.53
x-pred + x-loss	72.70	4.86	1.72	13.99	0.29	0.50
x-pred + v-loss	**63.05**	6.21	1.76	**15.58**	0.28	**0.50**

Raw-waveform Preprocessing. Table 6 validates our preprocessing pipeline, showing that both RMS normalization and amplitude scaling are essential for signal quality. Without scaling (1.0×), omitting RMS normalization severely degrades FD_{PaSST} (65.83 → 81.26) and DeSync (0.49 → 0.57). At 3.0× scale, the enhanced signal strength naturally narrows the performance gap, yet the combination of both techniques still yields the best overall results in diversity and fidelity (e.g., IS_{PANNs} of 15.58). We hypothesize that proper scaling helps the raw signal better align with the Gaussian prior during flow matching, whereas insufficient scaling (1.0×) leads to suboptimal optimization. Consequently, we adopt RMS normalization with a 3.0× scale as our default.

Table 6: Impact of RMS normalization and constant amplitude scaling on raw waveform modeling.

Category	Setting	FD_{PaSST}↓	FD_{PANNs}↓	KL_{PANNs} ↓	IS_{PANNs} ↑	IB↑	DeSync↓
RMS Norm. (at 1.0× scale)	w/	65.83	6.03	1.73	13.32	0.28	0.49
	w/o	81.26	8.69	1.91	11.64	0.24	0.57
RMS Norm. (at 3.0× scale)	w	63.05	6.21	1.76	15.58	0.28	0.50
	w/o	64.23	6.93	1.79	13.84	0.26	0.52

5 Conclusion and Limitations

We present WavFlow, a flow-matching framework for high-fidelity audio generation directly in raw-waveform space. By combining waveform patchify, x-prediction, and specific signal scaling with extensive data, WavFlow effectively stabilizes training and models complex raw signals. It achieves SOTA performance on VT2A and T2A benchmarks, outperforming existing latent-based systems.

Limitations. WavFlow currently lacks explicit speech or singing synthesis, as generated vocalizations do not constitute meaningful language. Extending to these domains requires finer linguistic granularity and larger speech datasets. By incorporating larger-scale corpora and fine-grained linguistic captions, this framework could be extended to jointly model environmental sounds and human speech, offering a robust and efficient alternative for future generative research.

Acknowledgments and Disclosure of Funding

This work was supported by Meta. The authors declare no additional competing interests.

References

[1] M. S. Albergo and E. Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023.

[2] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning*. MIT Press, 2006.

[3] H. Chen, W. Xie, A. Vedaldi, and A. Zisserman. Vggsound: A large-scale audio-visual dataset. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2020.

[4] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan. WaveGrad: Estimating gradients for waveform generation, 2020.

[5] S. Chen, C. Ge, S. Zhang, P. Sun, and P. Luo. Pixelflow: Pixel-space generative models with flow. *arXiv preprint arXiv:2504.07963*, 2025.

[6] T. Chen. On the importance of noise scheduling for diffusion models. *arXiv preprint arXiv:2301.10972*, 2023.

[7] H. K. Cheng, M. Ishii, A. Hayakawa, T. Shibuya, A. Schwing, and Y. Mitsufuji. Mmaudio: Taming multimodal joint training for high-quality video-to-audio synthesis. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 28901–28911, 2025.

[8] Y. Dai, Z. Chen, Y. Jiang, B. Gao, Q. Ke, J. Zhu, and J. Cai. Omni2sound: Towards unified video-text-to-audio generation, 2026.

[9] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi. High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438*, 2022.

[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[11] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[12] European Broadcasting Union. EBU R 128: Loudness normalisation and permitted maximum level of audio signals. Technical report, European Broadcasting Union, 2020.

[13] Z. Evans, C. Carr, J. Taylor, S. H. Hawley, and J. Pons. Fast timing-conditioned latent audio diffusion. In *Forty-first International Conference on Machine Learning*, 2024.

[14] E. Fonseca, J. Pons, X. Favory, F. Font, D. Bogdanov, A. Ferraro, S. Oramas, A. Porter, and X. Serra. Freesound datasets: A platform for the creation of open audio datasets. In *ISMIR*, pages 486–493, 2017.

[15] D. Ghosal, N. Majumder, A. Mehrish, and S. Poria. Text-to-audio generation using instruction guided latent diffusion model. In *Proceedings of the 31st ACM international conference on multimedia*, pages 3590–3598, 2023.

[16] R. Girdhar, A. El-Nouby, Z. Liu, M. Singh, K. V. Alwala, A. Joulin, and I. Misra. Imagebind: One embedding space to bind them all. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15180–15190, 2023.

[17] M. Haji-Ali, W. Menapace, A. Siarohin, G. Balakrishnan, and V. Ordonez. Taming data and transformers for audio generation. *International Journal of Computer Vision*, 134(3):87, 2026.

[18] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[19] E. Hoogeboom, J. Heek, and T. Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *International Conference on Machine Learning*, pages 13213–13232. PMLR, 2023.

[20] E. Hoogeboom, T. Mensink, J. Heek, K. Lamerigts, R. Gao, and T. Salimans. Simpler diffusion (sid2): 1.5 fid on imagenet512 with pixel-space diffusion, 2025.

[21] J. Huang, Y. Ren, R. Huang, D. Yang, Z. Ye, C. Zhang, J. Liu, X. Yin, Z. Ma, and Z. Zhao. Make-an-audio 2: Temporal-enhanced text-to-audio generation. *arXiv preprint arXiv:2305.18474*, 2023.

[22] R. Huang, J. Huang, D. Yang, Y. Ren, L. Liu, M. Li, Z. Ye, J. Liu, X. Yin, and Z. Zhao. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. In *International Conference on Machine Learning*, pages 13916–13932. PMLR, 2023.

[23] V. Iashin, W. Xie, E. Rahtu, and A. Zisserman. Synchformer: Efficient synchronization from sparse cues. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5325–5329. IEEE, 2024.

[24] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. Oord, S. Dieleman, and K. Kavukcuoglu. Efficient neural audio synthesis. In *International Conference on Machine Learning*, pages 2410–2419. PMLR, 2018.

[25] C. D. Kim, B. Kim, H. Lee, and G. Kim. Audiocaps: Generating captions for audios in the wild. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 119–132, 2019.

[26] J. Kong, J. Kim, and J. Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in neural information processing systems*, 33:17022–17033, 2020.

[27] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley. Panns: Large-scale pretrained audio neural networks for audio pattern recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2880–2894, 2020.

[28] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

[29] K. Koutini, J. Schlüter, H. Eghbal-Zadeh, and G. Widmer. Efficient training of audio transformers with patchout. *arXiv preprint arXiv:2110.05069*, 2021.

[30] F. Kreuk, G. Synnaeve, A. Polyak, U. Singer, A. Défossez, J. Copet, D. Parikh, Y. Taigman, and Y. Adi. Audiogen: Textually guided audio generation. *arXiv preprint arXiv:2209.15352*, 2022.

[31] R. Kumar, P. Seetharaman, A. Luebs, I. Kumar, and K. Kumar. High-fidelity audio compression with improved rvqgan. *Advances in Neural Information Processing Systems*, 36:27980–27993, 2023.

[32] S.-g. Lee, W. Ping, B. Ginsburg, B. Catanzaro, and S. Yoon. Bigvgan: A universal neural vocoder with large-scale training. *arXiv preprint arXiv:2206.04658*, 2022.

[33] T. Li and K. He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025.

[34] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[35] H. Liu, Z. Chen, Y. Yuan, X. Mei, X. Liu, D. Mandic, W. Wang, and M. D. Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. *arXiv preprint arXiv:2301.12503*, 2023.

[36] H. Liu, K. Luo, J. Wang, W. Wang, Q. Chen, Z. Zhao, and W. Xue. Thinksound: Chain-of-thought reasoning in multimodal large language models for audio generation and editing. *arXiv preprint arXiv:2506.21448*, 2025.

[37] H. Liu, K. Luo, W. Wang, Q. Chen, P. Sun, R. Huang, X. Li, J. Ye, and W. Xue. Prismaudio: Decomposed chain-of-thoughts and multi-dimensional rewards for video-to-audio generation. *arXiv preprint arXiv:2511.18833*, 2025.

[38] H. Liu, Y. Yuan, X. Liu, X. Mei, Q. Kong, Q. Tian, Y. Wang, W. Wang, Y. Wang, and M. D. Plumbley. Audioldm 2: Learning holistic audio generation with self-supervised pretraining. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:2871–2883, 2024.

[39] X. Liu, C. Gong, and Q. Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[40] S. Luo, C. Yan, C. Hu, and H. Zhao. Diff-foley: Synchronized video-to-audio synthesis with latent diffusion models. *Advances in Neural Information Processing Systems*, 36:48855–48876, 2023.

[41] N. Majumder, C.-Y. Hung, D. Ghosal, W.-N. Hsu, R. Mihalcea, and S. Poria. Tango 2: Aligning diffusion-based text-to-audio generations through direct preference optimization. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 564–572, 2024.

[42] X. Mei, C. Meng, H. Liu, Q. Kong, T. Ko, C. Zhao, M. D. Plumbley, Y. Zou, and W. Wang. Wavcaps: A chatgpt-assisted weakly-labelled audio captioning dataset for audio-language multimodal research. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:3339–3354, 2024.

[43] W. Peebles and S. Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[44] K. Petrini, S. Dahl, D. Rocchesso, C. H. Waadeland, F. Avanzini, A. Puce, and F. E. Pollick. Multisensory integration of drumming actions: musical expertise affects perceived audiovisual asynchrony. *Experimental brain research*, 198(2):339–352, 2009.

[45] A. Polyak, A. Zohar, A. Brown, A. Tjandra, A. Sinha, A. Lee, A. Vyas, B. Shi, C.-Y. Ma, C.-Y. Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

[46] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[47] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[48] T. Salimans and J. Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

[49] S. Shan, Q. Li, Y. Cui, M. Yang, Y. Wang, Q. Yang, J. Zhou, and Z. Zhong. Hunyuanvideo-foley: Multimodal diffusion with representation alignment for high-fidelity foley audio generation. *arXiv preprint arXiv:2508.16930*, 2025.

[50] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[51] Z. Tian, Y. Jin, Z. Liu, R. Yuan, X. Tan, Q. Chen, W. Xue, and Y. Guo. Audiox: Diffusion transformer for anything-to-audio generation. *arXiv preprint arXiv:2503.10522*, 2025.

[52] A. Tjandra, Y.-C. Wu, B. Guo, J. Hoffman, B. Ellis, A. Vyas, B. Shi, S. Chen, M. Le, N. Zacharov, et al. Meta audiobox aesthetics: Unified automatic quality assessment for speech, music, and sound. *arXiv preprint arXiv:2502.05139*, 2025.

[53] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, et al. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 12(1), 2016.

[54] I. Viertola, V. Iashin, and E. Rahtu. Temporally aligned audio for video with autoregression. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2025.

[55] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

[56] H. Wang, J. Ma, S. Pascual, R. Cartwright, and W. Cai. V2a-mapper: A lightweight solution for vision-to-audio generation by connecting foundation models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 15492–15501, 2024.

[57] J. Wang, X. Zeng, C. Qiang, R. Chen, S. Wang, L. Wang, W. Zhou, P. Cai, J. Zhao, N. Li, et al. Kling-foley: Multimodal diffusion transformer for high-quality video-to-audio generation. *arXiv preprint arXiv:2506.19774*, 2025.

[58] S. Wang, Z. Gao, C. Zhu, W. Huang, and L. Wang. Pixnerd: Pixel neural field diffusion. *arXiv preprint arXiv:2507.23268*, 2025.

[59] Y. Wang, W. Guo, R. Huang, J. Huang, Z. Wang, F. You, R. Li, and Z. Zhao. Frieren: Efficient video-to-audio generation network with rectified flow matching. *Advances in neural information processing systems*, 37:128118–128138, 2024.

[60] Y. Wu, K. Chen, T. Zhang, Y. Hui, T. Berg-Kirkpatrick, and S. Dubnov. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[61] J. Xu, Z. Guo, H. Hu, Y. Chu, X. Wang, J. He, Y. Wang, X. Shi, T. He, X. Zhu, et al. Qwen3-omni technical report. *arXiv preprint arXiv:2509.17765*, 2025.

[62] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

[63] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2021.

[64] Y. Zhang, Y. Gu, Y. Zeng, Z. Xing, Y. Wang, Z. Wu, B. Liu, and K. Chen. Foleycrafter: Bring silent videos to life with lifelike and synchronized sounds. *International Journal of Computer Vision*, 134(1):46, 2026.

A Training Details

Table 7 summarizes the full training configurations for all WavFlow variants. All models are trained on NVIDIA H100 GPUs and share the same optimizer (AdamW with β_1=0.9, β_2=0.95), EMA decay of 0.9999, gradient clipping at 1.0, and BF16 mixed precision. In our main experiments, 16 kHz VT2A models are trained from scratch with a learning rate of 1×10^{-4} and a global batch size of 10,752 on the mixture described in Sec. 4.1 (comprising 5 M media data and 200 K augmented VGGSound). Within each epoch, every dataset is traversed exactly once to ensure balanced supervision. The T2A model follows the same architecture but is trained separately on the 1 M T2A mixture with a batch size of 8,192. WavFlow-L-44.1kHz is obtained via supervised fine-tuning from the converged 16 kHz checkpoint on 44.1 kHz data, using a reduced learning rate of 1×10^{-5} and a batch size of 1,536.

Table 7: Training hyperparameters for all WavFlow variants. All models use 16 kHz sample rate except WavFlow-L-44k (44.1 kHz).

	VT2A			T2A
Hyperparameter	WavFlow-M-16k	WavFlow-L-16k	WavFlow-L-44k	WavFlow-M-16k
Backbone[†]	MMDiT-M	MMDiT-L	MMDiT-L	MMDiT-M
Joint layers L_{joint}	4	7	7	4
Single layers L_{fused}	8	14	14	8
Hidden dim d	896	896	896	896
Attention heads	14	14	14	14
Parameters	624M	1.03B	1.03B	624M
Training data	Media data 5M + VGG 200K	Media data 5M + VGG 200K	VGG 200K (44.1 kHz)	Media data 1M + T2A open source 300K
Optimizer	AdamW	AdamW	AdamW	AdamW
(β_1, β_2)	(0.9, 0.95)	(0.9, 0.95)	(0.9, 0.95)	(0.9, 0.95)
Learning rate	1e-4	1e-4	1e-5	1e-4
LR schedule	Constant	Constant	Constant	Constant
Warmup	20 epochs	20 epochs	20 epochs	20 epochs
Global batch size	10752	10752	1536	8192
Training epochs	400	400	650	400
Initialization	Scratch	Scratch	SFT from L-16k	Scratch
EMA decay	0.9999	0.9999	0.9999	0.9999
Gradient clipping	1.0	1.0	1.0	1.0
Precision	BF16	BF16	BF16	BF16
Patchify ($C \times D$)	640×200	640×200	1764×200	640×200
Audio scale	3.0	3.0	3.0	3.0
CFG drop rate (video / text)	10% / 10%	10% / 10%	10% / 10%	– / 10%
ODE steps (inference)	50	50	50	50
CFG strength (inference)	4.5	4.5	4.5	4.5

[†]Multimodal DiT backbone adopted from MMAudio [7]. "M" and "L" denote the medium and large configurations.

To determine the appropriate number of training epochs, we budget each convergence study to approximately 84 hours and scale the global batch size proportionally with dataset size so that all runs complete within this timeframe. We monitor validation metrics on the VGGSound validation set ($\sim$2 K samples) throughout training across four data scales (200 K, 1 M, 3 M, and 5 M). As shown in Figure 5, models trained with $\geq$ 1 M data and correspondingly larger batch sizes (5,632–10,752) converge by approximately 400 epochs, with FD_{PANNs}, KL_{PANNs}, and IB scores all plateauing beyond this point. We therefore adopt 400 epochs as the default for both main and ablation experiments at this scale. The 200 K VGGSound-only run requires approximately 650 epochs to stabilize due to its much smaller batch size of 1,536, and the 44.1 kHz fine-tuning setting similarly uses 650 epochs to reach convergence under its reduced learning rate. Additionally, the global batch sizes labeled in Figure 5 correspond to the fixed settings utilized in our ablation studies.

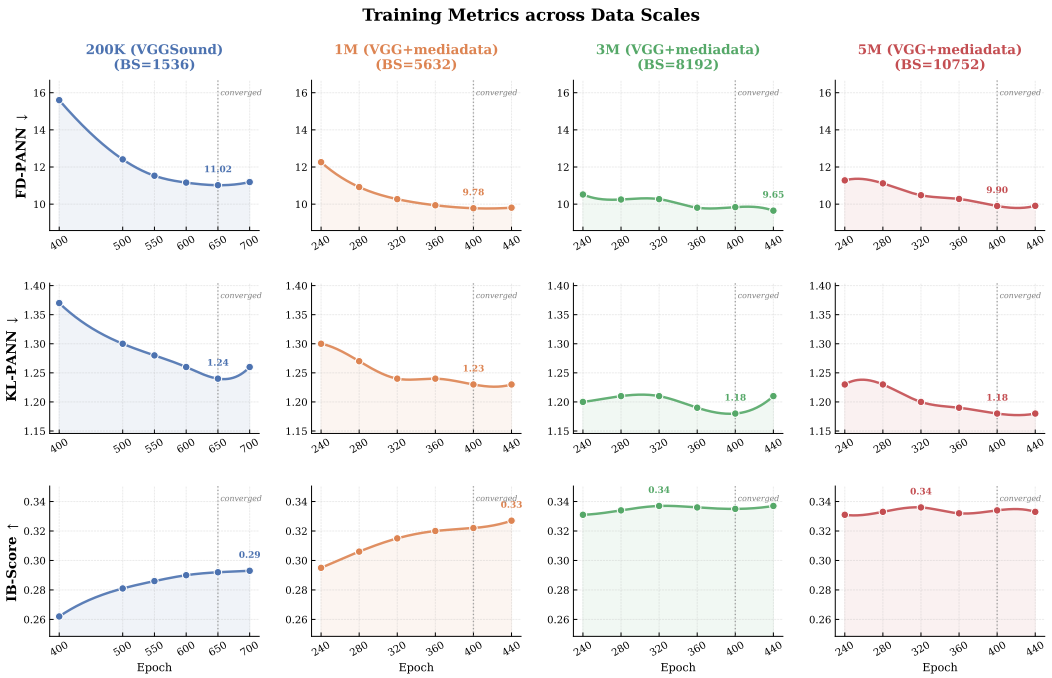


Figure 5: **Metrics on the VGGSound validation set as a function of training epochs across four data scales:** 200 K VGGSound (BS=1,536), 1 M (BS=5,632), 3 M (BS=8,192), and 5 M (BS=10,752). Top row: $\text{FD}_{\text{PANNs}}\downarrow$; middle row: $\text{KL}_{\text{PANNs}}\downarrow$; bottom row: IB $\uparrow$. Models trained with $\geq$ 1 M samples converge around 400 epochs, while the 200 K setting requires $\sim$650 epochs due to its smaller batch size. Dashed lines indicate the convergence points adopted for final training. Here, BS denotes the global batch size.

B Audio Amplitude Distribution

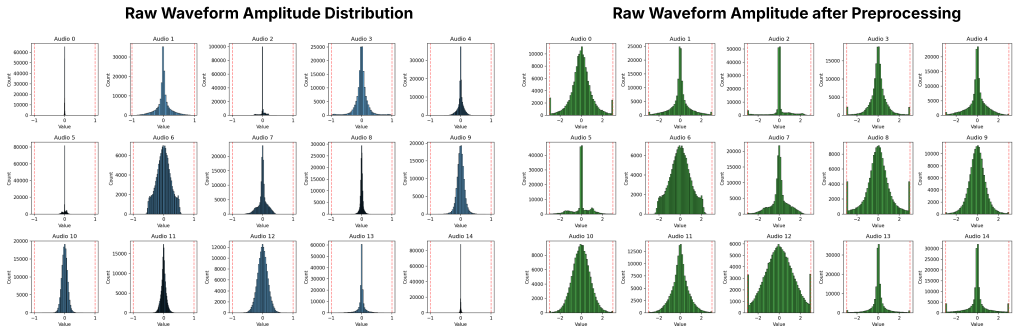


Figure 6: **Amplitude histograms of** 15 **randomly sampled audio clips before (left, blue) and after (right, green) preprocessing** (RMS normalization to 0.33 followed by $\times$3.0 scaling). Raw waveforms exhibit sharp zero-centered peaks with widely varying energy across clips. After preprocessing, the amplitude values are more consistent, lifting the signal to ensure it is not submerged by the Gaussian noise prior during flow matching.

Waveform-space amplitudes exhibit a wide dynamic range, yet values are heavily concentrated near zero, with empirical observations indicating that most waveform RMS levels typically remain below 0.2. The left panel of Figure 6 illustrates this by plotting the amplitude histograms of 15 randomly sampled clips. Quiet samples, such as Audio 0, 2, and 14, occupy only a minimal fraction of the available dynamic range, while others, such as Audio 4, 9 and 11, show broader yet still sharply

zero-centered distributions. These low-energy signals possess negligible magnitudes and are easily submerged by noise during training. This further results in vanishingly small loss values, providing insufficient gradient information and making the denoising objective difficult to optimize.

In contrast, after RMS normalization to a target level of 0.33 followed by $\times 3.0$ amplitude scaling (right panel), the signals are effectively lifted, with distributions spreading more broadly across the $[-3, 3]$ range. While the resulting distributions remain non-uniform, their amplitude statistics are much better aligned with the standard $N(0, 1)$ noise prior than the raw waveforms. This redistribution ensures that the signal remains discernible even at high noise levels, rendering the denoising objective significantly more tractable and training more stable. The quantitative impact of this preprocessing is validated through ablation experiments (Table 6).

C Data Mixture and Rationale for VT2A

This section describes our process for selecting the training data mixture for VT2A generation and the rationale behind our final configuration. We consider three primary data sources: (1) **VGGSound (200K)**, which consists of paired video-audio samples with **sparse captions** based on class labels (e.g., "dog barking" or "car passing by"); (2) **Open-source T2A data (300K)**, which includes audio-only samples from FreeSound and AudioCaps with **dense captions** (e.g., "a vehicle engine accelerating then running on idle"); and (3) **Media data**, a large-scale collection of 5M high-quality proprietary video-text-audio pairs, primarily sourced from MovieGen [45], with original dense captions similar in detail to the open-source T2A data. For the experiments in this section, we use a 1M subset of Media data to maintain experimental efficiency.

Our exploration began by directly mixing VGGSound (sparse captions) with open-source T2A data (dense captions). However, this configuration consistently led to training divergence, where the loss initially decreased but eventually spiked. We hypothesize that the mismatch in caption granularity, together with the absence of visual grounding in T2A samples, makes it difficult to learn a consistent conditioning relationship across the two datasets.

To address this, we use Qwen3-Omni [61] to generate dense audio-visual descriptions for VGGSound, aligning its caption style with that of the open-source T2A data. This dense-caption variant stabilizes training when mixed with T2A data. However, as shown in Table 8, its performance is inferior to the VGGSound-only baseline (DeSync $0.52 \rightarrow 0.57$, IB $0.29 \rightarrow 0.26$). This suggests that aligning caption styles alone is insufficient to match the performance of paired video-audio training data.

We then introduce the 1M Media data subset to provide additional paired video-audio supervision. Unlike the T2A mixture, combining VGGSound (sparse captions) with Media data (dense captions) converges reliably despite their different caption styles. This suggests that visual features may provide a common semantic anchor across datasets with different caption granularities, facilitating stable multimodal training.

Table 8: **Effect of data mixture and caption granularity on VT2A generation** (WavFlow-M-16k, using a 1M subset of Media data where applicable, evaluated on VGGSound-Val). "Open-source T2A" refers to a mixture of FreeSound and AudioCaps. **Note:** Dense captions are applied to the VGGSound training set only; the VGGSound validation set retains its original sparse labels.

Data Mixture	VGG Caption	FD_{PaSST} ↓	FD_{PANNs} ↓	KL_{PANNs} ↓	IS_{PANNs} ↑	IB↑	DeSync↓
VGGSound (200K)	Sparse	141.64	11.16	1.26	12.83	0.29	0.52
VGGSound + Open-source T2A	Sparse	*training diverged*					
VGGSound + Media data (1M)	Sparse	**121.09**	**9.58**	**1.20**	16.42	**0.33**	**0.47**
VGGSound + Open-source T2A	Dense	139.23	12.97	1.28	12.59	0.26	0.57
VGGSound + Media data (1M)	Dense	125.52	11.20	1.35	**17.05**	**0.33**	0.49

Finally, we compare caption granularity within the stabilized VGGSound + Media mixture. The dense-caption variant achieves higher IS_{PANNs} (17.05 vs. 16.42), suggesting that detailed descriptions may improve the audio diversity. However, the sparse-caption variant achieves better FD and DeSync scores (e.g., FD_{PaSST} of 121.09 vs. 125.52), potentially benefiting from greater consistency

between training and evaluation captions. Given its superior distributional fidelity and temporal synchronization, we select **VGGSound (sparse captions) + Media data** as our final training mixture.

Table 9: Examples of sparse and dense captions across different training datasets.Examples across datasets are not paired.

VGGSound (Sparse)	VGGSound (Dense)	T2A / Media VT2A
car passing by	A car approaches with growing engine noise, passes by with a loud rushing sound, and gradually fades into the distance.	A motor vehicle engine is running, a decelerating buzzing occurs, mechanical whines occur, and the motor vehicle engine accelerates.
goose honking	A goose makes a series of loud, nasal honking calls, with short pauses between the repeated honks.	A man speaking in a foreign language, can hear doves in the background.

D Waveform Patchify Configuration

As discussed in Section 3.2 and Section 4.5, we reshape an 8-second, 16 kHz waveform ($T = 128{,}000$ samples) into a $C \times D$ token grid. As illustrated in Figure 7, for audio samples where T is not exactly divisible by D, we apply zero-padding to the waveform prior to patching; this padding is subsequently truncated during the unpatchify process to restore the original waveform length T.

We sweep the patch dimension D from 512 down to 160 to determine the optimal granularity for waveform modeling (Table 10). This sweep includes specific configurations, such as $C = 576$ (192×3) and $C = 768$ (192×4), designed to align the audio token count with the Synchformer feature length (192 tokens) to test if such explicit choice benefits temporal alignment.

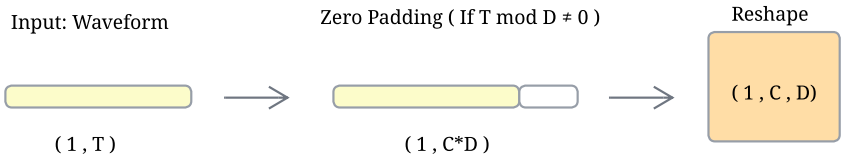


Figure 7: **Waveform patchify illustration.** A 1D waveform is reshaped into a 2D token grid of shape $C \times D$. Zero-padding is naturally applied to handle arbitrary waveform lengths, which is then removed during unpatchify to recover the original T samples.

Table 10: **Ablation on patchify granularity** (WavFlow-M-16k, 3M training data scale, evaluated on VGGSound-Val). All configurations target an 8 s waveform ($T = 128{,}000$).

Patchify ($C \times D$)	Token dur. (ms)	FD_{PaSST} ↓	FD_{PANNs} ↓	KL_{PANNs} ↓	IS_{PANNs} ↑	IB↑	DeSync↓
250×512	32.0	139.45	11.93	1.41	16.14	0.312	0.48
500×256	16.0	124.02	9.84	1.18	16.72	0.335	0.48
576×224	13.9	125.69	10.02	1.20	16.92	0.339	0.45
640×200	12.5	**120.88**	10.02	**1.16**	**17.20**	0.334	**0.45**
768×168	10.4	124.05	9.91	1.18	17.00	0.338	0.47
800×160	10.0	121.83	**9.90**	1.22	17.06	**0.340**	0.46

The results in Table 10 show that the coarsest configuration ($D = 512$, 32 ms per token) performs significantly worse, indicating that the model requires finer granularity to capture waveform details. Once the patch dimension D is reduced below 256 (16 ms), the generative performance and synchronization metrics stabilize at a high level.

Notably, we found that configurations specifically designed for sync-alignment (e.g., $C = 576$ or 768) did not yield substantial improvements over other fine-grained settings. This suggests that the

model is robust to various token counts as long as the temporal resolution is sufficient. Considering the balance between generative quality, computational efficiency, and synchronization, we select 640×200 as the default configuration for our 16 kHz experiments.

E Effect of Noise-Level Shift

We evaluate the impact of noise-level shifting on the VGGSound test set. Due to space constraints in the main text, these detailed results are provided here in the appendix. Prior work has shown that increasing the noise level during training can improve generation performance, particularly for high-resolution images [33, 19]. Motivated by this finding, we investigate whether noise shift similarly benefits waveform-space generation.

We parameterize noise shift as $t_s = t/(t + s \cdot (1-t))$ where s is the shift factor, which biases training toward higher noise levels and reduces the effective signal-to-noise ratio by a factor of s^2.

Table 11: **Effect of noise shift on VT2A generation** (WavFlow-M-16k, 1M training data scale, evaluated on VGGSound-Test). All configurations target an 8 s waveform ($T = 128{,}000$).

Noise shift s	$\text{FD}_{\text{PaSST}}\downarrow$	$\text{FD}_{\text{PANNs}}\downarrow$	$\text{KL}_{\text{PANNs}}\downarrow$	$\text{IS}_{\text{PANNs}}\uparrow$	IB↑	DeSync↓
1.0	**63.05**	**6.21**	1.76	**15.58**	**0.28**	**0.50**
3.0	73.17	7.52	**1.72**	13.64	0.26	0.50
5.0	92.21	10.25	1.80	11.81	0.23	0.55

As shown in Table 11, unlike in image generation, noise shift provides little benefit for waveform-space modeling and progressively degrades all metrics as s increases, with FD_{PaSST} rising from 63.05 ($s = 1.0$) to 73.17 ($s = 3.0$) and then to 92.21 ($s = 5.0$). We attribute this to the fundamental difference in signal characteristics: image pixels span a broad value range where high-noise training helps the model capture low-frequency global structure, whereas audio waveforms have inherently low information density even after our preprocessing. In this regime, increasing the noise level makes the already weak audio signal even harder to recover, degrading rather than improving learning. We therefore adopt $s = 1.0$ (no shift) as our default.

F Inference Hyperparameters

We ablate two key inference-time hyperparameters on VT2A (VGGSound validation set): the classifier-free guidance (CFG) strength and the number of ODE integration steps.

Table 12: **Effect of inference hyperparameters on VT2A** (WavFlow-M-16k, 1M training data scale, evaluated on VGGSound-Val). All configurations target an 8 s waveform ($T = 128{,}000$).

Setting	$\text{FD}_{\text{PaSST}}\downarrow$	$\text{FD}_{\text{PANNs}}\downarrow$	$\text{KL}_{\text{PANNs}}\downarrow$	$\text{IS}_{\text{PANNs}}\uparrow$	IB↑	DeSync↓
CFG strength (50 steps)						
1.0	145.53	16.34	1.54	9.86	0.26	0.71
2.5	**108.19**	**9.40**	1.26	15.44	0.32	0.51
4.5	121.09	9.58	**1.20**	**16.42**	**0.33**	0.47
7.0	142.00	9.92	1.22	16.19	0.32	**0.46**
ODE steps (CFG=4.5)						
10	118.90	10.73	1.30	14.93	0.31	0.47
25	**117.63**	9.78	1.23	16.19	0.32	0.47
50	121.09	9.58	**1.20**	**16.42**	**0.33**	0.47
100	124.53	**9.41**	**1.20**	16.39	**0.33**	0.47

As shown in Table 12, CFG strength has a pronounced effect on generation quality. Without sufficient guidance (CFG = 1.0), the model produces low-fidelity outputs with poor semantic alignment (IB:

0.26) and severe temporal artifacts (DeSync: 0.71). Increasing CFG to 2.5 yields the best distributional fidelity (FD_{PaSST}: 108.19), while CFG = 4.5 achieves the highest per-sample quality (IS: 16.42, IB: 0.33). Beyond 4.5, further increasing guidance to 7.0 causes FD_{PaSST} to degrade sharply, indicating reduced output diversity without meaningful quality gains.

For ODE steps, the transition from 10 to 25 steps brings a significant quality improvement, and going from 25 to 50 steps yields further gains in IS ($16.19 \rightarrow 16.42$) and IB ($0.32 \rightarrow 0.33$). Beyond 50 steps, metrics plateau entirely, with 100 steps providing no additional benefit. We therefore adopt CFG = 4.5 with 50 ODE steps as our default inference configuration, which offers the best generation quality before diminishing returns set in.

G Human Evaluation

We conduct a human evaluation with five non-author raters on 200 randomly sampled videos from VGGSound-Test. Each rater evaluates 40 distinct videos, rating the generated audio from three models: WavFlow, MMAudio, and HunyuanVideo-Foley. For each video, all three models are evaluated by the same rater, resulting in 200 ratings per model. We assess three dimensions on a 1–5 scale (higher is better): audio quality (MOS-Q), temporal synchronization (MOS-T), and audio-visual alignment (MOS-S). Model identities are anonymized, and the presentation order is randomized to reduce evaluation bias. We report the mean opinion score (MOS) and 95% confidence interval for each model and dimension.

Table 13: Human evaluation on VGGSound-Test with five non-author raters and 200 videos. Results are reported as MOS with 95% confidence intervals (higher is better).

Model	MOS-Q↑	MOS-T↑	MOS-S↑
HunyuanVideo-Foley	4.06 ± 0.12	3.81 ± 0.15	3.88 ± 0.15
MMAudio	3.98 ± 0.12	4.05 ± 0.13	4.07 ± 0.13
WavFlow	4.02 ± 0.12	4.04 ± 0.14	4.03 ± 0.13

As shown in Table 13, WavFlow achieves human evaluation scores comparable to MMAudio across all three dimensions. HunyuanVideo-Foley obtains a slightly higher audio quality score, while WavFlow and MMAudio achieve higher scores in temporal synchronization and audio-visual alignment. Overall, the differences are small relative to the reported confidence intervals, suggesting comparable perceptual performance among the evaluated models.

H Evaluation on MovieGen-Audio-Bench

To further evaluate the generalization of WavFlow, we conduct additional experiments on the MovieGen-Audio-Bench [45]. This benchmark is particularly challenging as it consists of AI-generated videos rather than real-world recordings. Performing audio synthesis for such synthetic content requires the model to establish a fundamental understanding of audio-visual correlation.

Since this benchmark does not provide ground-truth audio, we follow the protocol in [45] and report reference-free metrics: Inception Score (IS) for audio quality, CLAP and IB-score for semantic alignment, and DeSync for temporal synchronization. Table 14 presents the comparison between WavFlow, MMAudio, and MovieGen.

Table 14: **Evaluation on MovieGen-Audio-Bench.** Best results are in **bold**, and second-best are underlined. Reference-free metrics are used as no ground-truth audio is available for this benchmark.

Method	Params	Training Data	IS↑	CLAP↑	IB-score↑	DeSync↓
WavFlow (ours)	1.03B	$\sim$ 11.1K h	**8.95**	0.28	0.24	**0.77**
MMAudio	1.03B	$\sim$ 8.2K h	8.40	0.28	0.27	**0.77**
MovieGen	13B	$\sim$ 1,000K h	8.89	**0.29**	**0.36**	1.00

Experimental results show that WavFlow generalizes effectively to synthetic visual content. Notably, WavFlow achieves an Inception Score (IS) of 8.95 and a DeSync score of 0.77, outperforming or matching existing latent-based models in audio quality and temporal synchronization. While MovieGen maintains an advantage in semantic richness (IB-score), likely due to its significantly larger model capacity and training scale, WavFlow remains highly competitive. These results suggest that direct raw-waveform synthesis, without the aid of a pretrained VAE, achieves performance comparable to state-of-the-art latent-based paradigms across both real and synthetic benchmarks.

To qualitatively evaluate temporal synchronization, we provide spectrogram visualizations of samples from the MovieGen-Audio-Bench in Figures 8 to 10.

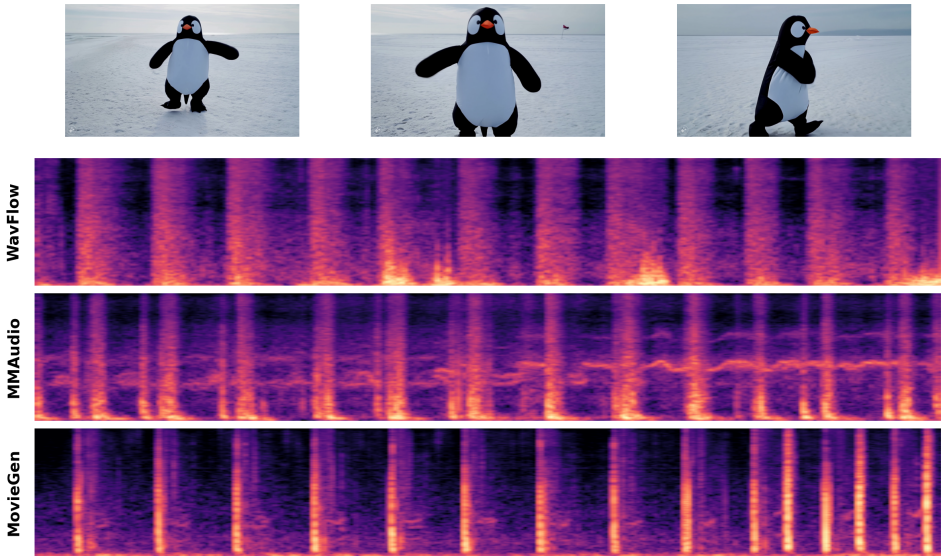


Figure 8: **Spectrogram comparison on the "Penguin Walking" scenario.** All evaluated models demonstrate precise temporal synchronization. WavFlow exhibits sharper and more vertical energy pulses, reflecting its ability to preserve acoustic transients through direct raw-waveform modeling.

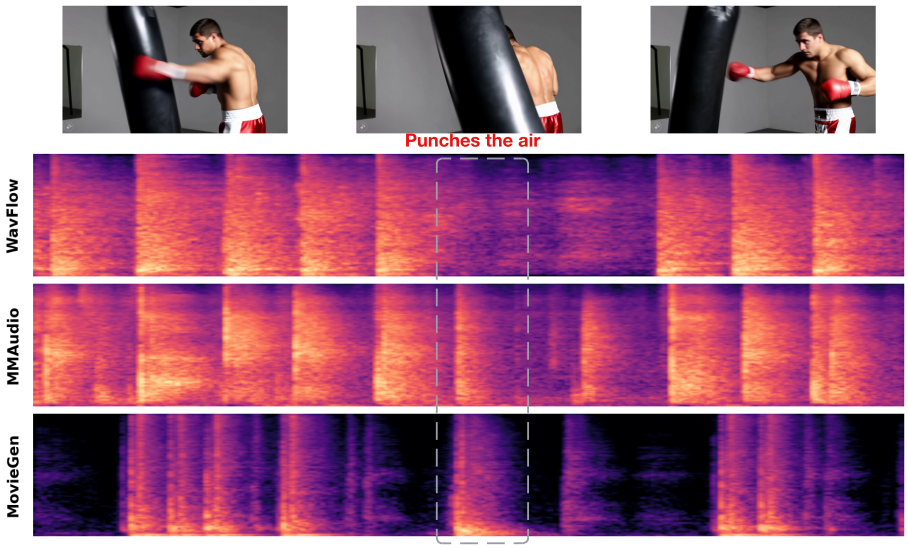


Figure 9: **Spectrogram comparison on the "Boxing" scenario.** Acoustically, both WavFlow and MMAudio achieve precise synchronization, while MovieGen exhibits desynchronization. Crucially, during the air-punching segment (no bag contact), WavFlow correctly omits the impact sound, whereas MMAudio and MovieGen erroneously synthesize strike energy. This highlights WavFlow's precision in discerning subtle visual nuances for accurate raw-waveform synthesis.

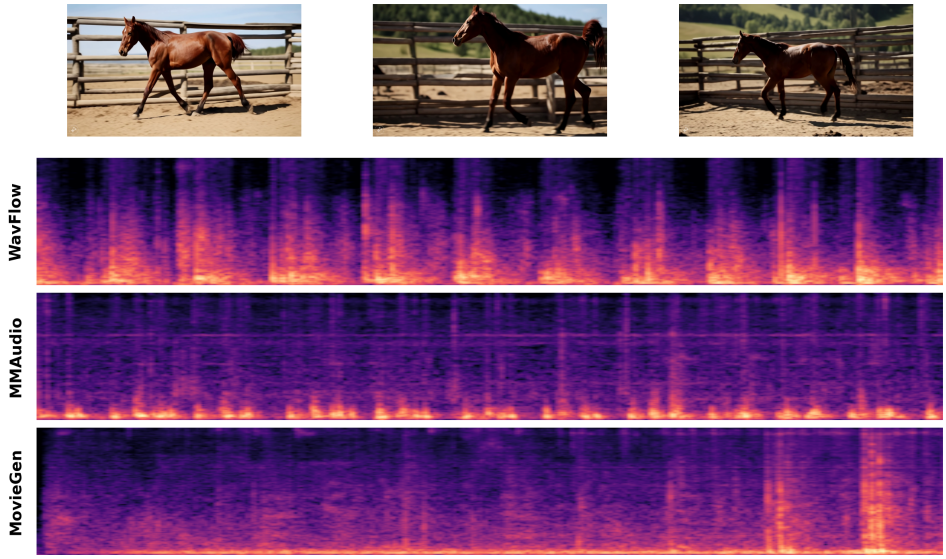


Figure 10: **Spectrogram comparison on the "Horse Trotting" scenario.** In this scenario featuring consistent, rhythmic hoofbeats, WavFlow achieves precise temporal synchronization, characterized by distinct and sharp vertical energy pulses that correspond to each footfall. While MMAudio maintains some rhythmic alignment, its spectral pulses are noticeably less defined and lack the transient sharpness seen in our model. In contrast, MovieGen fails to establish a clear temporal relationship with the visual motion.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction explicitly state the paper's scope and contributions, and these statements are aligned with the reported experiments and conclusions (Sec. 1, Sec. 4).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We explicitly discuss key limitations, including data dependence and scope of generalization, in the conclusion and related discussion (Sec. 5).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: This work is empirical and does not present new theorems, formal proofs, or theory-driven claims that require proof appendices.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide reproducibility-critical details, including data construction, model variants, training hyperparameters, and evaluation settings, in Sec. 4, Sec. A, and Sec. C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We plan to release code at publication. Our training set combines public data with proprietary data that cannot be redistributed due to licensing constraints. To support reproducibility, we describe the full pipeline and provide sufficient setup details so results can be approximated with public replacements.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We provide detailed training and evaluation settings, including data mixtures, model configurations, optimizer settings, and inference hyperparameters (Sec. 4, Sec. A, Sec. C, Sec. F).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: We report deterministic benchmark metrics for all methods and ablations, but we do not include repeated-run statistics or confidence intervals in this submission due to computational cost.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide compute resource details, including hardware platform, batch sizes, epochs, and training budget/convergence settings in Appendix A (Sec. A).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and conducted this work under those guidelines, including responsible reporting of data usage and release constraints.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss both beneficial use cases and potential misuse/limitations in the conclusion section (Sec. 5).

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This submission does not release high-risk generative assets or unrestricted proprietary data; therefore, release-time safeguard mechanisms are not applicable here.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We appropriately cite and credit all external datasets, models, and methods used in this work and comply with their applicable licenses and usage terms. Our proprietary media data is primarily sourced from the internal MovieGen data corpus [45], used under applicable internal data access and usage restrictions.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We plan to release the codebase upon publication.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: We conducted a small-scale, voluntary human evaluation with five non-author raters, as described in Appendix G. The study served only as supplementary perceptual validation of the generated audio.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [Yes]

Justification: In the supplementary human evaluation, no sensitive personal information was collected, and no foreseeable risks to participants were identified.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: We use Qwen3-Omni [61] only to generate dense captions for the ablation study in Appendix C. All main experiments use original dataset labels or captions, including the original dense captions of our proprietary Media data, without LLM-based re-captioning. The core model architecture and training objective do not rely on LLMs.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.