Towards Multi-Modal Interactive Systems that Connect Audio, Vision and Beyond

Xiulong Liu

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2025

Reading Committee:
Eli Shlizerman, Chair
Payman Arabshahi
Radha Poovendran

Program Authorized to Offer Degree:
Electrical and Computer Engineering

University of Washington

Abstract

Towards Multi-Modal Interactive Systems that Connect Audio, Vision and Beyond

Xiulong Liu

Chair of the Supervisory Committee:
Eli Shlizerman
Electrical and Computer Engineering

Human perception relies on the integration of multi-sensory signals such as vision, audio and language, enabling complex tasks such as interpreting environments and making decisions. In Artificial Intelligence (AI), multi-modal learning seeks to enable such abilities. Significant progress has been made in text-centered multi-modal learning, e.g., vision-language learning, while understanding the connections between vision and audio is still less characterized. Unlocking these relationships can enable AI systems to generate realistic audio, enhance contextual understanding for execution of complex tasks, and create interactive interfaces. These can support applications in augmented reality, robotics, and virtual experiences.

Audio-visual learning is challenging due to the high-dimensional and redundant nature of both vision and audio signals. The signals are continuous and contain various information that are not necessarily organized. Existing text-centered multi-modal learning methods do not generalize to audio-visual scenarios since they rely on strong semantic properties of text modality. In particular, when these methods are directly adapted to audio-visual tasks, they often fail to capture meaningful cross-modal interactions that are important for downstream tasks. It is therefore important to develop specific multi-modal systems for audio-visual scenarios to effectively handle redundancy. These systems should be capable of solving a range of tasks, from audio generation to complex multi-modal understanding problems such as question answering and navigation.

To address these challenges, in my PhD research I developed multi-modal interactive systems. These systems are of two types: **Multi-modal audio generation systems** and **Multi-modal task solving systems**. Audio generation systems aim to create as close as possible realistic audio from visual and other modalities inputs by modeling temporal and semantic correspondences. Task solving systems process audio-visual and other multi-modal inputs to support tasks such as retrieval, question answering, and navigation. They do so through multi-modal interactions to extract task-relevant cues.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I wish to express my deepest gratitude to my advisor, Prof. Eli Shlizerman, for his unwavering guidance throughout my research journey. His insights have been instrumental in shaping both my academic work and industry contributions, consistently helping to sharpen my research focus and deepen my understanding. Beyond academia, his thoughtful advice on life and career has had a lasting impact, for which I am truly grateful.

I would like to express my sincere appreciation to my supervisory committee members - Prof. Payman Arabshahi, Prof. Radha Poovendran, and Prof. Vikram Iyer - for their thoughtful and continued support throughout my Ph.D. I am am especially grateful to the A3D3 Institute, under the leadership of Prof. Shih-Chieh Hsu, for providing funding and essential computational resources that enabled my research.

I am grateful for the camaraderie and support of my colleagues and friends—Kun Su, Mingfei Chen, Zhikang Dong, Sudipta Paul, among others—whose presence has made my Ph.D. journey more meaningful and enjoyable.

My industry internships played a key role in shaping how I approach research. At Mitsubishi Electric Research Laboratories, I had the chance to work closely with Dr. Anoop Cherian and Dr. Moitreya Chatterjee, whose guidance helped me grow both technically and professionally. During my time at TikTok, Dr. Peng Zhang offered thoughtful feedback that sharpened my research thinking. At Meta Reality Lab, I worked in a supportive and collaborative environment with Prof. Ruohan Gao (now at the University of Maryland, College Park) and Dr. Vamsi Krishna Ithapu. I am also grateful to my peer groups at Meta - Anurag Kumar (now at Google DeepMind),

Paul Calamia, Sebastià V. Amengual, Ishwarya Ananthabhotla, Calvin Murdock, and Philip Robinson - for their steady support and guidance.

Looking back on the past five years, the challenges and setbacks I encountered during my Ph.D. at the University of Washington have shaped me into a more resilient and capable person. These experiences have strengthened my resolve, and I now face future challenges with renewed confidence.

DEDICATION

With heartfelt love, I dedicate this to my parents, Hong Yang and Wenping Liu, and to my entire family—including my two cherished pets, Toothless and Aurora—whose unwavering support, encouragement, and companionship have been a constant source of strength throughout this journey.

Chapter 1

INTRODUCTION

1.1 Background

Our perception relies on seamless integration of multi-sensory signals such as vision, audio and language. These modalities complement each other to enable us to perform complex tasks based on context, from interpreting environments to making decisions. In Artificial Intelligence (AI), the study of such interactions, known as multi-modal learning, aims to enable this integration by machine learning models to process and learn from diverse modalities. While significant advancements have been made in vision-language and audio-language learning, understanding and leveraging the intricate relationships between multiple modalities, especially vision-audio, remains a largely yet to be well explored domain. Unlocking such relationships between visual and audio can enable AI systems to perform tasks such as generating realistic sounds for visual scenes, understanding environmental contexts for navigation, or building more interactive and accessible interfaces. These capabilities are foundational for applications in augmented reality, robotics, and immersive virtual experiences.

Audio-visual learning is challenging due to the complex and redundant nature of vision and audio signals. Unlike text, which consists of discrete tokens with strong semantic meaning and minimal redundancy, vision and audio are continuous and high-dimensional. The information within is not semantically organized, such that its relevance to the specific tasks is not directly interpretable. This imposes challenges on focusing on the parts relevant to specific tasks. The relationships of the modalities are highly diverse, ranging from clear temporal alignments, e.g., finger movements corresponding to piano notes, to abstract semantic associations, e.g., environmental sounds paired with visual scenes. While text-based multi-modal learning, e.g., vision-language learning, has developed unified methods that can

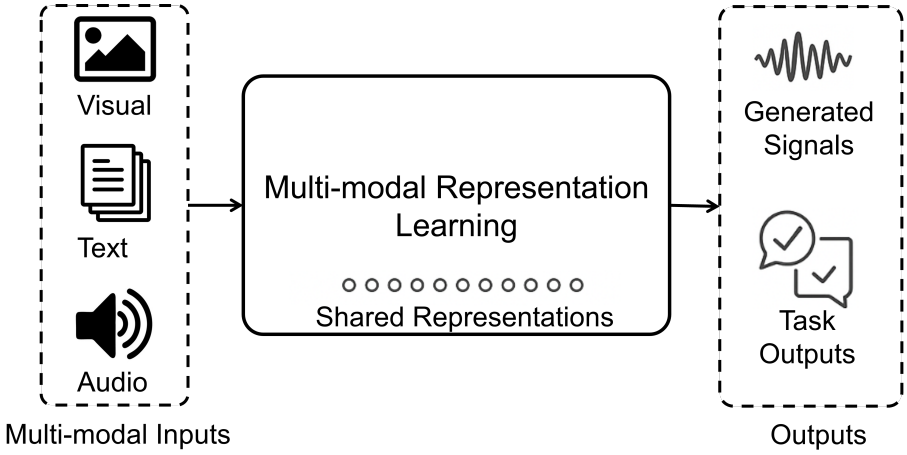

Figure 1.1: A high level conceptual overview of a design of multi-modal interactive system.

generalize across tasks, audio-visual learning remains focused on domain-specific problems and does not immediately generalize across scenarios. These challenges, such as managing redundancy, modeling cross-modal interactions, including temporal and semantic correspondence and ensuring generalization across tasks, require novel systems design tailored to the unique nature of audio-visual learning. To address the aforementioned challenges, my research focuses on developing systems that model the connections between vision and audio with possible inclusion of other modalities such as text, for a variety of tasks. From a high-level viewpoint, my works can be organized into proposing systems which receive multi-modal inputs and then transform them into different modalities outputs through multi-modal representation learning, as shown in Figure 1.1. From this core and abstract representation, my work can be organized into two concrete branches (as shown in Figure 1.2). These correspond to: 1. **Multi-modal audio generation systems**, designed for generating realistic audio from visual and other modality inputs by addressing both temporal and semantic correspondence. 2. **Multi-modal task solving systems**, designed to process audio-visual and other modality inputs. These systems support tasks such as retrieval, question answering, and navigation by leveraging interactions across multiple modalities. By developing efficient system pipelines and fusion methods for integrating multi-modal data, my work reduces re-

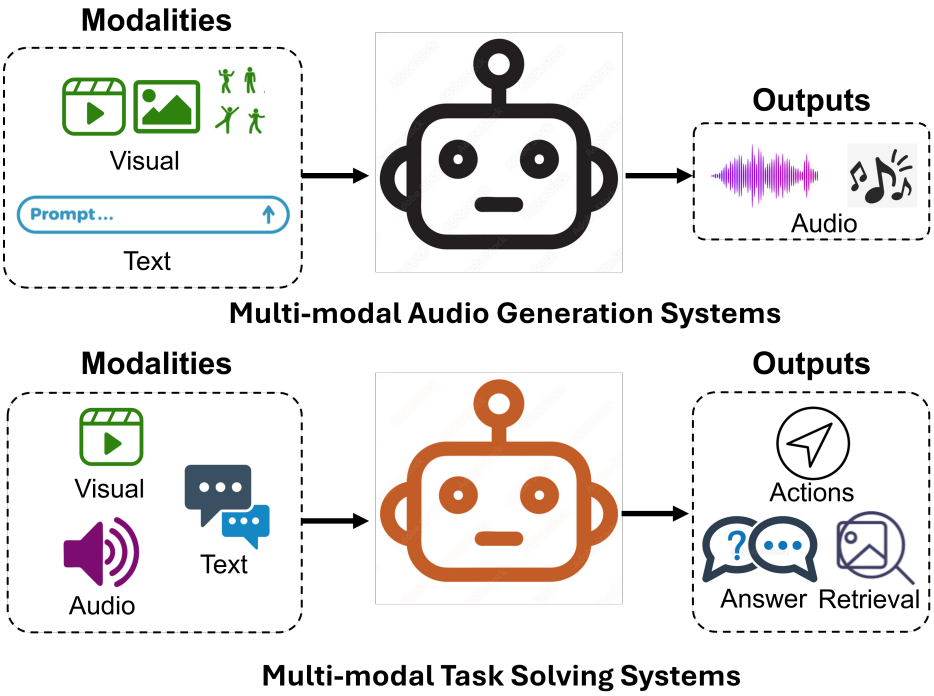

Figure 1.2: An overview of two types multi-modal interactive systems proposed. Top: Multi-modal Audio Generation Systems. Bottom: Multi-modal Task Solving systems.

dundancy in vision and audio, models their diverse interactions, and improves the ability to generalize across different scenarios.

However, modeling these interactions is non-trivial, especially due to the temporal nature of both video and audio. Unlike static images, these modalities unfold over time, introducing a dynamic dimension that complicates their alignment. For instance, the occurrence of a dog barking is closely tied to the moment it opens its mouth, rather than the mere presence of the dog in the scene. Instead, this example highlights the critical role of timing in the audio-visual correlation. Furthermore, while certain scenarios might allow for a direct mapping from visual cues to sound - such as identifying specific piano keys being pressed in a top-down piano performance video - this level of precision is not universally applicable. In other contexts, such as music dance videos, the audio-visual correspondence becomes more subjective, emphasizing dance beats and styles rather than explicit visual cues which

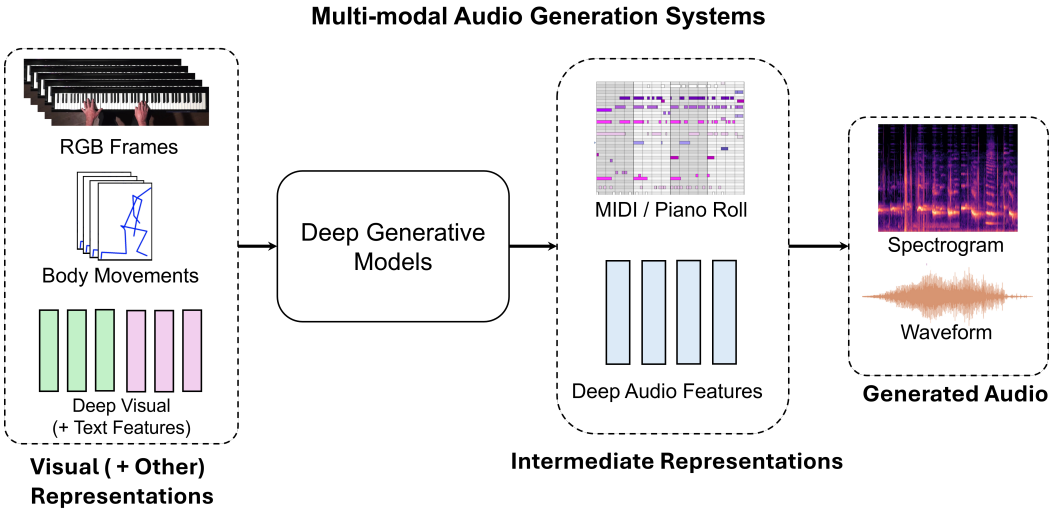

Figure 1.3: A high-level overview of Multi-modal Audio Generation Systems: Various visual (+ other modality) representations as well as the intermediate audio representations are selected for different audio generation tasks. Existing deep generative models such as GAN, autoregressive models are applied to transform the input reprentations into audio representations.

introduce ambiguity.

In terms of Multi-modal audio generation systems that generate audio from visual inputs (with optional extra modalities), these require addressing two key types of audio-visual correspondence: *temporal correspondence*, where visual and audio signals are closely synchronized (e.g., finger movements on a piano producing notes), and *semantic correspondence*, where the connection between visual and audio is based on context (e.g., sounds of fountains and chatter in a tourist scene). For temporal correspondence, capturing fine-grained spatial and temporal dynamics is essential, while for semantic correspondence, extracting meaningful, high-level visual cues for generating audio is key.

The success of audio generation systems depends on selecting suitable representations for both audio and visual data aligned with the specific requirements of correspondence types. These representations are then integrated using advanced deep generative models, enabling

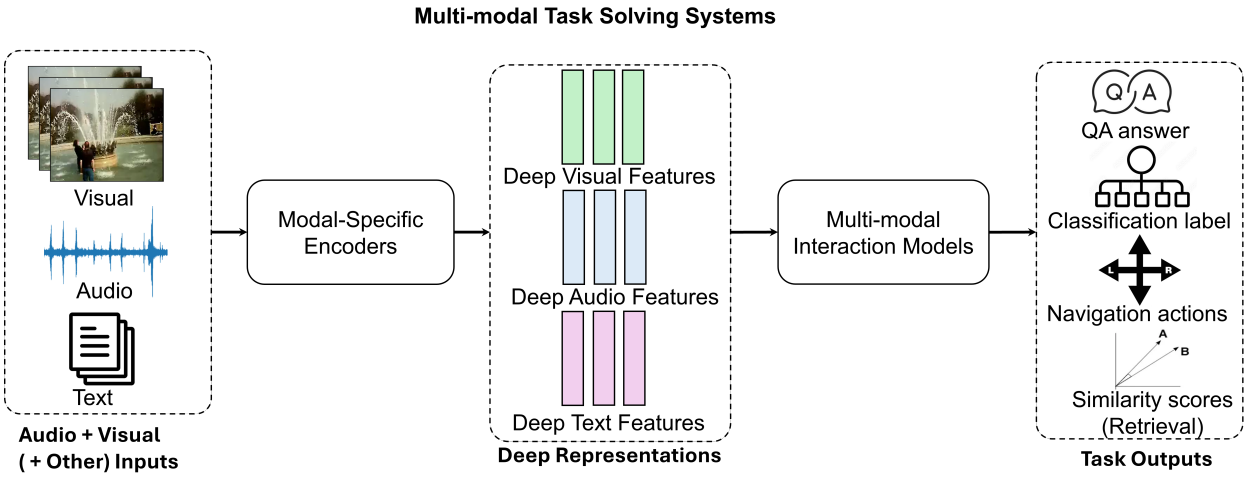

Figure 1.4: A high-level overview of Multi-modal task solving systems: The systems process inputs from visual, audio, and other modalities, e.g., text, through modality-specific encoders, which extract deep features that capture the essential characteristics of each modality. These features are then integrated into the multi-modal interaction models, which fuse information across modalities and focus on extracting task-relevant cues to solve downstream tasks.

the transformation from multi-modal inputs to realistic and contextually accurate audio outputs. The design choices for audio and visual representations, along with the deep generative approaches applied to generate the audio through various intermediate representations, are summarized in Figure 1.3.

In terms of systems for multi-modal inputs including audio, visual and other modalities to solve tasks, these require effective interactions among the modalities to extract task relevant information. At a high level, multi-modal interactions involve leveraging task-specific cues, such as a question or query in the context of question answering (QA), to guide the fusion and alignment of features across other modalities. Specifically, each modality is first processed by a modal-specific encoder to extract deep features that are then integrated into the multi-modal interaction models. These models, often employ attention mechanisms, which focus on the most relevant components of each modality, ensuring that the fused information directly

supports the target task.

The task solving pipeline, as shown in Figure 1.4, is designed to solve a variety of tasks such as classification, retrieval, QA and navigation effectively.

1.2 Summary of Lessons Learned and My Research Contributions

In **Multi-modal audio generation systems**, I developed multiple methodologies tailored to different types of audio-visual correspondences. These include systems that focus on temporal correspondence, semantic correspondence and their combinations. *These works have been among first explorations of generative models in the multi-modal audio generation domain. They uniquely address various correspondence types by designing novel audio-visual representations and set the foundation for multi-modal audio generation.*

- **Audeo** [XL1]: A two-stage generative model that uses Generative Adversarial Networks (GANs) to translate piano performance videos into corresponding music, leveraging fine-grained temporal visual-audio alignment.

- **RhythmicNet** [XL2]: A multi-stage system to generate rhythmic soundtracks for various human body movements (e.g., dancing, workouts) by capturing motion-based visual rhythms, generalizing vision-to-music applications to different kinds of human activities.

- **InteractiveBeat** [XL3]: This approach adapts **RhythmicNet** into a real-time, interactive soundtrack generation system for dynamic applications that follow causality.

- **MI-Net** [XL4]: A vision-to-music system for multi-instrument performance, which exploits the semantic correspondence between visual inputs and timbre in latent space.

- **VATT** [XL5]: A general video-to-audio generation system with optional text prompt, enabling sound generation across diverse categories from in-the-wild videos.

- **xRIR** [XL6]: A generalizable framework for room impulse response prediction, which captures the acoustic effects of different room environments with high resolution in order to render realistic spatial audio.

In **Multi-modal task solving systems**, I developed several multi-modal interaction models tailored to specific requirements that address different tasks. These systems can be categorized into basic perception tasks, such as audio-visual classification and retrieval, and more fine-grained tasks that require complex understanding. Such problems often rely on text inputs for better performance.

- **VAB** [XL7]: This model was adapted from a generative architecture to an understanding-based model, enabling tasks such as audio-visual retrieval and classification.

- **MuseChat** [XL8]: A conversational music recommendation system for videos that incorporates user preferences through text inputs, capturing dynamic user interactions.

- **MUSIC-AVQA-v2.0** [XL9]: An audio-visual QA system that handles complex spatial and temporal reasoning. It models the interplay among questions, audio and visual cues to provide accurate answers.

- **CAVEN** [XL10]: A dialogue-based audio-visual navigation system, which uses sparse text feedback and reinforcement learning to guide agents in challenging environments with intermittent or distracting audio cues.

1.3 Thesis Outline

This thesis is organized as follows: Chapter 2 provides an overview of fundamental concepts and related works. From Chapter 3 to 8, the thesis includes published works that are built for multi-modal generation audio generation systems. Chapter 3 describes the model for translating piano performance videos into corresponding music. Chapter 4 focuses on introducing the work in generation of rhythmic soundtracks for human activity videos with

various human body movements. Chapter 5 details how the rhythmic soundtracks generation can be adapted into a real-time soundtrack generation system to support interactive applications. Chapter 6 explores the generation of instrument music from musicians' body movements. Chapter 7 generalizes such generation from music domain into generic audio by combining audio-visual features with large language model. Chapter 8 further discusses leveraging audio-visual cues in room environments to render immersive and realistic spatial audio. From Chapter 9 to 12, published works of Multi-modal task solving systems are presented. Chapter 9 introduces equipment of an audio-visual generative model with audio-visual understanding capabilities using a unified framework to solve downstream tasks such as audio-visual retrieval and classification. Chapter 10 describes a design of a first-of-its-kind video to music recommendation system that allows interactions between user and system through conversation, refining the recommendation results in an iterative way while enhancing interpretability. Chapter 11 introduces a bias-free audio-visual QA benchmark and a system that can achieve the state-of-the-art performance. Chapter 12 extends the Multi-modal task solving system to embodied AI framework, introducing an audio-visual navigation system equipped with conversation capability with oracle for better performance in navigation. Chapter 13 concludes the thesis by summarizing key contributions for the two type of systems I have developed. The thesis incorporates works which have been published in the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) [XL6, XL8, XL11], Neural Information Processing Systems (NeurIPS) [XL5, XL2, XL1], International Conference on Machine Learning (ICML) [XL7], AAAI Conference on Artificial Intelligence (AAAI) [XL10], the IEEE Winter Conference on Applications of Computer Vision (WACV) [XL9, XL3]. [1]

[1]In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of University of Washington's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink. If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Chapter 2

FUNDAMENTAL CONCEPTS AND RELATED WORKS

This section provides an overview of key concepts and methods in the multi-modal learning literature. These are mostly related to audio and computer vision, known as audio-visual learning. Recent audio-visual learning includes the possible consideration of text as well. The discussion focuses on how each modality is represented and how they interact with each other given that deep learning tools are utilized for the problems within audio-visual learning. As mentioned in the previous section, this overview will be provided from two different perspectives: the generation perspective and the task solving perspective.

2.1 Audio Representations

Raw Waveform. An audio waveform is typically represented as one-dimensional time series, $\mathbf{x} = \{x_1, ..., x_t, ..., x_T\}$, where $x_t \in \mathbb{R}$ and T is the duration of the audio. While the waveform is a direct way for representation of audio, its high sampling rate poses significant modeling challenges. This is since a very short waveform that lasts one second would contain 16k to 44.1k samples, depending on the sampling rate. Furthermore, suitable approach to analyze such sequences would be sequence modeling. In the realm of deep learning, such models are known to be difficult to utilize since they often suffer from exploding or vanishing gradients [Re8]. To address these issues, specialized architectures have been proposed based on the design of effective techniques such as multi-scale dilated convolutional blocks. However, a more natural representation that fits into the deep learning framework would be the adoption of the time-frequency representations to substitute waveform representation.

Time-frequency Representations. The most common Time-frequency representation of audio is probably the spectrogram. The spectrogram is a 2D representation with two

axes, the temporal axis and the frequency axis. It is computed through Short-Time Fourier Transform (STFT). The algorithm starts with dividing the audio waveform into overlapping windows, with each window consisting of a certain number of samples. For each window, a function $w(t)$ such as Hann, is applied before calculating the Discrete Fourier Transform (DFT) on the window, $X(m,\omega) = \sum_t x(t)w(t-m)e^{-j\omega t}$. The index m denotes the temporal location, while ω represents the frequency location. The window size controls the number of terms in the summation while the FFT size determines the number of frequency bins that ω can take. These two parameters are critical when designing a spectrogram, as they dictate the trade-off between time and frequency resolution. Once the STFT $X(m,\omega)$ is computed, the spectrogram is obtained by taking the squared magnitude: $S(m,\omega) = |X(m,\omega)|^2$. The spectrogram captures the distribution of power across frequencies over time. Since the phase information is discarded in this representation, this results in a simpler representation. Additionally, since each time step in the spectrogram aggregates a wider temporal range of samples, the long-sequence modeling problem inherent to waveform representations is alleviated. A logarithmic transformation is often applied to the spectrogram to address dynamic range scaling and to better match human auditory perception.

Another popular Time-frequency representation is the Mel-scale spectrogram. The Mel scale is a perceptual scale in which equal distances correspond to equal perceived pitch differences. The conversion from frequency f in Hertz to Mel scale m is given by: $m = 2595\log_{10}(1+\frac{f}{700})$. A Mel spectrogram is obtained by applying this non-linear transformation to the frequency axis of a standard spectrogram. Since the number of Mel bins is typically smaller than the number of frequency bins in the linear scale, this representation can also be considered as a form of dimensionality reduction. Since the (log or Mel) spectrogram can be viewed as a 2D matrix similar to a grayscale image, many deep learning techniques from the image domain can be directly applied to audio analysis tasks.

Latent Representations While Time-frequency representation is expected to capture audio information at high resolution, direct generation of such representations using neural networks remains challenging. To address this, more effective intermediate representations

have been developed to simplify the generation task. In particular, latent neural representations have emerged as a promising approach [Re9, Re10, Re11]. These methods use an autoencoder to compress the high-resolution audio signal into a low-dimensional latent space, achieving high compression ratios. In this space, each latent vector corresponds to a short temporal segment of the original audio. For generation tasks, models are trained to synthesize sequences of latent vectors, which are then decoded back into the waveform or spectrogram via the decoder. This design reduces the burden on generative models to capture fine-grained details, instead relying on the autoencoder to preserve relevant information through effective compression.

The representations may remain continuous or be quantized into a discrete set, depending on the structure of the latent space. The quantization process further compresses the latent vectors by mapping them to a finite set of code vectors, each represented with a token. This combination of quantization and autoencoding is referred to as a neural codec, where each audio signal is represented as a sequence of tokens in latent space [Re10, Re11]. These token sequences can then be modeled using a variety of generative models tailored for discrete token generation.

2.2 Music Symbolic Representations

Similar to general audio, music can be represented using formats such as waveforms, spectrograms, or Mel-spectrograms. However, due to the structured nature of musical compositions, it is often beneficial to leverage symbolic representations for music modeling. These representations provide an explicit encoding of notes and other musical events, as exemplified by the Musical Instrument Digital Interface (MIDI) format [Re12]. A digital format is considered symbolic if it encodes information using a fixed alphabet of discrete symbols.

Piano-Roll A straightforward symbolic encoding is the piano-roll representation, which can be viewed as a two-dimensional matrix $M \in \mathbb{R}^{T \times P}$, where the horizontal axis represents time and the vertical axis represents musical pitch. Each entry M_{ij} specifies the velocity of the j-th pitch at the i-th time step. In MIDI terminology, velocity refers to the intensity or

force with which a note is played.

MIDI MIDI representations extend symbolic encoding by recording each note event—such as note onset, note offset, and velocity—as a sequence of messages, analogous to words in a sentence. This text-like structure makes symbolic representations especially well-suited for modeling with large language models. As a result, a wide range of unconditional music generation approaches have adopted various symbolic strategies to model musical sequences [Re13, Re14, Re15].

2.2.1 Visual Representations

Raw Video Representation. A common way to represent video data is by a four-dimensional tensor $\mathbf{v} \in \mathbb{R}^{T \times C \times H \times W}$, where T is the number of frames, C is the number of channels, and H and W denote the height and width of each frame, respectively.

Body Pose Representation. In many cases, raw video contains significant redundancy. For instance, in videos with people, their movements can be effectively captured by focusing solely on body movements, rendering background and appearance details unnecessary. This principle also holds for certain vision-to-audio generation tasks. For example, music accompanying a dance is primarily influenced by the rhythm of the movement, rather than the background scene or the visual appearance of the performer. Therefore, using body movements as the primary representation is both more effective and efficient. Such movements can be extracted using human pose estimation methods, such as OpenPose [Re1]. Specifically, body movements can be encoded as body keypoints, through a three-dimensional tensor $\mathbf{h} \in \mathbb{R}^{T \times J \times D}$, where T is the temporal dimension, J is the number of body joints, and D denotes the dimensionality of joint coordinates (e.g., 2D or 3D).

Deep Neural Representations in Latent Space. Beyond raw videos and body keypoints, one can leverage latent representations extracted from pre-trained visual models. For example, image classification architectures such as ResNet [Re16] and Inception [Re17] are commonly used to obtain compact feature embeddings $f_t \in \mathbb{R}^{D_f}$ for each video frame, where D_f denotes the feature dimension. Additionally, multi-modal embeddings from mod-

els like CLIP (Contrastive Language–Image Pre-training) are frequently employed across a wide range of downstream tasks including generative applications. Beyond frame-level representations, features from pre-trained video understanding models can be used to capture global temporal context [Re18, Re19, Re20]. In the audio–text domain, contrastive alignment techniques [Re21] are similarly used to learn latent vectors that encapsulate rich semantic information from audio, to be utilized for both understanding and generation tasks.

2.3 Deep Generative Models

Autoregressive Models. One of the primary approaches to audio generation is the autoregressive modeling approach, which learns an explicit probability distribution governed by a prior imposed by the model architecture. The joint probability of an audio waveform $\mathbf{x} = \{x_1, ..., x_t, ..., x_T\}$ can be factorized as a product of conditional probabilities:

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t | x_1, ..., x_{t-1}). \tag{2.1}$$

Each audio sample x_t is conditioned on the samples at all preceding samples. Autoregressive models are widely used in audio generation due to their ability to produce explicit probability densities, offer stable training, and align naturally with the sequential nature of audio signals. To implement such models, architectures based on 1D-convolutions [Re22], transformers [Re23], or recurrent neural networks [Re24] have been proposed.

More recently, neural codecs have emerged as powerful tools for compressing raw audio into a downsampled sequence of discrete tokens. This allows autoregressive models to operate at the token level rather than directly on the raw waveform, significantly simplifying the learning process. For example, FoleyGen [Re25] employs a transformer-based architecture to perform autoregressive generation by flattening audio tokens. Other works, such as VampNet [Re26] and SoundStorm [Re27], introduce an advanced autoregressive technique known as masked parallel decoding, in which selected masked tokens are predicted in parallel to improve generation efficiency.

GANs A Generative Adversarial Network (GAN) [Re28] is another useful approach

that has demonstrated impressive results across a range of generative tasks, including music synthesis [Re29]. GANs are inspired by game theory, in which two models, a generator and a discriminator, compete in a zero-sum game, gradually improving each other's performance. The GAN framework consists of two components: a discriminator D, which estimates the probability that a given sample comes from the real data distribution, and a generator G, which generates synthetic samples from a noise input z. The discriminator is trained to distinguish real samples from those generated by G, while the generator is trained to produce realistic outputs that can fool the discriminator into assigning a high probability to fake data. During training, the generator G and discriminator D engage in adversarial learning: G attempts to generate data that appears real, while D tries to detect fakes. This dynamic encourages both models to improve. Mathematically, the discriminator aims to maximize the objective

$$\mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))], \tag{2.2}$$

where $p_r(x)$ denotes the real data distribution and $p_z(z)$ is the prior distribution over noise inputs. Simultaneously, the generator is trained to minimize

$$\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]. \tag{2.3}$$

Combining both objectives, the GAN training process can be expressed as a minimax game:

$$\min_G \max_D \mathcal{L}(D, G) = \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]. \tag{2.4}$$

While GANs have achieved success in various domains, their training can be slow and unstable. In practice, hybrid approaches which combine autoregressive models with adversarial training are often used to improve the quality and the realism of generated samples.

Mask Token Generative Modeling Autoregressive (AR) models factorize the joint distribution over a token sequence $\mathbf{Y} = \{y_i\}_{i=1}^N$ as

$$p(\mathbf{Y}) \;=\; \prod_{i=1}^{N} p\Big(y_i \mid y_{<i}\Big),$$

and generate one token at a time. While AR decoding is conceptually simple, it requires N sequential passes through the Transformer, resulting in $O(N)$ latency. Masked-token generation generalizes AR by predicting—and "committing"—multiple tokens in each iteration, exploiting bidirectional context to unmask a subset of positions in parallel. At each step, the model fills all masked slots simultaneously, then selects the most confident predictions to fix, and repeats this process until no masks remain.

A broad class of generative frameworks, across modalities (e.g., text, images, audio), thus operates by learning to recover randomly masked discrete token sequences via parallel iterative decoding. Let $\mathbf{Y} = \{y_i\}_{i=1}^N$ denote the tokens (from a finite vocabulary of size K) obtained by a domain-specific tokenizer (e.g., text embeddings or VQ-VAE tokens for images/audio), and let $\mathbf{M} = \{m_i\}_{i=1}^N \in \{0,1\}^N$ be a binary mask indicating which positions are concealed. A bidirectional Transformer p_θ is trained to minimize the masked token prediction loss:

$$\mathcal{L}_{\text{mask}} = -\mathbb{E}_{\mathbf{Y},\mathbf{M}}\Big[\sum_{i=1}^{N} m_i \, \log p_\theta\Big(y_i \mid \mathbf{Y} \odot (1-\mathbf{M})\Big)\Big], \tag{2.5}$$

generalizing the Masked Language Modeling objective [Re30] to any discretized signal domain.

At inference it is initialized as

$$\mathbf{Y}^{(0)} = [[\text{MASK}], \ldots, [\text{MASK}]], \quad \mathbf{M}^{(0)} = \mathbf{1}_N,$$

and for each $t = 0, \ldots, T-1$ perform the following steps in sequence:

First, the model predicts logits for every position:

$$\mathbf{P}^{(t)} \;=\; p_\theta\Big(\cdot \mid \mathbf{Y}^{(t)}, \mathbf{M}^{(t)}\Big)\,. \tag{2.6}$$

Next, we sample candidates at each masked position and compute their "confidence":

$$c_i^{(t)} \;=\; M_i^{(t)}\; P_i^{(t)}\Big(y_i^{(t+1)}\Big) \tag{2.7}$$

The number of tokens to commit is decided by applying a monotonically decreasing schedule $\gamma : [0, 1] \to [0, 1]$:

$$n_t \;=\; \left\lceil \gamma\Big(\tfrac{t}{T}\Big)\, N \right\rceil, \quad \tau_t \;=\; \text{the } n_t\text{-th largest value in } \{c_i^{(t)}\}\,. \tag{2.8}$$

With this, tokens whose confidence exceeds the threshold are committed, and the rest remain masked:

$$(Y_i^{(t+1)}, M_i^{(t+1)}) = \begin{cases} \Big(y_i^{(t+1)}, 0\Big), & c_i^{(t)} \geq \tau_t, \\ \Big([[\text{MASK}]], 1\Big), & \text{otherwise.} \end{cases} \tag{2.9}$$

A common choice is a cosine decay schedule,

$$\gamma(r) = \tfrac{1}{2}\Big(1 + \cos(\pi r)\Big)\,, \quad r \in [0, 1]\,, \tag{2.10}$$

which ensures that early iterations unmask only the most confident tokens, and later iterations refine the remainder. After T iterations one obtains $\mathbf{Y}^{(T)}$, the fully generated sequence [Re31].

This Mask Token Generative Modeling paradigm has been instantiated in natural language generation [Re31], image generation [Re32, Re33], and audio token generation [Re27, Re34], demonstrating its modality-agnostic efficacy and significant decoding speedups over autoregressive approaches.

2.4 Conditional Audio Generation

Visual-to-Audio Generation task has drawn significant attention since generative frameworks such as diffusion and transformer-based architectures have been developed. Existing Visual-to-Audio generation approaches can be divided into two branches of studies based on audio categories: *visual-to-music* generation and *visual-to-natural sound* generation. In visual-to-music generation domain, earlier studies explored Midi or spectrogram generation from human body movements by studying the temporal and semantics alignment [XL1, XL4, XL3] [Re35]. More recently, diffusion-based methods have been proposed to generate music waveforms directly from videos [Re36]. In visual-to-natural

sound generation, earlier efforts pioneered the generation of sounds linked to various objects and materials [Re37]. Later works proposed an audio generation approach based on SampleRNN [Re38, Re24] that could generate several types of natural sounds from in-the-wild videos. While these approaches showcase promising results, they are often limited to specific audio categories. Neural codec [Re9, Re39, Re11, Re40, Re10] and autoregressive transformer architectures [Re41, Re42] addressed these limitations and as they have evolved, generative models now effectively generalize across a broader range of sounds or music, leveraging compressed latent spaces [Re43, Re44, Re25, Re45]. Similar advances have been shown with diffusion techniques such as [Re46, Re47]. In the discrete token modeling space, masked-token generative apparoach has also gained popularity since such method can efficiently generate the audio without excessive iterations [Re48, Re34], compared to the common autoregressive decoding schemes.

Text-to-Audio Generation As an alternative to the generation of audio soley from video, text can be used as an input to guide audio generation. When text is the input, audio generation becomes more controllable semantically. Existing approaches such as Make-An-Audio [Re49], AudioLDM [Re50], AudioLDM-2 [Re51] and others [Re52, Re53, Re54] enable general text-to-audio (or music) generation by adapting latent diffusion techniques, first developed in [Re55]. Concurrently, methods such as AudioGen [Re56], MusicGen [Re57], AudioLM [Re58], MusicLM [Re59], SoundStorm [Re27], VampNet [Re26] leverage transformer-based architectures and token-based modeling techniques to produce audio tokens, that are then decoded into waveforms using neural codecs like Encodec [Re40] and SoundStream [Re10]. In particular, SoundStorm and VampNet use an efficient technique known as masked token-based modeling which speeds up generation with parallel unmasking in the decoder.

2.5 Audio-Visual Learning for Acoustics

Each environment exhibits a distinct auditory experience. To understand how sound is realized in a space, a room impulse response (RIR) is typically recorded. This captures

how an ideal impulse emitted from a source reflects, gets absorbed, scatters, and ultimately reaches a microphone, all shaped by the room's specific geometry and surface characteristics. Early machine learning methods for room impulse response (RIR) prediction, such as Image2Reverb [Re60] and Fast-RIR [Re61], utilize a generative approach conditioned on semantic information like RGB images of the environment, source and listener locations, and T60 values. Recent advances on implicit neural representations [Re62, Re63] have inspired a series of works that approximate a function mapping spatial coordinates of source and listener locations to RIRs [Re64, Re65, Re66, Re67]. Some methods [Re68, Re69, Re70, Re71, Re72, Re73, Re74] also condition on room geometry and material properties of the visual environment. By explicitly modeling the 3D scene, these methods can render precise RIRs at novel locations within the same environment they are trained on. In addition to RIR prediction task, prior methods have combined both modalities for a series of other audio-visual learning tasks related to room acoustics, including audio spatialization using visual spatial cues from the environment [Re75, Re76, Re77, Re78, Re79], learning image features, scene structures, or human locations from echoes [Re80, Re81], ambient sound [Re82], or music [Re83] in the room, and using RGB images or videos of a target environment to guide sound transfer that aligns with the space's acoustics [Re84, Re85, Re86, Re87, Re88].

2.6 Multi-modal Large Language Models

Multi-modal Large Language Models (MLLMs), have been able to attain significant progress. With the advent of open source, pretrained and instruction-tuned LLMs such as LLama [Re89], Alpaca [Re90], Vicuna [Re91]. In particular, when extending these LLMs into MLLMs, a pretrained modality-specific encoder extracts the features and then a projection layer maps these features into vectors of the same dimension as text embeddings of the corresponding LLM. This approach led to developments in visual LLMs [Re92], audio LLMs [Re93, Re94], audio-visual LLMs [Re95] and showed improvement in multi-modal understanding tasks such as captioning and question-answering [Re96]. Recent efforts have also focused on specific tasks such as multi-modal retrieval [XL8], multi-modal embodied navigation [Re97], leverag-

ing LLM's strong reasoning capabilities to interpret or improve the results for specific tasks. In terms of generation, several works [Re98, Re99] aimed at achieving any-to-any modality generation using LLMs as a central medium.

Chapter 3

PIANO PERFORMANCE VIDEO-TO-MUSIC GENERATION

This chapter is based on the following published work: [XL1].

3.1 Motivation

Reconstructing musical audio from visual input presents a challenging problem at the intersection of computer vision and audio synthesis. In the case of piano performance, a top-down video contains rich visual information about hand movements, finger positions, and key presses, all of which are directly linked to the resulting sound. Learning a mapping from this visual data to the corresponding audio requires understanding both the temporal alignment and the physical interactions that generate sound.

Unlike tasks where audio is directly available, generating audio from video alone must address the lack of direct acoustic cues. The system must infer timing, pitch, and dynamics from visual observations, despite the lower temporal resolution of video compared to audio. This requires not only synchronizing events between modalities but also predicting fine-grained acoustic features that may not be explicitly visible in the input.

To address this, it is useful to introduce an intermediate representation between video and audio that captures the underlying musical structure. MIDI, a symbolic format encoding key events and timing, provides a compact and interpretable interface for music generation. By learning to predict MIDI sequences from video, the system can decouple the perception of musical actions from the synthesis of audio. This two-stage process—video to MIDI, and MIDI to audio—simplifies learning and improves generalization. Alternative intermediate representations, such as spectrograms, can also be used to bridge the gap between visual

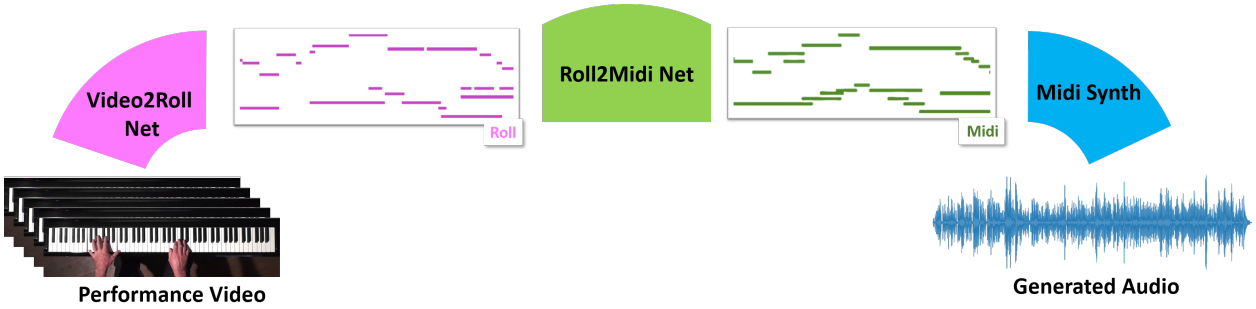

Figure 3.1: Given an input of video frames of a musician playing the piano, ***Audeo*** generates the music for that video. (Figure from [XL1])

input and audio output, offering a dense time-frequency view of sound that complements symbolic approaches.

In light of these observations, I proposed an entire pipeline, named ***Audeo***, to generate the piano music of a silent piano performance video. ***Audeo*** translates the performance from the video domain to the audio domain in three stages by recovering Midi signals. In the first stage, given a top-view video, I use a multi-scale feature attention deep residual network to capture the visual information and to predict which keys are pressed at each frame (**Video2Roll Net**). I formulate this as a multi-label classification task, and the collection of predictions can be seen as a 'Piano-Roll.' However, long-term temporal dependencies are not considered in the first stage and therefore cannot generate faithful music. Therefore, in the second stage, I utilize a Generative Adversarial Network (GAN) [Re28] to refine and enhance the Roll with musical attributes to output the Midi signal (**Roll2Midi Net**). This step turns out to be critical for providing symbolic musical representation. The third and last stage of the ***Audeo*** pipeline is the synthesis of Midi to the audio signal (Midi Synth). Since the predicted Midi is binary and missing expressive velocities, I propose using the same velocity to synthesize mechanical audio via a classical Midi synthesizer or a deep synthesizer to obtain more realistic audio. The deep synthesizer translates Pseudo-Midi to a spectrogram and then to audio. An overview of the ***Audeo*** system is shown in Figure 3.1.

3.2 Methods

I use Midi as an intermediate signal to translate piano video frames to output audio. Since piano performance videos from the Internet are usually without accompanying Ground Truth Midi, I retrieve the Pseudo Ground Truth (GT) Midi from the audio with the Onset and Frames framework [Re100]. The Pseudo GT Midi can be considered a two-dimensional binary matrix $M \in \mathbb{R}^{K \times T}$ where K is the number of pitches, and T is the number of frames. For each entry, $M_{k,j}$, 1 indicates if the key k is sustained at frame j and 0 otherwise. We describe the details of each component of the *Audeo* system in the following.

Video2Roll Net: The task in the first stage can be defined as a multi-label image classification problem. One video clip X can be seen as a four-dimensional tensor $X \in \mathbb{R}^{T \times C \times H \times W}$ where T, C, H, W are time, channel, height, and width dimension respectively. I use stacked five consecutive grayscale frames $X_{t-2,t-1,t,t+1,t+2}$ as the input into Video2Roll Net, which outputs a prediction of the keys pressed in the middle frame X_t. Mathematically, I estimate the conditional probability of the keys being pressed at frame t given video frames $X_{t-2:t+2}$. The probability of estimated keys at frame t will be $P(\hat{M}_{:,t}) = P(M_{:,t}|X_{t-2,t-1,t,t+1,t+2})$. I use ResNet18 as the backbone and the architecture takes into consideration the natural phenomena appearing in this task: 1) the visual cues of the sustained keys are relatively small compared to other objects in the image such as hands and fingers; 2) at each frame, the pressed keys may correlate due to the concept of musical harmony so some combinations have a higher chance to appear at the same time than others; 3) the spatial dependencies are significant to detect the sustained keys but the typical CNN is designed to be invariant to spatial positioning. To address these issues, I design a multi-scale feature attention network similar to [Re101]. Specifically, using ResNet18 as the backbone, Video2Roll Net contains three functional modules: feature transform, feature refinement, and correlation learning. The feature refinement setup is similar to a feature pyramid network (FPN) [Re102], which uses a top-down features propagation mechanism. The main difference to common FPN is that in our **Video2Roll Net** multi-scale features at residual blocks are first transformed and

re-calibrated via the feature transform module before passing to the next stage. This allows the network to detect visual cues on various scales better. As a final component, the correlation learning module is used to learn feature spatial dependencies and semantic relevance by the self-attention mechanism. The response of any location to attention is related to the features of other locations. Since detecting pressed keys is essential to generate meaningful music, the multi-scale feature attention strategy enables Video2Roll Net to find the region of visual cues associated with pressed keys more accurately.

Roll2Midi Net: The prediction of the Piano Roll (Roll) $\hat{M}_{:,1:T}$ of Video2Roll Net is not perfect due to various challenges. For example, hand occlusions in video frames pertain Video2Roll Net from detecting changes in pressed keys. Moreover, because $\hat{M}_{:,t}$ is predicted at each frame individually, Roll predictions do not have a temporal correlation. Also, since Pseudo GT Midi is generated from the Onset and Frames framework [Re100], which depends on the audio stream, one common phenomenon is that if the performer sustains a key for a sufficiently long time, the magnitude of the corresponding frequency will gradually decay to zero and this key in the Pseudo GT Midi will be marked as off, however, since our Video2Roll Net depends on short-time visual information only, all pressed keys are still considered as active. Hence this prediction will not match the reality of the audio. To mitigate these effects, I introduce a generative adversarial network (GAN) [Re28] to refine and complete the Video2Roll results $\hat{M}_{:,1:T}$ so that the outputs are closer to Pseudo GT Midi. The GAN includes a generator G and a discriminator D. The input of the generator is Roll predictions $\hat{M}_{:,T_1:T_2}$ and each column of $\hat{M}$ is the probability score retrieved from the last fully connected layer of Video2Roll Net after applying a sigmoid function. Using probability scores instead of threshold outputs enables the generator to re-calibrate the probabilities and generate a more robust Pseudo Midi representation. The GAN objective is defined by:

$$\min_{G}\max_{D}\mathbb{E}_{M\sim\mathcal{M}}[\log D(M)]+\mathbb{E}_{\hat{M}\sim\hat{\mathcal{M}}}[log(1-D(G(\hat{M})))]. \tag{3.1}$$

Our generator is a five depths U-Net [Re103] and the discriminator consists of 5 layers CNN. Having a discriminator instead of simply applying a U-Net, allows the model to learn a more

general pattern of the Pseudo Midi and the prediction becomes acceptable once it is 'real' enough. Since the variations of Midi in different music styles are significant, using U-Net only may overfit the training data. I use the Mean Square Error (MSE) to optimize both the generator and the discriminator. During inference, I pass the Roll representation to the generator and obtain the refined representation (Midi) $\hat{M}_R = G(\hat{M})$. The Roll2Midi Net can boost the correctness of overall predictions and the estimated Midi is sufficient to be synthesized to get meaningful music close to the ground truth.

Midi Synth: Both the Roll and the Pseudo Midi can be synthesized to audio using classical Midi synthesizers. I find it sufficient to get clear, robust, and reasonable music with the predicted Midi. Moreover, the classical Midi synthesizer is flexible and can support creative applications. For example, music with various timbres can be generated using a piano performance video only by setting instruments other than piano during the synthesis step. While interesting results can be obtained at this point, the audio synthesized from classical Midi synthesizers is mechanical since the predicted Pseudo Midi is binary and does not include expressive velocities. Moreover, estimating expressive velocities on the Midi-level requires having a Midi GT, which specifies them. However, the Onsets-and-Frames that I use as the GT generate velocity prediction with insufficient precision. I thereby investigate whether I can generate more realistic music with the Pseudo Midi predictions via deep synthesizers. To do that, I pre-train a PerfNet [Re104] with Pseudo GT Midi M. The PerfNet learns a transformation H between M and the spectrogram S. With the pre-trained PerfNet, I forward propagate the Midi $\hat{M}_R$ to obtain an initial estimated spectrogram $\hat{S}_R = H(\hat{M}_R)$. Note that even though our predicted Pseudo Midi has been refined, a discrepancy between M and $\hat{M}_R$ would still exist, and I find that using PerfNet to learn transformation from $\hat{M}_R$ to S directly can't be generalized. I conjecture that this is due to the sensitivity of the transformation between the Pseudo Midi and the spectrogram, which increases the difficulty in the generalization. To mitigate this problem, I train an additional U-Net to do the refinement on the spectrogram level. This U-Net can be formulated as a function U and I aim to minimize the L1 distance between $\hat{S}_R$ and S: $L_1(\hat{S}_R, S) = \|U(\hat{S}_R) - S\|$. I find

that estimating the initial rough spectrogram first and then performing the refinement on the spectrogram level later leads to better generalization. As the last step, the Griffin-Lim algorithm is used to convert the spectrogram to an audio waveform [Re105].

3.3 Experiments and Results

I evaluate ***Audeo*** pipeline directly on piano performance videos available on YouTube. The minor constraint for data collection is top-view piano performance with a fully visible keyboard. I use videos recorded by Paul Barton[1] at the frame rate of 25fps and the audio sampling rate of 16kHz. The Pseudo GT Midi are obtained via Onsets and Frames framework (OF) [Re100] and all Pseudo GT Midi sets are set as binary and are down-sampled to 25fps.

Evaluation Metrics. For the ***Midi Evaluation***, I evaluate predictions from **Video2Roll Net** and **Roll2Midi Net** by reporting the precision, the recall, the accuracy, and the F1 score on the frame-level defined in [Re106]. To compare with other methods, I reproduce proposed models in [Re107] and test them on our Pseudo Midi Evaluation set. For ***Audio Evaluation***, I use the popular music identification App SoundHound[2] to perform a detection test on the generated music. The detection is marked as a success if SoundHound successfully shows the correct source name of the music and failure if nothing or the wrong source shows up. I report the average detected rate at the segment level.

Midi Evaluation: Table 3.1 shows the results of ***Audeo*** on the generation of the Roll and the Pseudo Midi compared to other methods. The **Video2Roll Net** detects detailed visual cues that result in a higher recall, accuracy, and F1 score compared to previous works. It turns out that having fewer false negatives is essential to generate a complete melody without missing the notes. The relatively low precision of **Video2Roll Net** reflects the fact that mismatches in audio-visual information are a common phenomenon. Thereby, I do expect false positives in the predictions. The results indicate that to get a cleaner

[1] `https://www.youtube.com/user/PaulBartonPiano`

[2] `https://www.soundhound.com/`

Model	Precision	Recall	Accuracy	F1-score
ResNet [Re107]	64.3	54.7	40.4	49.7
ResNet+Aggregation+slope [Re107]	61.5	57.3	41.2	50.8
Video2Roll Net (Our)	61.2	65.6	46.4	56.4
Roll2Midi Net TS=0.4 (Our)	60.0	**77.0**	50.6	**61.5**
Roll2Midi Net TS=0.5 (Our)	**65.1**	69.9	**50.8**	60.4

Table 3.1: Precision, recall, accuracy and F1-score in (%) for pseudo Midi evaluation. If not specified, all results use threshold (TS) = 0.4 after the application of the sigmoid function. Bold number indicates the best result. (Table from [XL1])

	Total	Bach WTC B1 Variants	Bach WTC B2 & Other
ResNet+FluidSynth	55.9	74.2	52.9
Roll+FluidSynth	62.6	79.6	59.6
Midi+PerfNet	73.0	80.6	71.6
Midi+FluidSynth	**73.9**	**85.6**	**72.4**
Ground Truth	89.2	92.6	87.7

Table 3.2: Sound Hound music identification rate in (%). (Table from [XL1])

and more robust symbolic representation **Roll2Midi Net** is indeed necessary. The core of the generative adversarial network enables Roll2Midi Net to partially eliminate both false negatives and false positives by judging whether the generated Pseudo Midi is real enough. Indeed, **Roll2Midi Net** boosts the overall performance even further. The F1 score of Roll2Midi outperforms the best model in [Re107] by more than 10%.

Audio Evaluation on Music Identification: I compare the detection by SoundHound of samples generated from ***Audeo*** system to the ResNet baseline and the ground truth audio. Furthermore, I synthesize Roll and Pseudo Midi obtained from ***Audeo*** via FluidSynth or

PerfNet to test and exclude synthesizer effects. The results of music identification are shown in Table 3.2. It turns out that all ***Audeo*** methods outperform the ResNet baseline and synthesizing Pseudo Midi via FluidSynth or PerfNet. This indicates that ***Audeo*** can capture the core of learned music and is not sensitive to variance in performance. For test videos from the type not introduced in training, such as Scott Joplin, while the gap with the ground truth (72.4 vs. 87.7%) is still obvious, the identification results demonstrate the robustness and generality of the ***Audeo*** system. In terms of total average, Midi+FluidSynth performs better than other methods and outperforms the ResNet baseline by 18%. Notably, using PerfNet as synthesizer results in slightly lower detection than FluidSynth in this test. While deep synthesizer may recover emotion and naturalness in the spectrogram domain, it also introduces noise which is non-trivial to reduce.

Chapter 4

RHYTHMIC SOUNDTRACK GENERATION FROM VARIOUS HUMAN BODY MOVEMENTS

This chapter is based on the following published work: [XL2].

4.1 Motivation

Sound plays an essential role in enhancing how humans perceive and interpret visual information. In many natural and human-made environments, rhythmic patterns such as footsteps, clapping, or machinery are closely associated with repetitive or structured physical actions. In human activity videos, rhythmically aligned audio can reinforce temporal structure, highlight motion patterns, and increase the expressiveness of the scene. This is especially evident in professional video production, where rhythmically synchronized soundtracks are often carefully selected or manually composed to enhance the alignment between motion and audio.

Automating the generation of rhythmic soundtracks for human activity videos presents a unique challenge. It requires the system to infer temporal patterns in movement and translate them into rhythmic audio that is perceptually coherent and musically plausible. Unlike general-purpose music generation, this task is constrained by the motion dynamics visible in the video, such as periodic gestures or dance movements. The goal is to produce audio that not only synchronizes with these motions but also emphasizes and supports their rhythm.

Given the key role of drums in establishing musical rhythm, drum soundtrack provides an appropriate starting point for this task. Drum tracks often serve as the foundation of

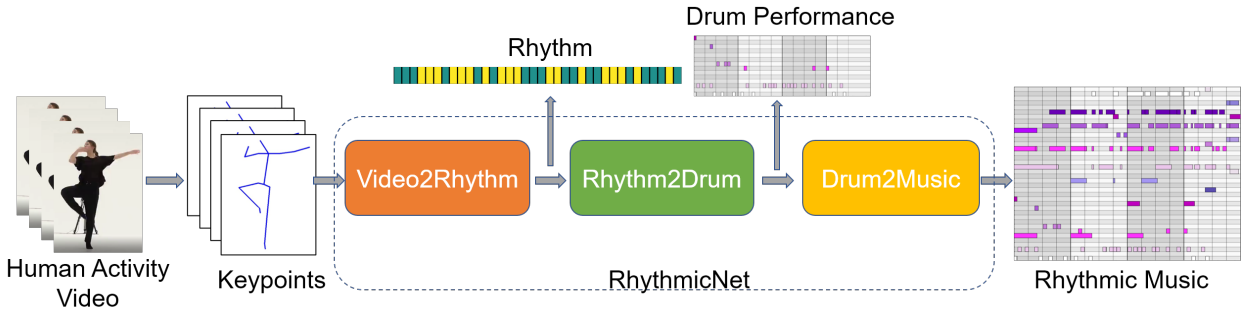

Figure 4.1: System Overview of RhythmicNet. Body keypoints are extracted from human activity video and are processed through the Video2Rhythm stage to generate the rhythm. Rhythm2Drum then converts the rhythm to drum performance. Finally, Drum2Music adds additional instrument tracks (guitar / piano) on top of the drum track. (Figure from [XL2])

musical compositions, and their structure aligns naturally with repetitive motion patterns in human activity. By first generating drum rhythms conditioned on motion cues, the system can establish a temporal base that mirrors the video's content. Additional instrumental layers, such as melodic or harmonic components, can then be introduced in a second stage, allowing for richer and more expressive soundtracks. This approach mirrors conventional music production workflows and provides a structured path toward aligning motion and music in a semantically meaningful way.

In Chapter 3, we showed that visual cues—such as hand movements—can be predictive of musical content, as in piano performance videos. Similarly, human body movements in activity videos often exhibit rhythmic patterns that align well with percussive music. Motivated by this connection, I introduce ***RhythmicNet***, a system that generates rhythmic music to accompany arbitrary human activity videos. In the first stage of ***RhythmicNet***, given a human movement video, we extract the keypoints from the activity video and use a spatio-temporal graph convolutional network [Re108] in conjunction with transformer encoder [Re23] to capture motion features for estimation of music beats. Since music beats are periodic and various visual changes occur in human movements, we propose an addi-

tional stream called the style, which captures fast movements. The combination of the two streams constitutes the rhythm of the movements and guides music generation in the next stage, called Rhythm2Drum. This stage includes an encoder-decoder transformer that, given the rhythm, generates the drums' performance hits and a U-net [Re103] which subsequently generates drums' velocities and offsets. We find that these two stages are critical for generating quality drum music. In the last stage, Drum2Music, we complete the drum music by adopting an encoder-decoder architecture using transformer-XL [Re109] to generate a music track of either piano or guitar conditioning on the generated drum performance. An overview of ***RhythmicNet*** is shown in Figure 4.1. Experiments on datasets of large-scale dance videos and 'in the wild' internet videos show that music generated by ***RhythmicNet*** will be consistent with human body movements in videos.

4.2 Methods

RhythmicNet includes three sequential components: 1) Association of *rhythm* with *human movements* (**Video2Rhythm**), 2) Generation of *drum track* from *rhythm* (**Rhythm2Drum**), 3) *Adding instruments* to the drum track (**Drum2Music**). We describe the details of each stage below.

Video2Rhythm. We decompose the rhythm into two streams: *beats* and *style*. We propose a novel model to predict music beats and a kinematic offsets based approach to extract style patterns from human movements.

Music Beats Prediction. Beat is a binary periodic signal determined by a fixed tempo. It is obtained by a music beat prediction network, which learns the beat by pairing body keypoints with ground truth music beats in a supervised way. To predict regular music beats from human body movements, we extract 2D skeleton keypoints via the OpenPose framework [Re1] and perform first-order differences to obtain the velocity for each video. Motion sequences are considered as three-dimensional tensor $X \in \mathbb{R}^{V \times T \times 2}$ where V is the number of keypoints, T is the number of frames, and the last dimension indicates the 2D coordinates. We formulate the prediction of music beats as a temporal binary classification

problem: Given the skeleton keypoints X, we aim to generate the output with the same length $Y \in \mathbb{R}^T$, where each frame is classified into 'beat' ($y = 1$) or 'non-beat' ($y = 0$).

We encode the keypoints using a spatio-temporal graph convolutional neural network (ST-GCN) [Re108]. Such encoding represents the skeleton sequence as an undirected graph $G = (V, E)$, where each node $v_i \in V$ corresponds to a key point of the human body, and edges reflect the connectivity of the body keypoints. The sequence passes through a spatial GCN to obtain the features at each frame independently, and then a temporal convolution is applied to the features to aggregate the temporal cues. The encoded motion features are then represented as $P = AXW_SW_T \in \mathbb{R}^{V \times T_v \times C_v}$, where X is the input, $A \in \mathbb{R}^{V \times V}$ is the adjacency of matrix of the graph defined based on the body keypoints connections. W_S and W_T are the weight matrices of spatial graph convolution and temporal convolution. T_v and C_v indicate the number of temporal dimensions and feature channels. By averaging the node features, we obtain the final motion features $P \in \mathbb{R}^{T_v \times C_v}$.

Given the motion feature P, we use a transformer encoder that contains a stack of multi-head self-attention layers to learn the correlation between different frames. Due to the periodicity of the music beats, we introduce two components to allow the model to capture them more accurately: 1) We adopt a relative position encoding [Re110] to allow attention to explicitly resolve the distance between two tokens in a sequence instead of using common positional sinusoids to represent timing information. This encoding is critical for modeling the timing in music where relative differences matter more than their absolute values [Re13]. 2) We use the temporal self-similarity matrix of motion features (SSM), which has been shown effective in human action recognition in the regularization of the transformer and counting the repetitions of periodic movements [Re111]. SSM can be constructed by computing all pairwise similarities $S_{ij} = f(P_i, P_j)$ between pairs of frame-level motion features P_i and P_j, where $f(\cdot)$ is the similarity function. We use the negative of the squared Euclidean distance as the similarity function, $f(a, b) = -||a - b||^2$, followed by taking softmax over the time axis. SSM has only one channel, and it goes through a convolution layer $\hat{S} = \text{Conv}(S)$ and

then added to every attention head in the self-attention component implemented as

$$\text{Attention}(Q, K, V) = \text{Softmax}(\frac{QK^T + \hat{S} + R}{\sqrt{D_k}})V,$$

where Q, K, V are the standard query, key, and value, respectively, and R is the ordered relative position encoding for each possible pairwise distance among query pairs and key on each head. We train the model using weighted binary cross-entropy loss that puts more weight toward the beat category to address imbalances. The network's output is the beat activation function; i.e., for each video frame, the model predicts its probability of being a '*beat*' frame. We apply an algorithm based on HMM decoding proposed in [Re112] to obtain beat positions.

Style Extraction. While *beats* represent the monotonic periodic pattern occurring at fixed time intervals (i.e., periodic signal), there are additional a-periodic components in the rhythm. In particular, between two music beats, various irregular movements typically contribute to the rhythm. In contrast to beats, these patterns are inconsistent, and it is unclear how to extract such patterns from visual information systematically. We, therefore, define an additional stream called style, which records incidences of transitional movements of the human body, such as rapid and sudden movements. For the prediction of such events, we apply a rule-based approach since the definition of style is implicit, and there is no data to learn a mapping from body keypoints to transitional movements. The style is defined as a binary stream that indicates transition time points as 1 and non-transitional time points as 0. We compose the style stream by implementing several steps based on spectral analysis of kinematic offsets of the motion [Re113]. The first step is to compute kinematic offsets. Kinematic offsets are 1D time series signals representing the average acceleration of the human body over time. To obtain kinematic offsets, we calculate the directogram of the motion by factoring it into different angles. Given $F_t(j, t)$ as the velocity magnitude of joint j at time

t, we formulate the directogram $D(t, \theta)$ [Re114] as:

$$D(t,\theta) = \sum_j F_t(j,t)\mathbb{1}_\theta(\angle F_t(j,t)), \text{ where } \mathbb{1}_\theta(\phi) = \begin{cases} 1 & |\theta - \phi| \leq 2\pi/N_{\text{bins}} \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

The indicator function $\mathbb{1}_\theta(\phi)$ is used to distribute the motion of all joints into N_{bins} angular intervals. Then the first-order difference of the directogram is calculated to obtain the acceleration of motion across different angles. The mean acceleration in the positive direction measures motion strength (i.e., the larger the value, the more remarkable in motion strength) and corresponds to the kinematic offsets.

Once kinematic offsets are obtained, in the next step, we perform a Short-Time-Fourier Transform (STFT) on them to identify peaks in the change of acceleration. The highest frequency bin in STFT (out of 8) represents the most profound transitions in the signal, and we use the highest frequency bin to extract the style patterns from motion. We mark the time points of the peaks as 1 and other time points as 0. The output signal is re-sampled to have the same sampling rate as the music beats.

Rhythm Composition. We obtain the rhythm by adding the streams of the *beats* and the *style* into a single signal. The rhythm should correspond to the correlation of body movements with the tempo of the soundtrack.

Rhythm2Drum. The stage of **Rhythm2Drum** interprets the provided rhythm from the previous stage into drum sounds. In this stage, we follow the GrooveVAE setup [Re15], where each drum track can be represented by three matrices: hits, velocities, and offsets. The hits represent the presence of drum onsets and is a binary matrix $H \in \mathbb{R}^{N \times T}$, where N is the number of drum instruments and T is the number of time steps (one per 16-th note). The velocities is a continuous matrix, V, that reflects how hard drums are struck, with values in the range of $[0, 1]$. The offsets O is also a continuous matrix that stores the timing offsets, with values in the $[-0.5, 0.5)$ range. These values indicate how far and in which direction each note's timing lie relative to the nearest 16th note. The matrices V, O, and H have the same shape.

Given the input rhythm sequence $Y \in \mathbb{R}^{1\times T}$, we aim to generate the H, V and O. We model H, V, and O smoothly in two steps using the combination of an encoder-decoder transformer [Re23] and a U-net [Re103]. In the first step, the binary rhythm is passed as input to the transformer encoder. In the decoder, the H matrix is converted into a word sequence defined by a small vocabulary set of all possible combinations of hits, and is mapped back to a binary matrix for the final output. We observe that by autoregressively learning the hits H as a word sequence, the transformer can generate more natural and diverse drum onsets. We train the transformer with the cross-entropy loss. In the second step, we add style patterns (velocity and offsets) to the onsets. Since H has the same shape as V and O, we can consider it as a transformation between 2 images of the same shape. To achieve such transformation, we adopt a U-net [Re103] to take the onset matrix H as an input and to generate V and O. We use Mean-Square Error (MSE) loss for U-net optimization. Finally, we convert the generated matrices H, V, and O to the Midi representation to produce the drum track.

Drum2Music. In this last stage, we add further instruments to enrich the soundtrack. Since the drum track contains rhythmic music, we propose to condition the additional instrument stream on the generated drum track. Specifically, we propose an encoder-decoder architecture, such that the encoder receives the drum track as an input, and the decoder generates the track of another instrument. We consider the piano or guitar as additional instruments since these are dominant instruments. We use Remi representation [Re14] to represent multi-track music. Compared to the commonly-used Midi-like event representation [Re13], the Remi representation includes information such as Tempo changes, Chord, Position, and Bar, which allow our model to learn the dependency of note events occurring at the same positions across bars. For both the encoder and decoder, we adopt the transformer-XL network model, which extends the transformer by including the recurrence mechanism [Re109]. The recurrence mechanism enables the model to leverage the information of past tokens beyond the current training segment and to look further into the history.

The encoder contains a stack of multi-head self-attention layers. Its output E_i can be represented as: $E_i = \text{Enc}(x_i, M_i^E)$, where M_i^E is the encoder memory used for the i-th bar input and the encoder hidden state sequence computed in previous recurrent steps. Similarly, in the decoder, the prediction of j-th token of the i-th bar $y_{i,j}$ is formulated as $y_{i,j} = \text{Dec}(y_{i,t<j}, M_i^D, E_i)$, where $y_{i,t<j}$ are the previously generated tokens in the same bar, M_i^D is the decoder memory used for i-th bar, and E_i is the corresponding encoder output of the same bar. The decoder consists of a stack of layers with casual self-attention, cross-attention to the encoder output, and a feed-forward network.

For the training data, we split the music piece into segments with a total number of bars. In the encoder, for recurrent step i, we provide the i-th bar of drum performance x_i to the transformer-XL. We adopt a teacher-forcing strategy and feed the ground truth tokens into the decoder to generate the next tokens. We minimize the negative log-likelihood (NLL) between generated tokens and ground truth tokens to optimize the model. During inference, the drum track is given to the encoder for each bar, and the tokens in the decoder are generated one by one. Finally, we use the temperature-controlled stochastic top-k sampling method [Re115] to randomly generate a new music track.

4.3 Experiments and Results

Datasets. We use the AIST Dance Video Database [Re117], a large-scale collection of dance videos in 60fps for training and testing of **Video2Rhythm**. For **Rhythm2Drum**, we use the Groove Midi dataset [Re15], which contains 1150 Midi files and over $22,000$ measures of drumming. For Drum2Music, we extract two subsets of Lakh Midi dataset [Re118] to separately train **Drum2Piano** and **Drum2Guitar** models.

Video2Rhythm Evaluation. Following the rubrics proposed for musical beat tracking [Re119], we compute the performance in terms of F-score measure, Cemgil's score (Cem), and Correct Metrical Level continuity required/not required ($\text{CML}_{c/t}$) score. To compare with existing approaches, we implement a baseline temporal convolutional network (TCN) for beat prediction [Re116]. The comparison and ablation results are shown in Table 4.1. The

Models\Metrics	CML_c (%)	CML_t (%)	Cem (%)	F (%)
TCN [Re116]	44.97	45.15	48.14	63.04
TF	16.07	16.24	32.85	46.90
ST-GCN	54.89	55.45	49.23	64.78
ST-GCN+TF	61.89	62.34	55.09	71.93
ST-GCN+TF+SSM	63.20	63.58	57.72	73.07
ST-GCN+TF+RelAttn	68.01	68.31	59.19	74.67
ST-GCN+TF+SSM+RelAttn	**71.43**	**71.94**	**61.59**	**75.79**

Table 4.1: Music beat prediction evaluation. The abbreviation of each component stands for: TF (transformer), ST-GCN (spatio-temporal graph convolutional network), SSM (Self-similarity Matrix), RelAttn (Relative Attention); F (F-score measure), Cem (Cemgil's score), CML_c (Correct metrical level continuous accuracy), CML_t (Correct metrical level total accuracy). Bold font indicates the best value. (Table from [XL2])

best method of Video2Rhythm (ST-GCN+TF+SSM+RelAttn) significantly outperforms the baseline model in all metrics by a large margin. In particular, the continuity scores outperform the baseline model by more than 25%, indicating the estimated beat sequence is significantly more consistent.

Rhythm2Drum Evaluation. We use several metrics to evaluate **Rhythm2Drum**. For measuring the diversity of the generated drum hits, we adopt the Number of Statistically-Different Bins (NDB) metric proposed and used in [Re120, Re29, Re35]. For the evaluation of velocities and offsets, we compute the Mean-Squared Error (MSE) for the test set. We compare our methods with the baseline model GrooveVAE [Re15]. The results are shown in Table 4.2. The results show that using hits sequences to generate the drum track enables a more diverse set of samples such that the next U-net component, which generates velocities and offsets, in turn will generate more realistic drumming sounds.

Drum2Music Evaluation. To evaluate the generated piano and guitar tracks, we use objective metrics such as PC/bar (pitch count per bar), PI (average pitch interval), IOI (average inter-

Model\Metric (lower better)	NDB	MSE Velocity	MSE Offsets
GrooveVAE [Re15]	46	0.0437	0.0402
TF multi-outputs w.o. hits sequence	44	0.0507	0.0348
TF multi-outputs w. hits sequence	39	0.0493	0.0369
TF w. hits sequence + Unet	**39** (↓15%)	**0.0267** (↓40%)	**0.0169** (↓58%)

Table 4.2: **Rhythm2Drum** performance evaluation. Abbreviations stand for: TF (encoder-decoder transformer), Multi-outputs (Predict the hits, velocities and offsets simultaneously), w./w.o. hits sequence (whether using word tokens to represent the hits). Bold font indicates the best value. (Table from [XL2])

onset interval) described in [Re121]. For these metrics, we compare the statistics calculated on the test dataset and on the generated music. For additional metrics of PCH (pitch class histogram) and NLH (note length histogram), we calculate the overlapping area (OA) between the statistics on the test dataset and the generated music for each sample and report the average of them. In addition, we compare the NLL loss based on the validation set. The numerical results are shown in Table 4.3. We compare two versions (with and without using memory) to show the effectiveness of the recurrence mechanism. Our results show that for both Drum2Piano and Drum2Guitar with recurrent encoder-decoder transformer (i.e. with memory), the NLL loss is lower for the validation set and the statistics of the generated samples are much closer to the test dataset than the no-memory counterpart.

Human Perceptual Evaluation of Soundtrack Music. In addition to the objective evaluation of the different components of ***RhythmicNet*** we also performed human perceptual surveys using Amazon Mechanical Turk. These surveys were intended to evaluate the effectiveness of ***RhythmicNet*** generated soundtracks to align with the movements and the extent that the generated soundtrack enhances the overall perception of the video compared with various soundtrack controls.

In the first survey, we asked people (non-experts) to choose the video that they prefer,

Metrics	PC/bar	PI	IOI	PCH ↑	NLH ↑	NLL ↓
Dataset (Piano)	5.48	6.16	0.31	-	-	-
Drum2Piano w.o. memory	7.17	4.63	0.12	0.63	0.52	0.77
Drum2Piano	**6.82**	**5.86**	**0.14**	**0.63**	**0.54**	**0.53**
Dataset (Guitar)	5.33	5.51	0.22	-	-	-
Drum2Guitar w.o. memory	3.54	8.94	0.52	0.56	0.46	0.58
Drum2Guitar	**5.63**	**5.69**	**0.13**	**0.64**	**0.51**	**0.40**

Table 4.3: Drum2Music evaluation. For PC/bar, PI, IOI values, the closer to the dataset the better. For PCH and NLH values, the larger, the better. (Table from [XL2])

including a video without a soundtrack and 3 variations of soundtracks generated by our approach (drums-only or drums with another instrument). Results in Table 4.4 clearly indicate a preference for a video with a soundtrack. Furthermore, interestingly, preference for which instruments are included in the track split almost equally between the 3 provided variations, with a slight preference for tracks with Drums+Piano.

In the second survey, we asked people to answer the question: "In which video the sound best matches the movements?". The given options of the soundtracks were generated soundtracks with Random, Shuffle and ***RhythmicNet*** rhythms. The Random drum track was generated with *Rhythm2Drum* method with a random rhythm with 50% chance to be ON or OFF at each time step. The Shuffle drum track was generated with the Rhythm2Drum method but the order in the rhythm is shuffled. ***RhythmicNet*** option corresponded to the drum track generated with the Rhythm2Drum method. From results shown in Table 4.5(left), we observe a clear indication that the drum tracks generated with our method are chosen to be the best match to the movements more frequently (41.4% (Ours) v.s. 30.8% (Random) and 27.8% (Shuffle).

In an additional survey, we performed a perceptual ablation study to test how the two components, Video2Rhythm and Rhythm2Drum, influence the perception of the soundtrack

	Soundtrack Preference			
	No Soundtrack	Drums Only	Drums + Piano	Drums + Guitar
votes	7.3%	31.2%	32.1%	29.4%

Table 4.4: Soundtrack preference. (Table from [XL2])

Soundtrack match to the video			Soundtrack match to the video (Ablation)		
Random	Shuffle	RhythmicNet	Random + GrooVAE	Video2Rhythm + GrooVAE	Video2Rhythm + Rhythm2Drum
30.8%	27.8%	41.4%	23.3%	33.3%	43.4%

Table 4.5: Soundtracks match to movements in the video. (Table from [XL2])

compared to baseline approaches. Survey results are shown in Table 4.5(right) and suggest that in comparison to the baseline, these two components significantly improve the perception of the soundtrack.

Chapter 5

INTERACTIVE VIDEO-TO-MUSIC GENERATION FROM HUMAN BODY MOVEMENTS

This chapter is based on the following published work: [XL3].

5.1 Motivation

In Chapter 4, the generated music soundtracks are well-aligned with the body movements, however, they could not be obtained in real-time. Instead of merely selecting from a predefined set of soundtracks generated a system, we need a tool that has the potential to generate soundscapes dynamically, adapting in real-time to movements of the user. This motivates the system that we propose in this work, ***InteractiveBeat***. Beyond generating soundtracks for various movements, the system provides an immersive experience of real-time interaction with the soundtrack, where any person can create rhythmic sound effects with their bodies. *InteractiveBeat* is a first-of-its-kind learning-based real-time vision-based system for rhythmic drum sound generation in response to human body movements being captured by a video camera. InteractiveBeat introduces (i) a learning-based approach that redesigns the traditional motion rhythm extraction algorithm (offline visual beat detection), enabling its seamless transition to a real-time operation, (ii) a style transfer module that maps motion rhythm to drums rhythm, (iii) a compact polyphonic drum generative model that translates rhythm to drum sounds. An overview of InteractiveBeat is shown in Figure 5.1. To complete the pipeline, we integrate real-time motion-estimation as the system's front-end, and design a producer-consumer workflow that includes updating rules to support real-time improvisation. Each component is implemented by compact networks and is able to run in

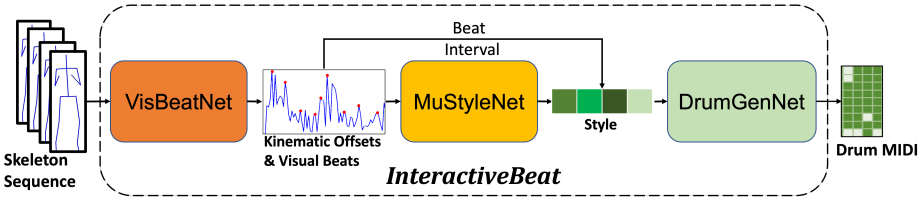

Figure 5.1: System Overview: Three stages of InteractiveBeat: (i) VisBeatNet for prediction of kinematic offsets and visual beats to estimate beat interval, (ii) MuStyleNet transfers kinematic offsets to 'style,' a vector representing drum rhythm, (iii) DrumGenNet translates 'style' to Drum MIDI in the next beat interval. (Figure from [XL3])

real-time.

5.2 Methods

We design our system, InteractiveBeat, to generate rhythmic drum sounds with core objectives of being real-time, aligned with human body movements, drum sounds with coherent rhythmic structures. To meet these objectives, InteractiveBeat consists of three neural network components:

- **VisBeatNet** predicts kinematic offsets, visual beats, and estimates tempo from a live stream of human motion.
- **MuStyleNet** transforms kinematic offsets into a drum 'style.'
- **DrumGenNet** synthesizes a polyphonic drum track based on the estimated tempo and inferred drum 'style.'

A real-time producer-consumer pipeline integrates the above network components with the motion estimation front-end.

This design is different than related recent work of RhythmicNet [XL2]. In particular,VisBeatNet and MuStyleNet components, as described below, implement a more effective way to bridge the motion rhythm and drum rhythm than 'Video2Rhythm' in RhythmicNet, and work in real-time. Furthermore, DrumGenNet adopts a similar network structure as the

first stage of 'Rhythm2Drum' component of RhythmicNet, but with a more compact design to meet the real-time requirement.

VisBeatNet To predict motion rhythm and to estimate the tempo, we develop a novel approach, *VisBeatNet*. This is inspired by an optical flow-based visual beats prediction method [Re114]. While the original method relies on offline operations such as windowing, filtering, and dynamic programming optimization, we adapt it for real-time applications. Specifically, *VisBeatNet* employs a compact neural network that is trained to predict visual beats in real-time. The training uses ground truth derived from a robust pre-computation of visual beats. In the following, we review the background of **visual beats pre-computation** and then describe **VisBeatNet**, our solution for real-time application.

Visual Beats Pre-computation We use a real-time human pose estimator (OpenPose) [Re1] to extract the 2D skeleton key-points from a real-time video stream, and pre-compute the visual beats ground truth via: computing the Directogram, converting the Directogram to Kinematic Offsets, and performing dynamic programming to obtain the Visual Beats. Each stage is detailed below.

- **Computing the Directogram from Skeleton Sequence:** Skeleton sequence is considered as a three-dimensional tensor $S \in \mathbb{R}^{T\times J\times 2}$ where T is the number of frames, J is the number of keypoints, and the last dimension indicates x and y coordinates. By computing the first order difference of this 3D skeleton tensor, $\Delta S_t = S_t - S_{t-1}$, we capture motion at each frame. Using polar coordinates of the last dimension and splitting the full circle $(0, 2\pi)$ into N equal bins, we assign the motion magnitude of every key point into one of the bins according to its motion angle. The motion magnitudes of each bin are summed to obtain the Directogram $D_G(t, \theta)$

$$D_G(t,\theta) = \sum_j \Delta S_t(j)\mathbb{1}_\theta(\angle S_t(j)), \text{ where}$$

$$\mathbb{1}_\theta(\phi) = \begin{cases} 1 & |\theta - \phi| \le 2\pi/N_{\text{bins}} \\ 0 & \text{otherwise} \end{cases}$$

- **Converting the Directogram to Kinematic Offsets:** Kinematic Offsets represent motion changes according to deceleration. The deceleration is computed through the negative first order difference of the Directogram ΔD_G to obtain Motion Flux M, which represents the deceleration in various directions. Low-pass filters are applied to M to filter noise. To find the Kinematic Offsets, negative values in M are removed and the mean over each frame is computed which constitutes K. Top 1% peaks in K are then used and normalized to $[0, 1]$ range to obtain the smoothened Kinematic Offsets.

- **Obtaining the Visual Beats from Kinematic Offsets:** Kinematic Offsets, a continuous signal, is converted to a *binary sequence* that indicates whether there is a significant change in the human motion. We call the sequence *Visual Beats*. Using dynamical programming with the objective 'beat score function' the Visual Beats are computed by finding a set of local peaks of Kinematic Offsets having close or equal interval

$$V(\mathbf{m}) = \sum_{j=1}^{n} u(m_j) + \alpha \sum_{j=1}^{n-1} V_T(m_j, m_{j+1}), \tag{5.1}$$

$$V_T(m_j, m_{j+1}) = \frac{T[\text{bin}(m_{j+1} - m_j)]}{T_{max}} - 1.0, \tag{5.2}$$

  where u is the Kinematic Offsets value of the candidate beat to encourage strong visual impacts. $\{m_j\}_{j=1}^{n} \in \mathbf{m}$ is a subset of candidate beats. $V_T(m_j, m_{j+1})$ penalizes the deviation from optimal tempos within a local window to encourage equal-spacing beats and it computes time-dependent autocorrelation function T on Kinematic Offsets to measure the deviation. α balances the weight between two terms and T is the autocorrelation average within local time window.

VisBeatNet: A real-time network for visual beats We introduce VisBeatNet, a light-weight neural network designed to predict Kinematic Offsets and Visual Beats from a motion sequence. This architecture is built upon two uni-directional Gated Recurrent Units (GRUs), RN1 and RN2.

- **RN1 (Kinematic Offsets Prediction)**: RN1 predicts the Kinematic Offsets K autoregressively from Directogram D_G as the given context.

- **RN2 (Visual Beats Prediction)**: RN2 takes both Kinematic Offsets K and residual connection from hidden states of prediction window in RN1 as inputs and outputs the Visual Beats P_b distribution via a linear prediction head.

Training. We train VisBeatNet using the pre-computed Kinematic Offsets and Visual Beats as theground truth. The training applies teacher-forcing to RN1, and employs two loss terms: (i) Mean Square Error (MSE) between the predicted Kinematic Offsets and the ground truth, and (ii) Weighted binary cross-entropy between the predicted beat distribution and the actual Visual Beats.

Post-processing using B-HMM. After training, the Visual Beats distribution P_b is processed by a pre-built beat Hidden Markov Model (B-HMM). Utilizing the Viterbi algorithm, the B-HMM yields the final output: a list of beat times T_b, specifying when each beat occurs.

While Visual Beats capture moments of strong visual impact in human motion, they inherently lack the drum rhythm, resulting in unnatural sounds when they are directly translated to audio, i.e the direct approach, termed **Mono**, directly overlays a monophonic drum sound atop of the visual beats, utilizing an auto-regressive drum notes generator. To address this limitation, we propose a refined approach which achieves more natural drum sounds, termed **Poly**, which periodically updates the tempo with VisBeatNet. Subsequently, a polyphonic drum language model conditioned on 'style,' a learning-based drum rhythm, are applied.

MuStyleNet: Style Transfer for Drum Rhythm The essence of drum audio lies in its rhythm - the pattern of drum onsets within each frame. In a realistic drum track every frame contains multiple onsets of different drum kits. Extracting the strongest onset in each frame gives rise to what we define as the drum 'style', a vector representing the rhythm of the drum audio.

To obtain the'style', it is crucial to establish a relationship between 'style' and kinematic offsets, which represent the rhythm of body movements. There are multiple plausible ways to transform these offsets into the drum 'style'. A straightforward method was presented

in 'RhythmicNet'. This method combined predicted music beats with motion peaks based on spectral analysis. Since drum rhythms contain regular patterns, while motion rhythms do not, it is unclear how to obtain associated 'styles'. Indeed, the crude approximation for the 'styles' can result in unnatural drum rhythms when used as the conditional input to the drum generation network.

Towards this end, we propose an adversarial style transfer module, *MuStyleNet*, which learns to translate the Kinematic Offsets into drum 'style' using a Generative Adversarial Network [Re28]. The network takes kinematic offsets $K = \{K_1, K_2, ..., K_t\}$ as input and outputs the corresponding onset envelope $O = \{O_1, O_2, ..., O_t\}$. The generated envelope is given to the discriminator component to decide whether the envelope is real or fake. The GAN objective is defined by:

$$\min_G \max_D \mathbb{E}_{O\sim\mathcal{O}}[\log D(O)] + \mathbb{E}_{\hat{O}\sim\hat{\mathcal{P}}}[log(1 - D(G(K)))]. \tag{5.3}$$

Once the generated onset envelope is obtained, O is accumulated by step size st to obtain $O_{acc}(k) = \sum_{t=T_{s_k}}^{T_{s_{k+1}}} O_t$, where st is determined by the beat interval estimated in VisBeatNet and $T_{s_{k+1}} = T_{s_k} + st$. For consistency with the drum MIDI dataset that is used in the next stage, we fix st to be an interval of a quarter beat, and normalize the compressed onset envelope to obtain the drum 'style.'

Drum Generation DrumGenNet This stage translates the 1D drums matrix, obtained in the previous stage, into polyphonic drum sounds. In contrast to the 2-stage Rhythm2Drum network in RhythmicNet [XL2], we keep the first stage network (with fewer number of layers and hidden size) that translates 1D drum rhythm into 2D drum hits matrix and due to real-time constraints discard the UNet that generates velocity and offset matrices. Furthermore, the 1D rhythm inout is continuous rather than binary, and thereby we use the continuous values as velocity for all drum hits at each time step. These changes make the drum generation network compact with the inference overhead compatible for real-time. We train DrumGenNet using a cross-entropy loss.

Real-Time Pipeline

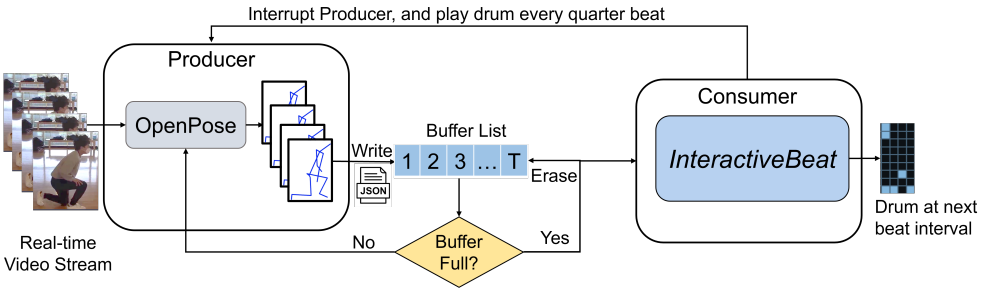

Figure 5.2: Real-Time Pipeline: A two-threaded pipeline with the Producer' and Consumer'. The Producer uses Openpose [Re1] to extract the skeleton sequence from live video and buffers it. When full, the sequence moves to the Consumer thread. The Consumer runs InteractiveBeat to produce drum MIDI for the next beat interval, dynamically updating. The Producer continues to add sequence to the buffer, pausing only for drum sound playback every quarter beat. (Figure from [XL3])

For real-time operation of the system, we integrate all components, i.e., motion estimation, beat inference, style transfer and drum generation, into a single and efficient pipeline. We design the pipeline to work in a producer-consumer mode, where the thread of the producer reads the body keypoints from motion estimation module, and sends them to a buffer list shared with the consumer thread. When the thread of the consumer obtains enough frames with keypoints, InteractiveBeat starts the inference by computing the Directogram and feeding it into the neural network components to generate the 'style'. The 'style' is provided to DrumGenNet to obtain associated drum MIDI. Meanwhile, InteractiveBeat dynamically updates the tempo every beat interval to keep up with the human motion. The InteractiveBeat real-time pipeline is illustrated in Figure 5.2.

5.3 Experiments and Results

Validation of Visual Beats as Ground Truth To validate the effectiveness of using visual beats as ground truth, we conducted a user study comparing pre-computed visual beats against music beats obtained from the audio of the videos. We collected videos from

	AIST dataset		'in-the-wild' dataset	
	Visual Beats	**Music Beats**	**Visual Beats**	**Music Beats**
Votes	46.7%	53.3%	63.7%	36.3%

Table 5.1: Preference of Visual Beats v.s Music Beats. (Table from [XL3])

AIST and 'in-the-wild' for evaluation (30 from each). The question, "Which beats do you think are better in sync with body moves?" is asked. As shown in Table 5.1, Visual Beats are preferred over Music beats. For 'in-the-wild', the preference is of large margin of **+27.4%** and for AIST the preference is of smaller margin of 6.6%. We suspect that the difference in the margin is due to data processing and annotation. AIST includes precise annotations and alignment of well-annotated music beats with dance movements while 'in-the-wild' dataset relies on music beats extracted with libraries, such as Demucs + Librosa, since there are no annotations. This process can introduce errors.

Visual Beats Prediction Comparison Learning-based Visual Beats. To compare visual beats, we train stage 1 of the Video2Rhythm component in RhythmicNet [XL2] and compare it against VisBeatNet. We use pre-computed visual beats as ground truth,. The original implementation of Video2Rhythm uses a bi-directional graph-transformer with SSM modules, which is non-causal and cannot be applied in real-time. Direct adaption would be adding a causal transformer decoder to it that decodes the Kinematic Offsets with auto-regression and a linear layer for Visual Beats estimation. We use a 1-layer decoder with same hidden size and number of attention heads as RhythmicNet. Given a 3-sec input, we predict the next 1 second, and repeat the process for 3 times on a rolling basis to collect 3-sec Visual Beats predictions for evaluation 5.2.

Rule-based Visual Beats. We also implement a rule-based Visual Beats prediction baseline that satisfies the constraints of maximum within a pre-defined window size of 0.25s, minimum time wait after previous peak 0.25s, and above the average within the window than a threshold, which is set to 0.015. These parameters are the same as the ones used to

	Num Params	Inference time	AIST Dataset					'in-the-wild' Dataset				
			Pr ↑	**Rec ↑**	**Cem ↑**	**F↑**	**B-Aln ↑**	**Pr ↑**	**Rec ↑**	**Cem ↑**	**F↑**	**B-Aln ↑**
Rule-based	-	**1ms**	38.62%	57.31%	33.53%	45.10%	0.286	39.25%	56.96%	33.42%	45.7%	0.2813
RhythmicNet[XL2]	800k	90ms	**69.78%**	**49.60%**	45.93%	**59.45%**	**0.4147**	**67.25%**	51.29%	44.52%	**55.70%**	0.4098
VisBeatNet	**30k**	5ms	69.01%	**50.71%**	46.02%	58.19%	0.4032	66.17%	**51.91%**	44.31%	55.43%	**0.4175**

Table 5.2: Visual Beat prediction evaluation on AIST dance dataset(lab environments) and 'in-the-wild' dataset. The abbreviation of each component stands for: Pr(Precision), Rec(Recall), F (F-score measure), Cem (Cemgil's score), B-Alg(Beat Alignment Score). (Table from [XL3])

extract candidate Visual Beats during Visual Beats pre-computation.

Beat Objective Evaluation. We follow the rubrics proposed for musical beat tracking [Re119] to evaluate Visual Beats prediction. We compute the performance in terms of Precision, Recall, F-score measure, and Cemgil's score (Cem). We also compute the Beat Alignment Score[Re122] which measures the correlation between motion and music. As shown in Table 5.2, VisBeatNet achieves on-par performance with RhythmicNet on visual beats prediction, while using only 2 GRU layers (*30k parameters*) with *5ms* inference time compared to the RhythmicNet (*800k parameters*) with *90ms* inference time. In Section 4.6, we show that such inference time introduces large system latency which hinders real-time application. Further, it is noteworthy that the rule-based approach achieves reasonable performance in recall score. However, its precision is low, which generates excessive beats even when no movements appear.

Beat Subjective Evaluation. We compare the predicted visual beats by VisBeatNet against the ground truth visual beats to evaluate the perceptual gap between the prediction and the ground truth. As shown in Table 5.3, the gap is of 11.4% or 6% for AIST and 'in-the-wild' datasets respectively, which demonstrates the relative effectiveness of VisBeatNet. The gap for 'in-the-wild' is smaller than the gap on AIST. The reason is that the pre-computed visual beats for 'in-the-wild' videos are more noisy than AIST videos due to the quality of the

	AIST dataset		'In-the-wild' dataset	
	VisBeatNet	**GT Visual beats**	**VisBeatNet**	**GT Visual beats**
Votes	44.3%	**55.7%**	47.0%	**53.0%**

Table 5.3: Visual beats prediction v.s GT perceptual preference. (Table from [XL3])

body keypoints inputs, but in terms of alignment, visual beats align better with movements than accompanying music beats, as we show in Table 5.1.

Drum Generation Comparison The drums can be generated by whether directly adding monophonic drum notes to the visual beats, or applying a polyphonic drum generative model conditioned on drum 'styles'. For *monophonic* drum, we train a 1-layer GRU model, with hidden size 64 on Groove MIDI dataset to auto-regressively generate monophonic drum notes. *Polyphonic* drums are generated using 'DrumGenNet'. We generate the drum sounds using 'styles' extracted from the videos rather than real drum rhythm extracted from drum tracks. This is important because the drum generator input comes from the rhythm in motion modality rather than drums, while the evaluation of the original 'Rhythm2Drum' stage of 'RhythmicNet' [XL2] ignores this crucial point. To be clear, we name the baseline methods as 'Mono' or 'Poly' method as follows.

Rule-based-Mono uses the rule-based visual beats to generate 'style' patterns, and use 'Mono' to generate drum sounds.

RhythmicNet-Mono use the 'Video2Rhythm' stage of 'RhythmicNet' to generate 'style' patterns, and use 'Mono' to generate drum sounds.

RhythmicNet-Poly uses 'Video2Rhythm' for 'style', and 'DrumGenNet' to generate drums. The tempo is estimated by 'Video2Rhythm'.

InteractiveBeat uses 'MuStyleNet' to generate 'style' from Kinematic Offsets, and 'DrumGenNet' to generate drum. The tempo is estimated by VisBeatNet.

We use two *audio* objective metrics to compare the drum sound quality.

1) FID was introduced to evaluate image quality in GANs. It is adapted in audio-visual

	AIST dataset			'In-the-wild' dataset		
	FID↓	**NDB↓**	**Votes↑**	**FID↓**	**NDB↓**	**Votes↑**
Rule-based-Mono	68	49	3.7%	72	49	3.0%
RhythmicNet-Mono [XL2]	66	49	10.3%	70	49	12.7%
RhythmicNet-Poly [XL2]	47	46	33.7%	49	47	35.3%
InteractiveBeat	**37**	**41**	**52.3%**	**43**	**41**	**49.0%**

Table 5.4: Audio quality metrics(NDB, FID) and soundtrack preference between InteractiveBeat and other baselines. (Table from [XL3])

domain for audio spectrograms of generated soundtracks. It measures the distance between InceptionV3 pre-classification feature distributions for real and generated samples. By adapting InceptionV3 input for a 2D magnitude spectrogram and training it on GrooveMIDI for classifying 12 drum genres, we extract 2048-sized vectors from the last layer for both sets of samples. FID is then computed from these vectors.

2) NDB metric is used to evaluate the diversity of generated samples; the lower the NDB score, the better the diversity. Following RhythmicNet [XL2], We select k = 50 for k-means algorithm to cluster Voronoi cells in log-spectrogram space.

For each baseline, we generate 5000 samples separately from the test set of AIST and 'in-the-wild' to perform the objective evaluations. As shown in Table 5.4, InteractiveBeat achieves better drum quality than 'RhythmicNet'. A polyphonic drum generative model with a clear bar-level structure is necessary to produce quality drum sounds.

We also perfrom a perceptual experiment where we select 30 videos from each of the datasets AIST and 'in-the-wild' and ask the raters to compare the drum sounds for different methods. In particular, we ask: "Which drum track sounds most natural, coherent, and rhythmic?" As shown in Table 5.4, InteractiveBeat consistently outperforms other baselines by a large margin, **+18.6%** and **+13.7%**, for AIST and 'in-the-wild' respectively.

Real-Time System Evaluation To evaluate the real-time system, we use latency as the main metric and evaluate the baselines (Rule-based-Mono, RhytmicNet-Mono,RhytmicNet-

Methods	Causal or Non-Causal(C/NC)	**Total Latency(ms)**↓	**Votes**↑
Rule-based-Mono	NC	158	3.7%
RhythmicNet-Mono [XL2]	NC	133	11.3%
RhythmicNet-Poly [XL2]	C	103	35.3%
InteractiveBeat	C	**34**	**49.7%**

Table 5.5: Real-Time evaluation on InteractiveBeat v.s other baseline methods. (Table from [XL3])

Poly) along with InteractiveBeat. The system latency can be analyzed based on the following aspects:

Camera Frame Rate (CFR): Our off-the-shelf web camera for experiments operates at $CFR = 30fps$.

OpenPose Inference Speed (OIS): On a TitanX GPU, OpenPose achieves $OIS = 70fps$ on AIST and 'in-the-wild' videos.

Network Inference Speed (NIS): This is the inference speed of all networks combined.

Causal v.s Non-causal (C/NC): generate the drum for the present with delay (NC) v.s generate the drum for the next interval, compensating for delay (C).

For *CFR* and *OIS*, *CFR*= 30fps is the minimum possible latency (L_{cam} = 33ms) of the system , while OpenPose achieves faster speed (70fps). For *NIS* and *C/NC*, we describe the integration of different network choices into the real-time pipeline, and analyze the total latency. A summary of total latency is shown in Table 5.5 and we summarize the latency metrics below.

- **Rule-based-Mono**: A non-causal method that directly adds monophonic drums to the 'style'.

  Inference time: $L_{ni} = 125ms$ (visual beats are determined after 125ms, half of the window size (0.25s).

  Total latency: $L_{total} = L_{cam} + L_{ni} = 158ms$.

- **RhythmicNet-Mono**: A non-causal method with the Video2Rhythm stage of 'RhythmicNet' [XL2]. It uses a 3-sec input window for real-time adaptation and checks for visual beats in the latest 60ms.

  Inference time: $L_{ni} = 70ms$.
  'Style' check delay: $T_r = 30ms$.
  Total latency: $L_{total} = L_{cam} + L_{ni} + T_r = 133ms$.

- **RhythmicNet-Poly**: A causal method which compensates for camera frame rate delay by forecasting the style' for the next interval. Apply a 3-sec Video2Rhythm inference window followed by a 1-layer GRU forecaster.

  Inference time: Video2Rhythm & forecaster: $81ms$, 'DrumGenNet': $22ms$.
  Total latency: $L_{total} = L_{ni} = 103ms$.

- **InteractiveBeat**: Our method for causal forecasting visual beats in the next interval.

  Inference time: Directogram calculation ($1ms$), VisBeatNet ($5ms$), MuStyleNet ($6ms$) and DrumGenNet ($22ms$).
  Total latency: $L_{total} = L_{ni} = 34ms$.

As shown in Table 5.5, InteractiveBeat achieves significantly lower latency than other baselines due to compact networks design that reduces inference speed and a causal scheme which compensates for the delay. Notably, 'RhythmicNet-Poly' is an adaption from 'RhythmicNet' with addition of real-time constraint. Our results show that merely adding such constraint without change of network design would still correspond to larger latency.

To further evaluate the operation in real-time in terms of its perceptual experience, we conducted a human study where we generated drum sounds for a total of 30 videos from AIST and 'in-the-wild' set in real-time for each method. A question "Which drum soundtrack do you prefer, considering the alignment with body movements and latency?" was presented

to raters. As shown in Table 5.5, InteractiveBeat receives most votes, higher by **+14.4%** than the second top pipeline of 'RhythmicNet-Poly'.

Chapter 6

MULTI-INSTRUMENT MUSIC GENERATION FROM MUSIC INSTRUMENT PERFORMANCE VIDEOS

This chapter is based on the following published work: [XL4].

6.1 Motivation

The approach in Chapter 3 mainly solves the scenarios where detailed audio-visual correspondence can be found, such as piano performance, and it is limited to a single type of instrumental music. However, in many cases where the details of movements of playing instruments can not be observed, it is impossible to find a one-to-one mapping directly from vision. In this case, I generalize the visual input from single instrument video to different types of instruments by associating musicians' performance movements and the timbre attribute of instruments. I propose to generate multi-instrumental music from videos without labeling the instruments in an completely unsupervised way so as to generalize to as many types of instruments as possible. To achieve that, I introduce a novel approach named 'Multi-Instrumentalist Net' (***MI Net***) that first learns a discrete latent representation of the music of various instruments and leverages human movement of the instrument playing as the condition to drive the music generation. The pipeline is a novel adaptation of a Vector Quantized Variational Autoencoder (VQ-VAE) [Re9] that can encode and generate log-spectrogram audio representations. We train the pipeline with an autoregressive prior conditioned on the musician's body keypoints movements encoded by a recurrent neural network. Joint training of the prior with the body movements encoder disentangles the music into latent features indicating the musical components and the instrumental features. The

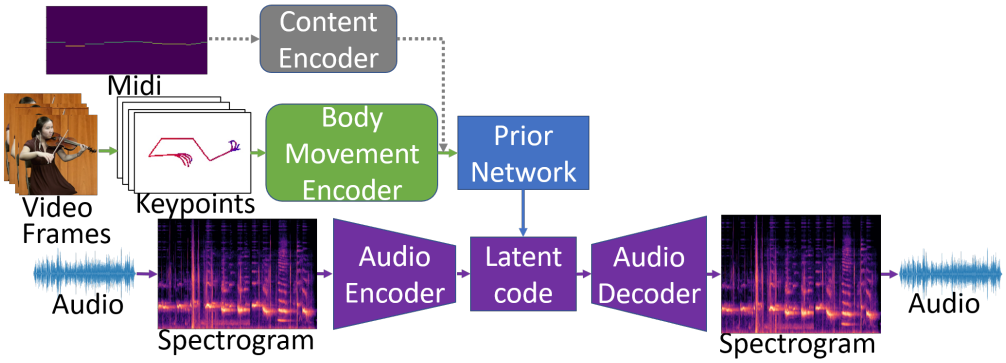

Figure 6.1: ***MI Net*** system overview. VQ-VAE (bottom) is used to reconstruct audio sequences of various instruments and infers a latent representation of music (latent code). VQ-VAE is conditioned by a prior network (middle) that encodes body movements w/wo MIDI content. Further, given an input of Video Frames of the musician playing the instrument, Multi-instrumentalist Net (***MI Net***) generates the music for that instrument. (Figure from [XL4])

latent space generates distributions disentangled into clusters of distinct instruments from which new music can be generated. Furthermore, the VQ-VAE architecture supports detailed music generation with additional content conditioning via Midi to generate the exact content of the music played by each instrument in the video. It thus supports novel mixing applications on the instrument level.

6.2 Methods

Encoding of Audio Representations. We train a neural audio codec architecture to first encode the magnitude of the log-spectrogram into the latent space and obtain the audio representations for conditional audio modeling. We introduce a multi-band residual 1D convolutional Vector Quantized Variational Autoencoder (VQ-VAE with MBR), as shown in Figure 6.1. VQ-VAE [Re9] encoder outputs a discrete latent representation, and the decoder decodes this representation and reconstructs the input. Direct application of original VQ-

VAE to spectrogram is not optimal since the log-spectrogram is of high dimension due to the high resolution of the frequency bins. Therefore, we introduce novel components capable of processing spectrograms within VQ-VAE. In particular, we propose to include a multi-band residual learning method on the audio encoder and the decoder to capture the spectral features of musical overtones better. Such a block was shown effective in Midi to music synthesis [Re104]. The multi-band residual (MBR) block splits the input into a specific number of frequency bands and then feeds each band individually to identical sub-blocks consisting of the following layers: 1D-convolution, ReLU, and 1D-convolution. The output of all sub-blocks is then concatenated along the frequency dimension, and a residual connection sums up the output with the input of the block. In the audio encoder, we progressively divide the spectrogram into more bands in the earlier layers and into fewer bands in the latter layers. The decoder then decodes the latent representation from fewer bands to more bands in a symmetric way. Since both the encoder and the decoder are fully 1D convolutional, the system supports any input length. Our VQ-VAE model incorporates two additional terms in its objective to align the vector space of the code with the output of the encoder. (i) The embedding loss is applied to the codebook variable and e_k and brings the selected codebook $\mathbf{e}$ closer to the output of the encoder $E(S)$. (ii) The commitment loss is applied to the encoder weights, which aims to keep the encoder output as close as possible to the chosen codebook vector to prevent it from fluctuating from one code vector to another. As proposed in [Re9], we use the exponential moving average updates for the codebook to replace the embedding loss. The resulting loss is $\mathcal{L} = \|S - D(e)\|_2^2 + \beta\|E(S) - sg[e]\|_2^2$, where the first term is the reconstruction loss and the second term, β, is a hyper-parameter that depends on the scale of the reconstruction loss and sg stands for the stop gradient operator defined as the identity at forward computation time and has zero partial derivatives.

Encoding Visual Representations and Learning a Prior over the Audio Latent Code. Given a sequence of 2D human pose key-points $P = \{p_1, p_2, ..., p_T\} \in \mathbb{R}^{J\times T}$, where J is the number of joints and T is the total number of time steps, we first encode the key-points into a latent representation. If optimal, the latent space should differentiate movements of

playing different instruments and self-organize itself into separate clusters as shown in my previous unsupervised skeleton-based action recognition approachs [XL11]. To achieve this, we use a bi-GRU as the body movement encoder E_b. Given the input sequence P, the last hidden state of the encoder $h = E_b(P)$ is taken as the global representation of the musician's movement. To associate body movements with audio features, we jointly train the body key points encoder with the prior latent space of the audio features. Superimposed with the latent audio features, the body movement features differentiate the type of instrument performance and other characteristics of the performed music. This allows us to use body movements to generate new music for the corresponding instrument.

The prior distribution over the discrete latent audio features $p(Z)$ is a joint distribution of categorical variables across time and can be learned autoregressively. When training our system, the prior is kept constant and uniform. After training, we fit an autoregressive distribution over Z to generate new samples via ancestral sampling. We use the encoder of transformer structure [Re123] over the discrete latent space. We concatenate the last hidden state of the body keypoints encoder h to every time step of the discrete latent representation. This forces the prior of audio features to align and correlate to body motion features when autoregressively learning to predict the next latent code. Subsequently, the concatenated features are passed through multi-head self-attention and feed-forward layers. The outputs are passed to a softmax layer to predict the probability of the next latent code over the codebook.

Content Conditioning. In addition to the unconditional generation of music, we could generate the exact music content for each instrument in the video by using the Midi matrix $M \in N \times T$ as the content signal. Since instruments considered in our experiments are monophonic, there is a single note at each time step. We first convert the 2D matrix into a binary matrix by ignoring the expressive dynamics (i.e., the loudness of music). We then transform the binary matrix to a 1D sequence containing the activated note's index at each time step. We use a transformer-based encoder-decoder architecture to achieve content conditioning [Re23]. The content encoder is the transformer encoder that inputs

the Midi information. We then concatenate the encoded body movement representation to the embedded Midi at each time step and pass them to two self-attention and feed-forward layers. The content encoder output will go through a fully connected layer to become the conditioned signal C. The transformer decoder includes a cross-attention layer after the self-attention layer to compute the attention between the conditioned signal and the latent code representation. Considering the output of the self-attention module $A \in \mathbb{R}^{T_a \times C}$ and the Midi conditional signal $M \in \mathbb{R}^{T_m \times C}$, where C is the feature dimensions, the cross-attention is defined as: Cross Attention$(A, M) = \text{softmax}(\frac{AM^T}{\sqrt{D_k}})M$. Here, the feature dimension of A and M are designed to be the same. During sampling, we provide both Midi and body movements of each instrument's performance to the content and body movement encoder, respectively. The prior network autoregressively generates discrete latent representations via ancestral sampling.

6.3 Experiments and Results

Dataset. We evaluate the ***MI Net*** on **URMP** dataset [Re124], a high-quality multi-instrument video dataset recorded in a studio. It includes 13 instruments and provides the musical score in Midi format, which we use for evaluation. We further evaluate the MI Net on **Solos** dataset [Re125], a recent dataset of YouTube videos of musicians' recitals. It contains the same 13 instruments as the URMP dataset. Solos also contain pre-processed skeleton key points extracted via Openpose, however, doesn't include Midi files due to the 'in the wild' nature of the videos.

Comparison with Other Models. For comparison, we implement two baselines of current systems: (i) *RNN-based Seq2Seq Network:* An encoder-decoder recurrent neural network. The encoder inputs body key points, and the decoder generates the expected spectrogram. (ii) *Graph-Transformer Network:* A Graph-Transformer network similar to the architecture of Foley Music [Re35]. It is based on Spatio-temporal Graph Convolution Network (ST-GCN) [Re108], which encodes the body key points to pose features, then fed into the Transformer decoder where each block contains self-attention, cross-attention, and feed-forward

modules. Instead of predicting the Midi events, we directly generate the log-spectrograms.

Number of Statistically-Different Bins (NDB). We adopt the metric proposed in [Re120] and used in [Re29, Re35] to measure the diversity of the generated examples. NDB is reported as the number of cells where the number of training examples is statistically significantly different from the number of generated examples by a two-sample Binomial test. For each model, we generate 1600 samples from the testing set and perform the comparison. The NDB results are shown in Table 6.1 and indicates how well the model learns from the training set. The largest non-optimal NDB score is 50. The lower NDB score the better it is. For the URMP dataset, the MI Net outperforms other methods by a large margin. Both the RNN-based Seq2Seq and Graph-Transformer do not generate music with sufficient quality for the unsupervised setup. Thereby, the generated samples of MI Net without conditioning on content have a distribution closer to the training data, resulting in a lower NDB score than the reference itself. For the Solos dataset, the NDB result of our method is better than others, however, not as realistic as for URMP. One of the limitations is that the body motions of 'in the wild' videos have large variations.

Classification with Body Motion. To evaluate whether the encoded pose features are separated to generate exclusive music for specific instruments, we extract the body movement encoder final hidden state and fit K-Nearest-Neighbors classifier ($K = 1$) using the cosine similarity metric. For comparison, we use the encoder final state for RNN-based Seq2Seq, and take the mean of both temporal and joints dimensions of the outputs of ST-GCN for Graph-Transformer to perform the classification. The results are shown in Table 6.2. Notably, for URMP dataset, our method associates the body movements with the instruments at a high accuracy, evident by the score in Table 6.2 and t-SNE plots of the latent representations in Figure6.2. These plots show that the models that we compare against do not distinguish well the instruments. Furthermore, these models appear to separate the instruments according to the difference in pose only and thus cannot generate music across instruments. This is particularly challenging for instruments with similar poses (e.g., Viola v.s. Violin, Oboe v.s. Clarinet). In comparison, our method obtains a latent space that self-organizes visual and

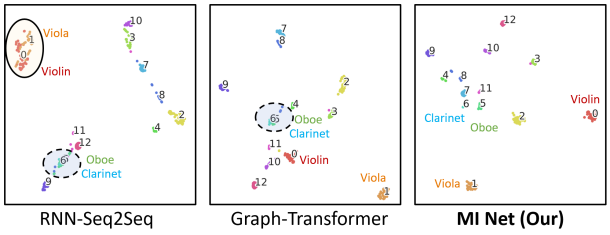

Figure 6.2: T-SNE plots of the encoded body movements representation in URMP test set. the number next to clusters indicates instruments. Dotted circle: Mixing of samples belonging to Oboe and Clarinet instruments. Solid circle: Mixing of samples belonging to Violin and Viola instruments. (Figure from [XL4])

Model	URMP	Solos
RNN-based Seq2Seq	48	44
Graph-Transformer	45	41
MI Net (Our)	**33**	**36**
Content Cond. MI Net (Our)	**35**	-
Testing Set Data	36	31

Table 6.1: NDB results. **Lower is better**. (Table from [XL4])

Model	URMP	Solos
RNN-based Seq2Seq	82.6	37.8
Graph-Transformer	89.2	38.3
MI Net (Our)	**98.7**	**61.5**

Table 6.2: KNN Classification Accuracy in %. (Table from [XL4])

audio events by instruments.

Chapter 7

AUDIO GENERATION FROM ARBITRARY VIDEOS WITH OPTIONAL TEXT PROMPTS

This chapter is based on the following published work: [XL5].

7.1 Motivation

In previous chapters such as Chapter 3, Chapter 6 and Chapter 4, the vision signals come from human motions, including the body movements or hand movements. Such systems are constrained by the existence of human movements. To generalize such system further, we create a vision to sound generation system where the video of any category can be taken as the input and the system generates the audio for that. Therefore, I propose a general video-to-audio generation system called ***VATT*** (shown in Figure 7.1) that can take optional text prompts as inputs to generate the audio that fits both the context of video and the text prompts. VATT consists of two modeling stages: i) Video-to-Caption stage, which converts video features into an audio caption through a pretrained large language model (LLM) with a learnable projection layer. Through this cross-modal conversion, visual features that are relevant to audio concepts are extracted. These features are closely connected to audio-related tasks such as audio captioning and audio generation. ii) Video + Text to Audio stage, that generates audio conditioned on the hidden states extracted from the LLM in the prior modeling stage. At its core, the proposed model in this stage is a bi-directional transformer decoder that generates audio using a token-based representation similar to [Re32, Re27]. To obtain the conditioning on the hidden states of the preceding component, the projected video features along with the optional text prompts are concatenated together and fed into

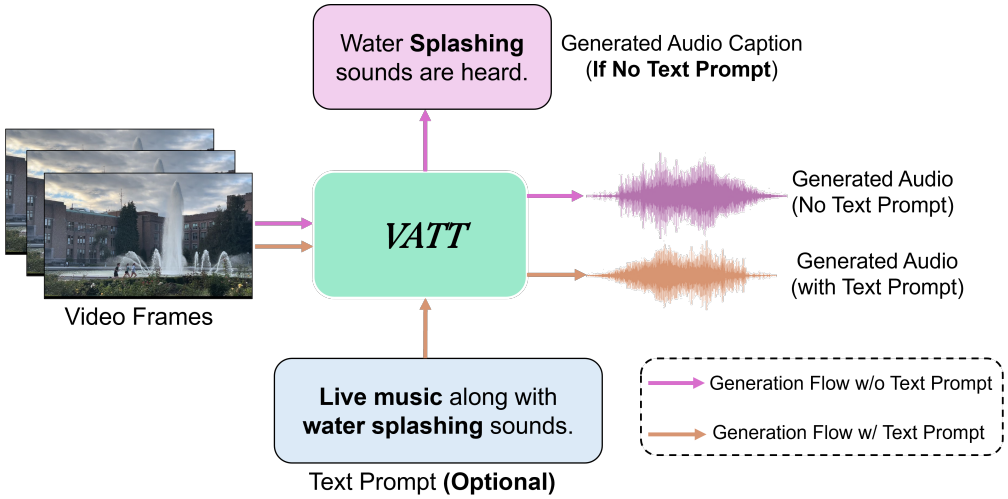

Figure 7.1: VATT is a flexible audio generative model capable of generating audio in two modes: i) When a silent video is the sole input, the model generates the audio along with a caption describing the possible audio that could match the video. ii) When in addition to the video, a text prompt is provided, the model generates audio aligned with both the video and the given text prompt. (Figure from [XL5])

the LLM in stage i), with the hidden states from the last layer extracted and attached to the audio tokens for the decoder. The decoder is trained using masked token modeling, where the objective is to predict masked audio tokens from unmasked ones at varying masking ratios. During inference, starting from all tokens being masked, an efficient parallel decoding algorithm is implemented which gradually unmasks multiple tokens in parallel based on video and text inputs until a stop condition is met. Finally, the generated tokens are converted into audio waveforms through a neural audio codec decoder. An overview of ***VATT*** is shown in Figure 7.2.

7.2 Methods

VATT consists of two modeling stages: i) **Video-to-Caption** : This stage utilizes a Large Language Model (LLM) with a learnable projection layer that converts video features into

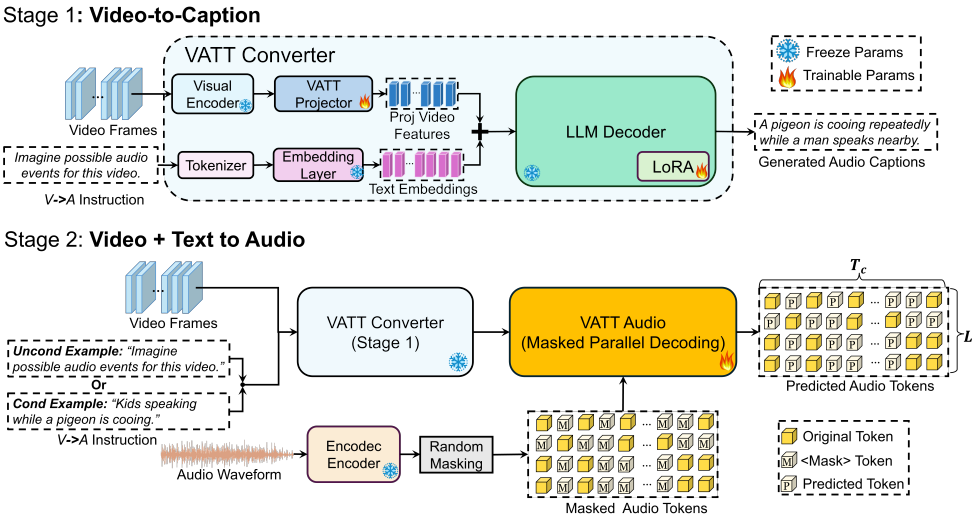

Figure 7.2: Two stages of *VATT* system training pipeline: (1) **Video-to-Caption** stage that maps video features into an audio caption through LLM. (2) **Video + Text to Audio** stage that learns to generate audio tokens through masked tokens prediction conditioned on Stage (1) features. (Figure from [XL5])

embeddings compatible with the LLM. The model receives an instruction to generate audio captions from video inputs. ii) **Video + Text to Audio**: This stage incorporates an encoder-decoder architecture. The encoder uses the finetuned LLM from Video-to-Caption stage with frozen weights. During inference, VATT generates audio tokens from video and optional text prompts through iterative parallel decoding. These tokens are then converted into audio waveforms using Encodec [Re11].

VATT Converter is designed to integrate visual and textual prompts for audio generation as well as audio captioning. The core component, *VATT Projector*, is an embedding layer that maps video features into the text embedding space of the LLM. Given visual features extracted from frame-level vision encoders $V_f = \{v_1, v_2, ..., v_T\}$, a Linear layer is applied to project each feature from its original dimension d_v to the LLM text embedding dimension d_{lm}, producing a sequence of transformed features $V_{lm} = V_f W_l + b_l$, where W_l and

b_l are learnable parameters of the linear projection.

V2A Instruction Tuning: The key functionality of VATT Converter is to extract from visual stream semantic features relevant to audio. Drawn on the success of multi-modal LLMs, such as visual-LLM [Re126] and audio-LLM [Re93], we employ multi-modal instruction tuning to align the visual inputs of videos with the ground truth audio captioning of the same videos. Given a prompt instruction, $T_i = \{t_{i1}, t_{i2}, ..., t_{iK}\}$, such as "Describe the audio that the video could generate:" and the projected visual features V_{lm} as inputs, we model conditional distribution of audio descriptions $T_a = \{t_{a1}, t_{a2}, ..., t_{aN}\}$, as $P_\theta(T_a|T_i, V_{lm})$ by fine-tuning an instruction-tuned LLM, e.g., Vicuna-7B [Re91]. For training efficiency, we fine tune the LLM with VATT Projector by integrating LoRA [Re127] adaptors while keeping the original LLM weights frozen. We minimize the negative log-likelihood of audio caption tokens conditioned on visual inputs and prompt instruction

$$\mathcal{L}_{v2t}\left(\widehat{T_a} \mid T_i, V_{lm}\right) = -\sum_{l=1}^{N} log\left[P_\theta\left(\hat{t}_{al} = t_{al} \mid T_i, V_{lm}\right)\right], \tag{7.1}$$

where t_{al} is the l-th text token in the ground truth audio description T_a, and θ is the set of trainable weights including VATT Projector and LoRA adaptor.

Video + Text to Audio Stage Once the audio-related visual features are aligned with the text features in the LLM embedding space, the LLM effectively encodes multi-modal information that serves as a representation for text generation and audio generation. Indeed, in the second stage of VATT, there are two generation modes to generate audio: i) When no conditional text prompt is provided, the video features along with a *standard template* prompt (e.g., "Describe possible audio that the video could infer.") are fed as inputs to VATT Converter. ii) When an audio caption is provided as the text prompt, the video features and the audio caption are fed together into VATT Converter. In such a case, the provided audio caption helps guide the video-to-audio generation process and overrides the need for generated audio caption.

Audio Token Decoder To generate audio, we design an audio token-based decoder, VATT Audio, conditioned on the encoded features from VATT Converter. In contrast to

existing methods, which typically use auto-regressive token modeling [Re56, Re57, Re25] or latent diffusion techniques [Re50, Re51], we adopt a novel token-based modeling technique based on masking tokens. The method, originally derived in image generation tasks [Re32] and recently adapted to text-to-audio generation [Re27, Re26], is capable of achieving competitive generation quality while improving efficiency through an iterative parallel decoding algorithm during inference.

Token-based Representation for Audio To represent audio waveforms using discrete tokens, we adopt a pretrained audio neural codec, Encodec [Re11], similarly to Foley-Gen [Re25]. Encodec is a multi-level residual vector-quantized (RVQ) autoencoder trained with waveform reconstruction and adversarial objectives, capable of high-fidelity reconstruction from compressed tokens. Specifically, Encodec uses $L = 4$ codebooks of tokens to represent the audio. Lower-level codebooks encode coarse semantic information, while higher-level codebooks capture fine-grained details. We adopt an open source Encodec model pretrained using audio waveforms at $Sr_w = 16kHz$ sampling rate. The model compresses a waveform into tokens at $Sr_t = 50Hz$ sampling rate, leading to $r_{tw} = \frac{Sr_w}{Sr_t} = 320$ waveform samples per token. For any waveform $A_{wav} \in \mathbb{R}^{1\times T_w}$, we extract corresponding audio tokens representation $A_{tok} \in \mathbb{N}^{L\times T_c}$ ($T_c = \frac{T_w}{r_{tw}}$) from Encodec encoder part. Once the model generates A_{tok}, the embedding vectors of L levels of tokens at each time step are summed up before being sent to Encodec decoder to obtain the waveform.

Masked Audio Token Generative Modeling We model the distribution of audio token matrix $A_{tok} \in \mathbb{N}^{L\times T_c}$ by developing a token masking strategy which learns the joint distribution of the audio tokens in full parallelism. This is different than using "delayed patterns" proposed in [Re57] which enables parallelism but only on the level of codebook dimension. At each time step of A_{tok}, embedding vectors of L tokens are summed up to represent audio waveform at the corresponding segment. In order to perform masking operation at any position, we introduce an additional learnable $\langle$MASK$\rangle$ token in each codebook. By randomly replacing some of the tokens entries in the A_{tok} with $\langle$MASK$\rangle$ at corresponding codebook we obtain the masked audio token matrix $A^M_{tok} \in \mathbb{N}^{L\times T_c}$. We obtain $E^M_a \in \mathbb{R}^{d_{em}\times T_c}$ by

summation of the embedding vectors of each token in A^M_{tok} along the level axis.

Conditional generative modeling is implemented as follows. We extract the hidden states of the last layer $H_{lm} \in \mathbb{R}^{d_{lm} \times T_{lm}}$ (before the LLM prediction head) from VATT Converter as the conditional inputs into the audio token decoder. We use a linear layer to project H_{lm} to $E_{lm} \in \mathbb{R}^{d_{em} \times T_{lm}}$ with same feature dimension as the masked audio embeddings E^M_a. A straightforward way to model the relationship between E^M_a and E_{lm} is to use an interleaving self-attention and cross-attention block as proposed in Vanilla Transformer architecture [Re23]. However, we find that such interleaved interaction between audio and multi-modal input condition does not capture the fine-grained correspondence between them. Therefore, we propose to use a bi-directional self-attention architecture to fuse the features.

Specifically, we concatenate E_{lm} with E^M_a along the temporal axis to obtain the fused features $E_{mm} = Concat(\left[E_{lm}, E^M_a\right])$. The decoder consists of L_{mm} layers of self-attention blocks. The output hidden states in the last layer of the decoder, $H^{out}_{mm} = Dec\,(E_{mm})$, represent fused audio and conditions features. We only extract the part of the hidden states corresponding to the audio tokens, $H^{out}_a \in \mathbb{R}^{d_{mm} \times T_c}$, and pass it through L Linear layers in parallel to perform classification on masked tokens at each level of the codebooks. For each masked audio token in matrix A^M_{tok}, we calculate the cross-entropy loss between the predicted token $\hat{a}_{tok}$ and the ground truth token a^{gt}_{tok}, formulated as

$$\mathcal{L}_{VATT} = - \sum_{a_{tok} \in A^M_{tok}} \mathbb{I}\left(a_{tok} = <\text{MASK}>\right) log\left[P_\phi(\hat{\mathbf{a}}_{\mathbf{tok}} = a^{gt}_{tok} | A^M_{tok}; H_{lm}))\right], \tag{7.2}$$

where ϕ is the set of trainable parameters in the audio token decoder, and $\mathbb{I}$ is the indicator function.

7.2.1 Masking Design and Iterative Parallel Decoding

Masking Distribution Design Inspired by [Re32, Re27], we incorporate variable random masking. In particular, it was shown that masking ratio plays an important role in audio token decoder to generate meaningful signals. While in [Re32, Re27] arc-cosine masking

distributions is used by default, here we study several masking strategies that include distributions along with different hyper-parameters to find the strategy that reaches more optimal generation quality. Our study shows that normal distribution with a mean of 0.75 and standard deviation of 0.25, truncated from 0.5 to 1.0 is such optimal strategy. The general interpretation of this strategy is that a relatively high range masking ratio enables models to generate better initial tokens when most of the entries in the token matrix are masked. This is essential for future decoding steps to generate meaningful tokens.

Iterative Parallel Decoding Scheduling of masking plays a key role as well. During inference, we follow the cosine scheduling scheme proposed in [Re32] to gradually resolve the audio tokens. The iterative sampling procedure starts with all $\langle$MASK$\rangle$ in the audio token matrix. At a step t, the model takes the audio token matrix A_{t-1} from the previous step along with the conditions as inputs and samples a new audio token matrix $\hat{A}_t$ in parallel with all tokens unmasked. Based on the confidence at each entry of $\hat{A}_t$ only tokens with top-k confidence are kept while the remaining entries are re-filled with $\langle$MASK$\rangle$, resulting in A_t. The cosine scheduling scheme determines the ratio of re-masked tokens by $r_t = cos\left(\frac{\pi}{2} \cdot \frac{t}{T}\right)$. Notably, to resolve the confidence of each entry in the matrix, we adopt the *"gumbel-top-k trick"* [Re128] with temperature that varies, i.e., $c_i = \frac{log(p_i)}{\tau} + G$, where $G \sim \text{Gumbel}(0, 1)$ and p_i denotes the output probability of the sampled token at the entry i. This is equivalent to sampling k values from multinomial distribution from the softmax probabilities without replacement. The temperature τ controls the degree of stochasticity. We use $\tau = \tau_0 \cdot (1 - \frac{t}{T})$ with linear decay during generation, where τ_0 is the initial temperature. Similarly to [Re32, Re27], our method achieves optimal quality and fast speed within a few decoding steps (typically 10 - 20).

7.3 Experiments and Results

Evaluation Metrics: To evaluate video-to-audio generation quality, we follow the method of [Re25], which proposed the metrics Kullback-Leibler-Divergence (KLD) with PassT [Re129], Fréchet Audio Distance (FAD) [Re130] and Align Accuracy (Align Acc) [Re46]. KLD mea-

sures how closely the generated audio matches the GT through pairwise comparison, reflecting how well the audio captures the concepts in the video. FAD evaluates the overall distribution, indicating the overall quality of the audio. Align Acc assesses the relevance and temporal alignment of the audio and the video. Additionally, we incorporate generation speed (time taken per waveform sample) to measure efficiency. We also compute the CLAP score [Re21] to evaluate the adherence of generated audio to text prompts to compare our results with text-to-audio generation.

For video-to-audio captioning, we use two types of metrics, natural language generation (NLG) metrics and audio-text relevance metric. NLG metrics evaluate the generated captions with respect to the ground truth audio captions using rule-based matching in terms of precision and recall. These metrics include BertScore [Re6], BLEU-4 [Re131], ROUGE-L [Re132] and CIDEr [Re133]. To assess the relevance of generated audio captions with the actual audio, we compute the CLAP-score [Re21] as cosine similarity between audio and text embeddings.

Quantitative Evaluation of Audio Generation: We evaluate audio generation of VATT models on the VGGSound test split. For each of the 15,446 video samples, we generate a 10-second audio waveform. We compare VATT variants against existing video-to-audio generation methods as well as text-to-audio generation methods including AudioLDM-2 [Re51] and AudioGen [Re56] using different text prompts. The results on the metrics described above are summarized in Table 7.1 and Table 7.2. **VATT models** achieve best KLD score and Align Acc against other methods while maintaining competitive FAD (top 2). Notably, when guided by GT audio captions (VATT-LLama-T and VATT-Gemma-T; bottom) our models generate sounds that match the GT audio more accurately, as indicated by lowest KLD score of 1.41 and 1.66 for VATT models with two LLM backbones, surpassing both video-to-audio and text-to-audio methods. In comparison to text-to-audio methods, VATT models achieve competitive audio-text alignment in terms of CLAP score, demonstrating a strong capability to follow text prompts.

Quantitative Evaluation of Video-to-Audio Captioning: We evaluate video-to-

Table 7.1: Quantitative results against video-to-audio generation methods on VGGSound test set. '-T' refers to model with text prompts. (Table from [XL5])

Methods	KLD ↓	FAD ↓	Align Acc ↑	Speed (s) ↓
SpecVQGAN [Re43]	3.78	6.63	48.79	7.2
IM2WAV [Re44]	2.54	6.32	74.31	289.5
Diff-Foley [Re46]	3.15	6.40	82.47	4.4
FoleyGen [Re25]	2.89	2.59	73.83	6.9
V2A-Mapper [Re47]	2.78	**0.99**	74.37	11.54
VATT-LLama (Ours)	2.39	2.38	80.32	**1.1**
VATT-Gemma (Ours)	**2.25**	2.35	**82.81**	**0.65**
VATT-LLama-T (Ours)	**1.41**	2.54	80.16	**1.2**
VATT-Gemma-T (Ours)	**1.66**	2.98	81.48	**0.76**

audio captioning by prompting VATT Converter to generate audio captions. We use the prompt "Describe the possible audio for this video:" to generate captions for all VGGSound test videos. For baselines, we prompt LLAVA-13B-v1.5 model in two zero-shot modes to generate visual and audio descriptions respectively. Since LLAVA can take a single image as an input only, we select the middle frame of videos. We use "Provide a concise, descriptive caption for the following image." as the visual prompt, and "Describe the sounds that this scene could yield in a short sentence without reasoning" as the audio prompt. We also compare against a video LLM baseline, Video-LLAMA-7B, to perform zero-shot video-to-audio captioning. Specifically, we directly input VGGSound videos into the VL branch of the Video-LLAMA model, and prompt it to generate audio captions using the instruction "User/ What sounds could match the video?" Since Video-LLAMA has not been pretrained on VGGSound dataset and LTU generated captions, we implement a similar structure of Video-LLAMA and train on our LTU-generated captioning data. We replaced the original BLIP-2 visual features used by Video-LLAMA with our eva02-CLIP-L visual features due to the expensive pre-processing time for all BLIP-2 features from videos in VGGSound

Table 7.2: Quantitative results comparing VATT with text-to-audio generation methods on VGGSound test set. '-T' refers to model with text prompts. CLAP score is calculated as the cosine similarity of generated audio with respect to the GT audio caption. (Table from [XL5])

Methods	Text Prompt	KLD ↓	FAD ↓	Align Acc ↑	CLAP Score ↑
AudioGen [Re56]	LLAVA visual caption	3.65	6.03	41.66	-
AudioGen [Re56]	GT audio caption	2.19	3.17	48.96	**0.409**
AudioLDM-2 [Re51]	LLAVA visual caption	3.54	3.62	53.49	-
AudioLDM-2 [Re51]	GT audio caption	2.09	**2.46**	51.84	0.326
VATT-LLama-T (Ours)	GT audio caption	**1.41**	2.54	**80.16**	0.347
VATT-Gemma-T (Ours)	GT audio caption	**1.66**	2.98	**81.48**	0.310

and AudioSet. For the Video-QFormer component of Video-LLAMA, we keep it the same as Video-LLAMA, and we name this model as VATT-Qformer - LLama. Our evaluation is summarized in Table 7.3. VATT models with LLMs outperform LLAVA-prompted and Video-LLAMA zero-shot results demonstrating a stronger capability to infer sounds from videos semantically. In particular, when measuring audio-text relevance, our model with LLama achieves an increase of **+5.0%** in accuracy when compared with LLAVA visual caption baselines. For reference, the ground truth audio captions generated by LTU [Re93] have an average CLAP score of 0.379.

Table 7.3: Comparison of video-to-audio captions on NLG evaluation metrics and text-audio relevance (CLAP Score). (Table from [XL5])

Methods	BertScore (F1) ↑	BLEU-4 ↑	ROUGE-L ↑	CIDEr ↑	CLAP Score ↑
LLAVA w/ Visual Prompt	0.855	0.089	0.137	0.026	0.213
LLAVA w/ Audio Prompt	0.870	0.123	0.155	0.095	0.182
Video-LLAMA w/ Audio Prompt	0.861	0.091	0.117	0.021	0.204
VATT Converter - Gemma (ours)	0.900	0.345	0.337	0.926	0.229
VATT-Qformer - LLama	0.907	0.419	0.375	1.264	0.245
VATT Converter - LLama (ours)	**0.909**	**0.424**	**0.384**	**1.354**	**0.263**

Chapter 8

SPATIAL AUDIO EFFECTS IN ANY ROOM ENVIRONMENTS

This chapter is based on the following published work: [XL6].

8.1 Motivation

In mixed reality applications, a realistic acoustic experience in spatial environments is as crucial as the visual experience for achieving true immersion. Despite recent advances in neural approaches for Room Impulse Response (RIR) estimation, most existing methods are limited to the single environment on which they are trained, lacking the ability to generalize to new rooms with different geometries and surface materials. We aim to develop a unified model capable of reconstructing the spatial acoustic experience of any environment with minimum additional measurements. To this end, we present xRIR, a framework for cross-room RIR prediction. The core of our generalizable approach lies in combining a geometric feature extractor, which captures spatial context from panorama depth images, with a RIR encoder that extracts detailed acoustic features from only a few reference RIR samples. More specifically, xRIR features three key components: i) a Geometric Feature Extractor, which utilizes a vision transformer to process panorama depth images from the receiver's perspective, capturing the spatial relationships between the source and receiver positions within the room; ii) a Reference RIR Encoder, which extracts spatio-temporal features from a few reference RIRs, capturing the unique energy decay and reverberation characteristics associated with room materials; and iii) a Fusion and Weighting Module, which predicts the target RIR through a weighted combination of the reference RIRs. By integrating complementary geometric and acoustic features, xRIR effectively approximates both structural and

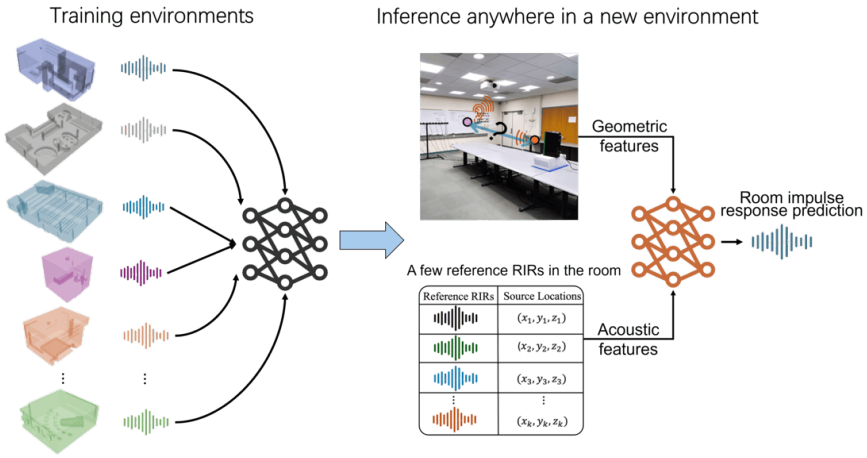

Figure 8.1: Our xRIR framework can predict accurate room impulse responses (RIR) of any new environment, by integrating the geometric prior learned from a large simulated RIR dataset of diverse training environments and the nuanced acoustic profile extracted from a few reference RIR measurements in the new environment. (Figure from [XL6])

material properties of any room, enabling precise RIR predictions not only at new locations in the training environments, but also in any new environment of interest. To evaluate our method, we introduce ACOUSTICROOMS, a new dataset featuring high-fidelity simulation of over 300,000 RIRs from 260 rooms, specifically curated for this task.

8.2 Methods

We tackle the *cross-room* room impulse response (RIR) prediction task, which aims to predict single-channel (omnidirectional) RIRs for any source-receiver pair across diverse room environments, including those *unseen* during training. We aim to develop a generalizable model that can accurately predict RIRs in any environment without labor-intensive data collections or training a separate model for each room. xRIR achieves this by utilizing only minimal additional measurements from the new room, such as only a few panorama depth images and reference RIR measurements, to quickly adapt to new acoustic environments

with minimal effort, thereby facilitating generalization to previously unseen environments. Next, we formally define the cross-room RIR prediction task by outlining its data, inputs, and the modeling objective.

Data. Let $R = \{R_1, R_2, \ldots, R_M\}$ represent a dataset of M rooms, split into a training set, $R_{\text{train}} \subset R$, and a test set, $R_{\text{test}} = R \setminus R_{\text{train}}$. Each room R_m includes a set of receiver locations, denoted $L_m = \{P_r^{(m,1)}, P_r^{(m,2)}, \ldots, P_r^{(m,N_m)}\}$, where N_m is the number of receivers in the room. For each receiver $P_r^{(m,i)}$ in room R_m, RIRs are measured at various source locations $P_s^{(m,i,j)}$, resulting in measurements $A_{m,i,j}$ of the source-receiver pair $P_s^{(m,i,j)}$ and $P_r^{(m,i)}$.

Inputs. To capture the necessary observation conditions for predicting a target RIR A_t, we define an observation tuple $\mathbf{O} = (P_s, P_r, G_r)$, where P_s is the target source location, P_r is the receiver location, and G_r represents the local geometry near the receiver location P_r, e.g., room boundary points or depth maps around the receiver. Additionally, we introduce a set of K reference RIRs measured at the target receiver location P_r from various reference source locations $\mathbf{P}_{\text{ref,s}} = \{P_{\text{ref,s}}^{(1)}, P_{\text{ref,s}}^{(2)}, \ldots, P_{\text{ref,s}}^{(K)}\}$. These references, denoted as $\mathbf{A}_{\text{ref}} = \{A_{\text{ref}}^{(1)}, A_{\text{ref}}^{(2)}, \ldots, A_{\text{ref}}^{(K)}\}$, are crucial for capturing essential acoustic characteristics that encode nuanced information about room materials. Note that in the above formulation, while we fix the receiver location and set reference RIRs at different source locations, exchanging the receiver and source in the input yields *an equivalent alternative formulation.*

Modeling Objective. The objective of cross-room RIR prediction is to train a model F that predicts the target RIR $\hat{A}_t$ using the observation tuple $\mathbf{O}$ along with the reference RIRs $\mathbf{A}_{\text{ref}}$ and their respective source locations $\mathbf{P}_{\text{ref,s}}$: $\hat{A}_t = F(\mathbf{O}, \mathbf{A}_{\text{ref}}, \mathbf{P}_{\text{ref,s}})$. In this formulation, $\mathbf{O}$ provides the geometric and positional context, while $\mathbf{A}_{\text{ref}}$ and $\mathbf{P}_{\text{ref,s}}$ give sparse acoustic observations that help bridge the lack of explicit material properties by capturing key room acoustics characteristics.

Unlike the *single-room* RIR prediction task [Re65, Re64], which assumes consistent geometry and material properties and fits a separate model for each scene, our cross-room formulation aims to train a single model that generalizes across multiple scenes with diverse

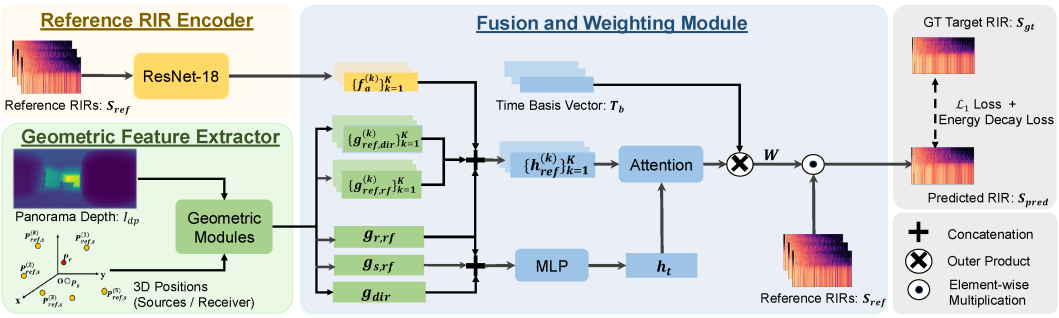

Figure 8.2: **System Overview of Our xRIR Model for Cross-Room RIR Prediction.** The model architecture consists of three main components: i) a *Geometric Feature Extractor*, which captures spatial relationships among the source, receiver, and room geometry; ii) a *Reference RIR Encoder*, which extracts spatiotemporal features from reference RIRs; and iii) a *Fusion and Weighting Module*, which integrates these spatial and acoustic features to predict the target RIR. (Figure from [XL6])

room geometries and materials, while with the extra condition of only a few RIR measurements. Models that are designed for single-room RIR prediction task [Re64, Re65] must be re-trained with dense data when applied to a new environment. While our method uses one unified model to predict accurate RIRs across different rooms, seen or unseen. Our formulation can also be easily adapted to the single-room RIR prediction setting by fitting dense measurements in the room as in prior work. Please see supp. for results on single-room experiments.

The xRIR Model: To solve the cross-room RIR prediction task, we propose a new architecture, *xRIR*, which processes not only geometry and positional features of source and receiver, but also leverages the reference RIRs to accurately predict the target RIR. As illustrated in Figure 8.2, xRIR consists of three main components: i) a *Geometric Feature Extractor* (Sec. 8.2.1), which encodes the spatial relationships among the source, receiver, and room surface geometry, capturing important geometric features that shape acoustic behavior; ii) a *Reference RIR Encoder* (Sec. 8.2.2), which processes the spatiotemporal char-

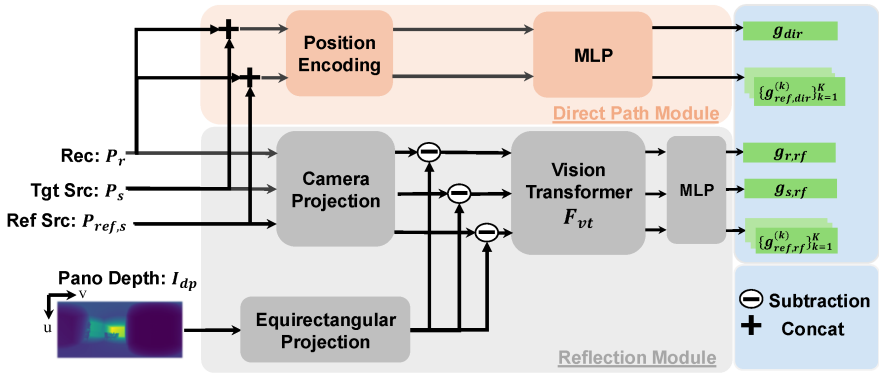

Figure 8.3: **Illustration of the Geometric Feature Extractor.** Rec: Receiver, Tgt Src: Target Source, Ref Src: Reference Source. (Figure from [XL6])

acteristics of the reference RIRs to extract features that represent their acoustic properties within the room.; and iii) a *Fusion and Weighting Module* (Sec. 8.2.3), which integrates the spatial features from the Geometric Feature Extractor with the reference RIR features from the Reference RIR Encoder, generating a set of weights to combine reference RIRs as the predicted target RIR.

8.2.1 Geometric Feature Extractor

The Geometric Feature Extractor module captures spatial relationships among the source, receiver, and room geometries, which is important for accurate acoustic modeling. It consists of two geometric sub-modules: the *Direct Path Module*, and the *Reflection Module*. These two modules emulate the process of sound propagation. The Direct Path Module extracts the feature of the direct path between source and receiver, while the Reflection Module models the sound propagation path through the reflections from the room boundaries. A detailed overview of the Geometric Feature Extractor is illustrated in Figure 8.3.

Direct Path Module. To capture the direct path between each source and the receiver, we concatenate their 3D coordinates. For the target source, we define $P_{\text{dir}} = (P_s, P_r)$, where P_s and P_r are the coordinates of the source and the receiver, respectively. For each reference source $P^{(k)}_{\text{ref,dir}}$, we define $P^{(k)}_{\text{ref,dir}} = (P^{(k)}_{\text{ref},s}, P_r)$. P_{dir} and each $P^{(k)}_{\text{ref,dir}}$ encode locations of every

source-receiver pair, thereby encoding the direct path information. To extract their features, we apply sinusoidal positional encoding [Re134] followed by a multi-layer perceptron (MLP) to project them into high-dimensional vectors, resulting in $g_{\text{dir}} \in \mathbb{R}^{1\times C_d}$ for the target source-receiver pair and $g^{(k)}_{\text{ref,dir}} \in \mathbb{R}^{1\times C_d}$ for each reference source-receiver pair.

Reflection Module. Inspired by INRAS [Re64], we also model the reflection paths between source and receiver via the room boundary. But differently, instead of using a fixed set of bounce points per room, we propose to use a panorama depth map at the receiver's location as a proxy for local room geometry to unify the representation across different rooms.

Given the panorama depth map $I_{dp} \in \mathbb{R}^{H\times W\times 3}$ centered at receiver's viewpoint, we first project I_{dp} to a 3D coordinate map $I_{\text{coord}} = F_{\text{rect}}(I_{dp})$ via an equi-rectangular projection transformation F_{rect}. Each pixel in I_{coord} represents the 3D coordinate of a visible boundary point in the room from the receiver's view. Each source in the room has chance to reflect through these points until finally reaching the receiver. To model such interactions, we create a set of reflections-based maps by subtracting the 3D coordinate map from the source and receiver positions.

To perform the subtraction, it is necessary to unify the coordinates between the 3D coordinate map and the source / receiver positions. We achieve this by projecting the world coordinates of the sources and the receiver into camera coordinates at the receiver's position, resulting in the same coordinate system as the 3D coordinate map. We obtain the target source position as $P_{\text{rel},s} = R(P_s - P_r)$ and each reference source location as $P^{(k)}_{\text{rel, ref}} = R(P^{(k)}_{\text{ref},s} - P_r)$, where R is the world-to-camera transformation matrix. Then we create reflection-based maps $I_{\text{rf},s}$ for the target source, $I^{(k)}_{\text{rf, ref}}$ for reference sources, as well as $I_{r,\text{rf}}$ for the receiver by performing subtractions: $I_{s,\text{rf}} = P_{\text{rel},s} - I_{\text{coord}}$, $I^{(k)}_{\text{ref,rf}} = P^{(k)}_{\text{rel, ref}} - I_{\text{coord}}$ and $I_{r,\text{rf}} = I_{\text{coord}} - \mathbf{0}$, where $\mathbf{0}$ means the origin, where the receiver is located.

These reflection-based maps encode the dense interaction between room geometry and the sources / receiver. To further extract features, we utilize a vision transformer module [Re135] F_{vt} that partitions each reflection-based map into patches, aggregates local features, and builds spatial dependencies among patches. This results in compact patch-level geometry

representations: $g'_{\text{r,rf}} = F_{\text{vt}}(I_{r,\text{rf}})$, $g'_{\text{s,rf}} = F_{\text{vt}}(I_{s,\text{rf}})$ and $\{g'^{(k)}_{\text{ref,rf}} = F_{\text{vt}}(I^{(k)}_{\text{ref,rf}})\}_{k=1}^{K}$, where each feature map has dimensionality $N_p \times C_p$. Finally, we apply a MLP layer to project the patch dimension N_p to 1, resulting in $g_{\text{r,rf}}$, $g_{\text{s,rf}}$, and $\{g^{(k)}_{\text{ref,rf}}\}_{k=1}^{K}$, respectively.

8.2.2 Reference RIR Encoder

To capture acoustic features related to energy decay and reverberation patterns within the room, we leverage reference RIRs as proxies for the acoustic characteristics at various source locations relative to the receiver. To encode these acoustic features, we first compute the log-magnitude spectrogram of each reference RIR using the Short-Time Fourier Transform (STFT): $\mathbf{S}_{\text{ref,k}} = \log(\|\text{STFT}(A_{\text{ref,k}})\|)$, where $\mathbf{S}_{\text{ref,k}} \in \mathbb{R}^{F \times T}$. To extract robust acoustic features, we implement the Reference RIR Encoder using ResNet-18, and use the mean pooled features $f_a^{(k)} \in \mathbb{R}^d$ from the last layer to encode each reference RIR.

8.2.3 Fusion and Weighting Module

The Fusion and Weighting Module integrates the outputs from the Geometric Feature Extractor and the Reference RIR Encoder to generate the target RIR prediction. This module combines geometric and acoustic features for reference sources as well as the geometric features of target source, finally computing the weights that are applied to reference RIRs.

Fusion of Geometric and Acoustics Features. For each reference source, we combine the geometric feature $g^{(k)}_{\text{ref,dir}}$, $g^{(k)}_{\text{ref,rf}}$, $g_{\text{r,rf}}$ and the acoustic feature $f_a^{(k)}$ by concatenating them along the feature dimension, resulting in: $\mathbf{h^{(k)}_{ref}} = \text{Concat}(g^{(k)}_{\text{ref,dir}}, g^{(k)}_{\text{ref,rf}}, g_{\text{r,rf}}, f_a^{(k)})$.

Similarly, for the target source, we combine the geometric feature g_{dir}, $g_{\text{s,rf}}$ and $g_{\text{r,rf}}$ via concatenation, yielding: $\mathbf{h'_t} = \text{Concat}(g_{\text{dir}}, g_{\text{s,rf}}, g_{\text{r,rf}})$. We then project the fused feature $\mathbf{h'_t}$ to $\mathbf{h_t}$ through a MLP to make the feature dimension the same as $\mathbf{h^{(k)}_{ref}}$.

To align the target and reference features, we compute the attention between the target fused vector $\mathbf{h}_t$ and each reference fused vector $\mathbf{h^{(k)}_{ref}}$. Specifically, given the reference fused features $\mathbf{H_{ref}} = \{\mathbf{h^{(k)}_{ref}}\}_{k=1}^{K}$ and the target fused vector $\mathbf{h}_t$, the attention output $\mathbf{Z}$ is computed as:

$$\mathbf{Z} = \text{softmax}\left(\frac{\mathbf{H_{ref}} \cdot \mathbf{h_t^T}}{\sqrt{C}}\right) \odot \mathbf{H_{ref}},$$

where $\cdot$ and $\odot$ denote matrix multiplication and element-wise multiplication with broadcasting respectively, and $\mathbf{H_{ref}} \in \mathbb{R}^{K\times C}$, $\mathbf{h_t} \in \mathbb{R}^{1\times C}$, $\mathbf{Z} \in \mathbb{R}^{K\times C}$. These attention outputs $\mathbf{Z} = \{\mathbf{z}_k\}_{k=1}^{K}$ for each reference RIR is now attended by the fused feature of the target RIR.

Time-Aligned Weighting Matrix. Given the attention outputs $\mathbf{Z} \in \mathbb{R}^{K\times C}$, we next generate a time basis vector $\mathbf{T}_b$ based on the temporal indices of the spectrogram $[0, 1, 2, \ldots, T]$. Specifically, we compute $\mathbf{T_b'}$ using sinusoidal positional encoding [Re134] and then apply a MLP layer to project $\mathbf{T_b'}$ to C, resulting in $\mathbf{T_b} \in \mathbb{R}^{T\times C}$. We generate the time-aligned weighting matrix $\mathbf{W} \in \mathbb{R}^{K\times T}$ by computing the outer product between $\mathbf{Z}$ and $\mathbf{T_b}$: $\mathbf{W} = \mathbf{Z} \cdot \mathbf{T_b^T}$. Each row of $\mathbf{W}$ corresponds to the weights applied to the log-magnitude spectrogram of each reference RIR, adapting them to match the temporal structure of the target spectrogram. This weighting matrix $\mathbf{W}$ effectively shapes each reference spectrogram to align with the characteristics of the target RIR.

Finally, we predict the target RIR's log-magnitude spectrogram $\mathbf{S}_{\text{pred}}$ via the weighted sum of the log-magnitude spectrograms of the reference RIRs: $\mathbf{S}_{\text{pred}} = \sum_{k=1}^{K} \mathbf{W}_k \odot \mathbf{S}_{\text{ref},k}$. $\mathbf{W}_k$ is the k-th row of the weight matrix $\mathbf{W}$, applied to the corresponding log-magnitude spectrogram $\mathbf{S}_{\text{ref},k}$.

Training and Inference During training, we use the magnitude STFT L_1 Loss to compute the error between the magnitude spectrograms of the predicted target RIR and the ground-truth RIR: $\mathcal{L}_{\text{STFT}} = \| \exp(\mathbf{S}_{\text{pred}}) - \exp(\mathbf{S}_{\text{gt}}) \|_1$. Additionally, following [Re4], we incorporate an energy decay loss to optimize the decay patterns of the predicted spectrogram. The energy decay loss $\mathcal{L}_{\text{ED}}$ is defined as: $\mathcal{L}_{\text{ED}} = \left\| \text{EDC}(\mathbf{S}_{\text{pred}}) - \text{EDC}(\mathbf{S}_{\text{gt}}) \right\|_1$, where $\text{EDC}(\cdot)$ denotes the energy decay curve of RIR in the frequency domain. The total loss becomes $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{STFT}} + \lambda \mathcal{L}_{\text{ED}}$, where λ is a weight to balance the contribution of the energy decay loss.

During inference, we randomly samples K RIRs $\{A_{ref,k}\}_{k=1}^{k=K}$ along with corresponding source locations $\{P_{\text{ref},k}\}_{k=1}^{k=K}$ from a test room as reference inputs. The model predicts the

magnitude spectrogram of a target RIR, which is then converted back to a waveform via the Griffin-Lim [Re136] algorithm.

8.3 Experiments and Results

Method	Seen Splits			Unseen Splits		
	EDT error (s)	**C50 error (dB)**	**T60 error (%)**	**EDT error (s)**	**C50 error (dB)**	**T60 error (%)**
Random Across Rooms	0.290	6.831	37.35	0.313	7.802	35.15
Random Same Room	0.129	3.567	12.80	0.172	5.440	16.08
Few-shot RIR [Re4] (K=1)	0.157	3.957	31.42	0.130	3.225	20.10
Few-shot RIR [Re4] (K=4)	0.157	4.026	31.63	0.136	3.568	19.30
Few-shot RIR [Re4] (K=8)	0.174	4.451	32.71	0.187	4.470	21.15
Linear Interpolation (K=8)	0.094	2.421	9.76	0.121	3.090	13.73
Nearest Neighbor (K=8)	0.064	1.717	8.94	0.090	2.667	11.64
xRIR (K=1)	0.046	1.183	9.50	0.075	1.841	13.47
xRIR (K=4)	0.040	1.005	8.15	0.068	**1.335**	13.28
xRIR (K=8)	**0.038**	**0.940**	**8.13**	**0.055**	1.457	**10.53**

Table 8.1: **Cross-Room RIR Prediction Results for Both the Seen and Unseen Splits.** We report EDT Error (EDT) in seconds, C50 Error (C50) in dB, and T60 percentage error (T60), with lower values indicating better performance. For Few-shot RIR [Re4] and xRIR (Ours), we evaluate in a few-shot manner by setting the number of reference RIRs K to 1, 4, and 8. (Table from [XL6])

8.3.1 Implementation Details

In the ACOUSTICROOMS dataset, RIRs are sampled at 22,050 Hz with a maximum length of 9600 samples (0.435 s). We compute the magnitude spectrogram S with FFT size 124, window size 62, and hop size 31, yielding a shape of 63×310. Panorama depth maps of room geometry are rendered at a resolution of 256×512 from the receiver's location, and source/receiver positions are recorded as 3D coordinates (x, y, z). For xRIR, we implement a Vision Transformer block F_{vt} with 6 multi-head attention layers (8 heads, hidden size 512).

Method	Classroom			Dampened Room			Hallway			Complex Room		
	EDT	**C50**	**T60**	**EDT**	**C50**	**T60**	**EDT**	**C50**	**T60**	**EDT**	**C50**	**T60**
Random Across Rooms	0.546	8.740	19.03	0.771	18.726	-	0.874	11.025	21.71	0.472	7.392	16.01
Random Same Room	0.160	3.092	3.12	0.099	6.840	-	0.308	6.461	16.61	0.218	4.566	5.66
Linear Interpolation (K=8)	0.113	2.172	4.42	0.058	4.584	-	0.088	2.127	4.55	0.124	2.848	5.17
Nearest Neighbor (K=8)	0.108	1.949	**2.71**	**0.044**	**3.278**	-	0.068	0.990	**3.02**	0.091	1.936	**2.53**
Diff-RIR [Re5] (K=12)	0.113	2.147	12.39	0.100	3.796	-	0.160	2.049	14.34	0.115	2.027	12.76
xRIR (K=8) (Ours)	**0.093**	**1.628**	6.25	0.044	3.302	-	**0.062**	**0.954**	3.20	**0.077**	**1.688**	4.33

Table 8.2: **Sim-to-Real Transfer Results in Four Real Environments from the Hearing-Anything-Anywhere Dataset [Re5].** We report EDT Error (EDT) in seconds, C50 Error (C50) in dB, and T60 percentage error (T60). Due to noisy measurements in the dampened room, resulting in low SNR and invalid T60 calculations on the EDC curve, we omit this metric for the dampened room. (Table from [XL6])

Depth maps are divided into 16×32 patches, resulting in all reflection-based features such as $g_{\text{r,rf}}$ and $g_{\text{s,rf}}$ of dimension 256×512. Direct path features are calculated using sinusoidal encoding on each 3D coordinate with 20 frequency bins, and are then projected into 256-dimensional vectors via MLP. For loss calculation, we set $\lambda = 0.01$ to balance the STFT loss and the energy decay loss.

8.3.2 Baselines

We compare with a series of baselines as well as prior methods [Re4, Re5]:

- **Random Across Rooms**: Randomly sample a RIR from the entire dataset as the prediction for the target RIR.

- **Random Same Room**: Randomly sample a RIR from the same room as as the prediction for the target RIR.

- **Nearest Neighbor**: Sample k-shot reference RIRs and select the RIR with the closest spatial distance to the target source as the prediction.

- **Linear Interpolation**: Linearly interpolate between k-shot reference RIRs based on the distance between each reference and the target source location.

- **Few-Shot RIR [Re4]**: Few-Shot RIR implements a transformer architecture that fuses features from separate encoders of multi-modal conditional inputs and then generates the target RIR by decoding the transformer outputs via a UNet decoder. We adapt their model to our task by replacing the binaural echos with our single-channel reference RIRs (different source and receiver locations) and using panorama depth images as inputs to the image encoder instead of egocentric RGBD images.

- **Diff-RIR [Re5]**: We compare with Diff-RIR in evaluation on sim-to-real transfer. The framework utilizes the few-shot, i.e., 12 reference RIRs, to train a differentiable rendering pipeline to learn acoustics parameters of the room geometry. For fair comparison, we finetune our xRIR model pre-trained on ACOUSTICROOMS on the same set of reference RIRs as Diff-RIR in each room, and then test on the same test split. Note that Diff-RIR requires training one model per each room and the training process becomes computationally infeasible for large space with complex room geometries. Therefore, we do not include it in our comparison on ACOUSTICROOMS.

8.3.3 Metrics

We evaluate the energy pattern of the generated RIRs against ground-truth RIRs using three key acoustic metrics, which are strongly correlated with hearing perception and commonly used in prior work on RIR prediction [Re64, Re65]:

- **Early Decay Time (EDT) Error:** To evaluate early reflection characteristics, we use the EDT error, which measures the time taken for the initial 5 dB decay in the energy curve.

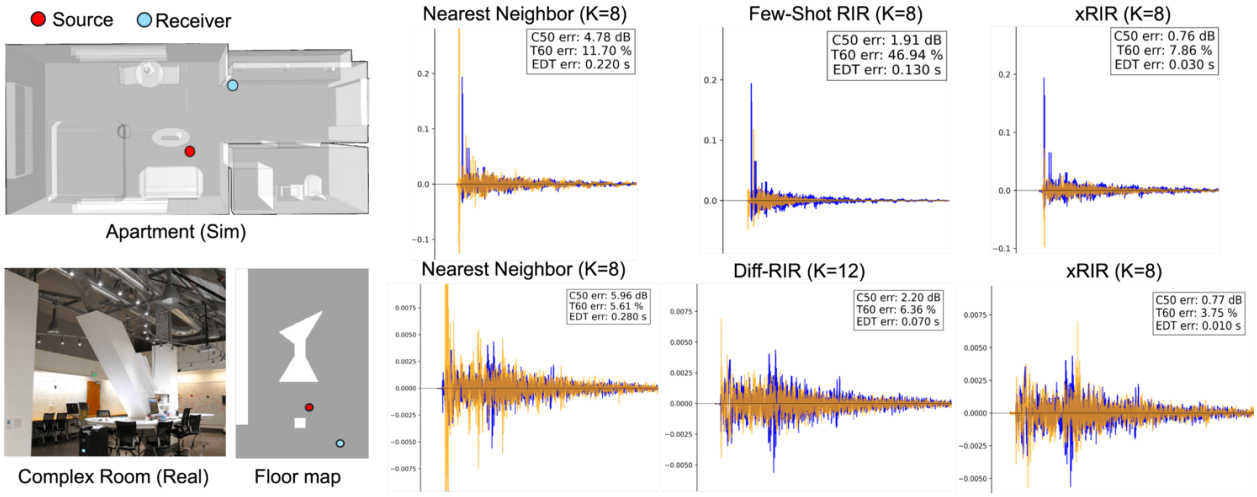

Figure 8.4: **Qualitative Comparisons of RIR Predictions.** We compare the performance of our method and the baselines both in simulated (top row) and real (bottom row) environments. Room geometry, sample RIR predictions, and the corresponding error metrics are included. xRIR shows more accurate RIR predictions in both settings. (Figure from [XL6])

- **Clarity (C50):** For comparing early-to-late energy ratios, we employ the clarity metric $C50$, which provides insights into the prominence of early reflections over later reverberations.

- **T60 Error:** We evaluate the accuracy of reverberation time by comparing the $T60$ value of the predicted RIR and the ground-truth RIR. We calculate $T60$ using $T20$, based on a linear fit between $-5\,\text{dB}$ and $-25\,\text{dB}$ on the logarithmic energy decay curve obtained from Schroeder Backward Integration [Re137].

8.3.4 Quantatitive Results on ***ACOUSTICROOMS***

We show cross-room RIR prediction results in both environments seen during training as well as unseen new environments. As shown in Table 8.1, our xRIR model significantly

Figure 8.5: **Qualitative Comparisons of Acoustic Map Predictions in Two Real Environments: a Hallway and a Classroom.** We visualize the acoustics maps by computing the C50 metric at dense locations in the entire room and compare with the ground-truth acoustic map. xRIR achieves C50 distributions that better matches the ground-truth. (Figure from [XL6])

outperforms all baselines across all metrics (EDT error, $C50$ error, and $T60$ error). In the seen split, our xRIR model yields the lowest errors across metrics, particularly in $C50$ and $T60$. Our gains persists in the unseen split. In particular, our model with $K = 8$ reduces $T60$ error to around 10%, while other baselines exhibit much higher errors. This result highlights our model's robustness in capturing reverberation characteristics across different room configurations and the ability to generalize to unseen environments with different room acoustic properties.

The Few-Shot RIR approach from [Re4] does not perform well on ACOUSTICROOMS. We suspect that this is due to two factors: i) their UNet decoder struggles to reconstruct high-fidelity RIRs on our data, as it relies on highly compressed fusion features; ii) their method uses binaural echoes with co-located source and receiver positions, which fundamentally differ from our setup, where reference RIRs are measured with the source and receiver at different locations. This spatial disparity likely impacts feature relevance, limiting its performance on our dataset.

8.3.5 Sim-to-Real Transfer to Real Environments

To evaluate whether our model can also generalize to real-world environments, we use four real rooms from the Hearing-Anything-Anywhere Dataset [Re5]. We compare our method against Diff-RIR [Re5], a physics-based differentiable RIR rendering pipeline that utilizes 12 reference RIRs per room to predict RIRs for new locations. As shown in Table 8.2, our model compares favorably against all baselines. In partilar, despite using only 8-shot references, our method outperforms Diff-RIR that uses 12 reference RIRs in all acoustic metrics, demonstrating its strong generalization capabilities. We observe that our method underperforms on the T60 metric compared to the Nearest Neighbor baseline across all four rooms. We suspect this is because T60, as a global metric, is more sensitive to measurement noise due to its aggregation of all acoustic interactions within the room. Our learning-based method can struggle with low SNR beyond the early parts of the waveform, as it is trained on simulation data with higher SNR than real room measurements. In contrast, EDT focuses on early reflections with high SNR, make it less noise-sensity, and C50 is similarly robust due to noise smoothing in the integration beyond the early parts. Despite this, our results demonstrate that xRIR's effectiveness in adapting from simulated rooms to real environments, successfully capturing diverse room acoustics with fewer reference RIRs than prior methods.

8.3.6 Qualitative Results

We present qualitative results by comparing the predicted RIRs and acoustic maps between our model xRIR and the baseline methods, in both simulated and real environments.

RIR Predictions. In Figure 8.4, we visualize sample results of RIR waveforms on a simulated apartment and a real room with complex geometry. Side-by-side comparison shows that predicted RIRs from xRIR align more closely with the ground-truth RIR waveforms in the early part than baselines. This observation is consistent with the low acoustics metrics errors achieved by our method in the quantitative results shown in Table 8.1.

Acoustics Maps. Furthermore, we compute the RIRs at dense locations across the entire real rooms, and compute the clarity of the predicted and ground-truth RIRs to reconstruct acoustics maps according to the floor plans. As shown in Figure 8.5, across dense locations in these rooms, overall xRIR achieves better C50 distribution than Diff-RIR [Re5] compared to the ground-truth acoustic maps, especially at moderate-to-low intensity regions. These qualitative results demonstrate the effectiveness of xRIR in accurate RIR prediction in both simulation and real-world settings.

Chapter 9

FROM VISION TO AUDIO AND BEYOND

This chapter is based on the following published work: [XL7].

9.1 Motivation

Creating a general-purpose audio-visual model that solves both audio-visual understanding and generation task poses significant challenges. First, raw video and audio signals are both high-dimensional and time-dependent data, creating an intricate scenario for comprehension of their joint events and necessitating extensive training computations. Existing contrastive learning and masked autoencoder-based audio-visual learners [Re138, Re139] are deterministic models that cannot support generative purposes, and the reconstruction loss is made on the level of image and spectrogram patches. The state-of-the-art audio generative frameworks [Re56, Re140, Re25], on the other hand, usually perform modeling on the latent space and sophisticated models are often essential to generate high-quality audio, requiring multi-stage modeling [Re141, Re26, Re59]. It remains uncertain whether I can employ similar self-supervised representation learning techniques in the latent space while consistently producing high-quality audio. To address these challenges, I introduce an innovative audio-visual framework 'Vision to Audio and Beyond' (VAB), which is an efficient unified audio-visual framework capable to learn to associate audio with visual signals and enables vision-to-audio generation within the same model. At its core, this framework involves a pre-training task of predicting masked audio from visual inputs. In order to facilitate the learning of audio-visual representation and audio generation, I perform the pre-training task within the latent space instead of using raw images and audio spectrograms. Specifically, I tokenize

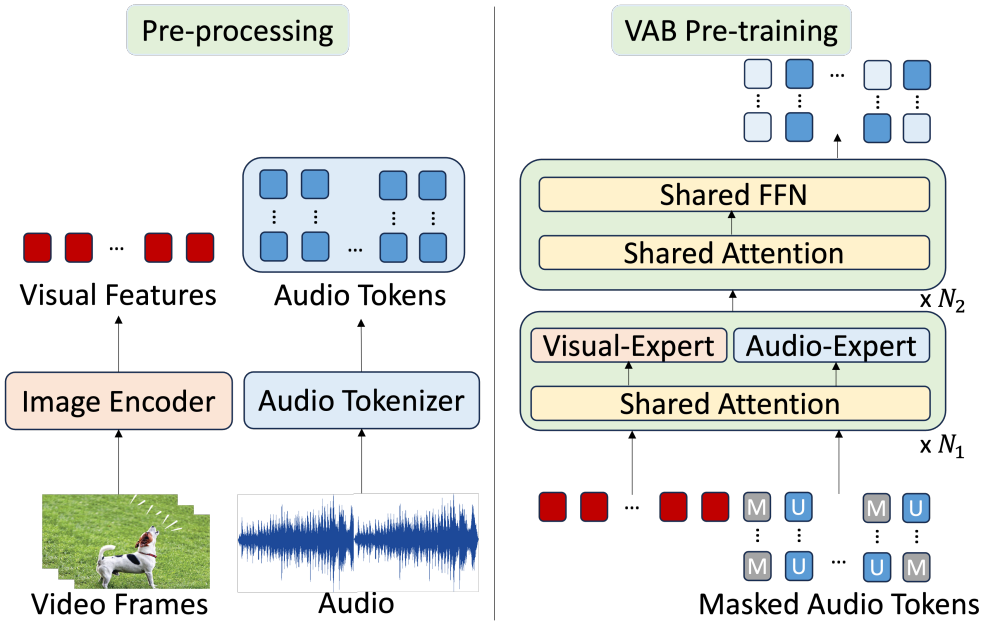

Figure 9.1: Pre-processing (left) and masked audio token prediction pre-training (right) of VAB framework. (Figure from [XL7])

the audio data into discrete tokens utilizing a public open pre-trained neural audio codec and extract frame-level visual features from a self-supervised pre-trained image encoder. During VAB pre-training, I employ an encoder-only multi-way transformer to predict masked audio tokens from visual features using a variable masking scheme. After completion of the pre-training phase, the VAB model can function as a uni-modal or multi-modal encoder, and it is prepared for fine-tuning in cross-modal retrieval and classification tasks. Meanwhile, it can also support zero-shot visual-conditioned audio generation through efficient parallel decoding strategies.

9.2 Methods

I detail the design of Vision to Audio and Beyond (VAB) framework and its capability of generating high-quality audio from silent video and acquiring semantic audio-visual rep-

resentations for subsequent tasks. I first describe VAB *pre-processing* stage that consists of converting raw audio and visual signals into latent spaces through the utilization of a pre-trained audio neural codec and image encoder. With audio tokens and frame-level visual features as inputs, I outline VAB components that facilitate *self-supervised pre-training* stage centered around masked audio token prediction conditioned on visual features. At this stage VAB establishes its representation and learns to generate audio for video. In a subsequent stage, I elaborate on how the pre-trained VAB model is leveraged for *vision-to-audio generation* and *fine-tuning across various downstream tasks*.

Audio and Video Transformation into Latent Spaces There are two primary motivations for converting audio and video frames into latent spaces. First, prior research [Re56, Re140, Re25] indicates that performing generative modeling within latent spaces enhances training convergence and facilitates the generation of high-quality audio samples. Second, operating on raw video frames instead of latents, entails significantly longer sequences and places a substantial computational burden. For instance, a ten-second video at 1fps results in 1960 patches when employing a standard Vision Transformer (ViT) [Re135]. For such number of patches even employment of tubelets does not result in significant relief and therefore more compact representation through latents could be advantageous. I thus have utilized a frozen pre-trained image encoder, such as CLIP [Re3], to extract frame-level features and serve as latents. This choice enables a significant reduction in the length of visual sequences and benefits the semantic features obtained from image-text pre-training. In fact, I discovered that these Ill-established image-text features can be adapted to capture audio-video relationships effectively, thereby mitigating the potential information loss associated with compressed audio tokens. Indeed, I use CLIP image encoder to extract frame-level features at a rate of 1fps, resulting in 10 seconds of video features $V \in \mathbb{R}^{10\times d}$, where d is the dimension of CLIP image features.

To transform audio waveforms into tokens, I explore two off-the-shelf pre-trained neural codec variants, DAC [Re142] and Encodec [Re11]. DAC and Encodec are both trained in a fully self-supervised manner on reconstruction tasks without reliance on any labels and pro-

vide audio tokens that represent compressed versions of the original signals. Both DAC and Encodec use K residual vector quantization (RVQ) to encode 1D audio waveform $A_w \in \mathbb{R}^{T_a}$ (16kHz) into audio tokens $A \in \mathbb{N}^{K \times S}$, where K is the number of residual codebooks and $S = T_a/d_a$, $d_a = 320$ is the downsample factor for both codec. A 10 seconds audio results in $A \in \mathbb{N}^{K \times 500}$ tokens in total. In DAC, it comprises $K = 12$ codebooks. These codebooks constitute a hierarchy wherein codes in lower levels represent coarse acoustic features, while codes in higher levels capture finer acoustic details. On the other hand, the Encodec contains only $K = 4$ codebooks, exhibiting poorer audio reconstruction quality compared to the DAC. To conduct a comprehensive study, I explored the use of both DAC and Encodec tokens. During VAB pre-training, I employed the first four levels of audio tokens $A_c = A^{0:4,500}$ for both DAC and Encodec. For the remaining 8 levels of DAC tokens, I folloId Vampnet [Re26], applying an additional coarse-to-fine model solely for the purpose of audio generation.
It is important to emphasize that both audio tokens and frame-level visual features are extracted before the VAB pre-training, which enables us to model temporal and audio-visual relationships during the pre-training process more efficiently.

Conditional Masked Audio Token Prediction

Masking: Given audio tokens A_c and visual features V, I employ visual-conditioned masked audio tokens prediction as the VAB pre-training task. First, I randomly mask the audio tokens using a masking strategy with variable masking ratios. Specifically, I sample the masking ratio M_r from a truncated Gaussian distribution centered at 0.55 with $std = 0.25$, left truncated by 0.5 and right truncated by 1. It is important to have a variable and reasonable portion of masked audio tokens to enable learning both the representation and the generation. I reuse the codebook embeddings in neural codecs and add a new $[MASK]$ token. The masked audio tokens $A_m \in \mathbb{N}^{4,500}$ are embedded and summed to obtain masked audio embeddings $A_{\text{emb}} \in \mathbb{R}^{500,d_{\text{emb}}}$, where d_{emb} is the embedding dimension. The visual features V are projected linearly to the same dimension as d_{emb} to have visual embeddings V_{emb} and to add modality-specific embeddings E_v and E_a to V_{emb} and A_{emb}, respectively. I then concatenate the visual and audio embeddings to form the final input sequence $x =$

$[V'_{\text{emb}}, A'_{\text{emb}}]$, where $V'_{\text{emb}} = V_{\text{emb}} + E_v$, $A'_{\text{emb}} = A_{\text{emb}} + E_a$.

Multiway Transformer Encoder: The architecture of VAB is similar to the Multiway Transformer Encoder [Re143, Re144] where each transformer layer contains a bi-directional multi-head attention shared for audio and visual embeddings, layer normalization, feed-forward networks, and residual connections. For VAB model with N layers, I use modal-specific feed-forward networks for the first N_1 layers and shared feed-forward networks for the rest of $N_2 = N - N_1$ layers. The shared bi-directional self-attention allows audio and visual embeddings to associate with each other during training. The modal-specific feed-forward networks can be considered experts in learning information specifically for audio or vision so that I can use the first N_1-th layers as either audio or visual encoders for single modality tasks. The upper-level joint feed-forward networks are useful for the more challenging vision-to-audio generation. Finally, multiple linear projection heads are used to predict each level of masked audio tokens.

Let A_u be the set of all unmasked audio tokens and VAB model parameters θ. The objective of the pre-training is to minimize the negative log-likelihood

$$l_{\text{vab}} = -\sum_{\forall a \in A_m} \log p(a|A_u, V, \theta) \tag{9.1}$$

The overview of VAB pre-processing and pre-training is shown in Fig 9.1.

Zero-Shot Video-to-Audio Generation After VAB pre-training, audio tokens can be generated using efficient iterative decoding similar to previous masked generative modeling approaches [Re32, Re145, Re26]. Specifically, I initialize all audio tokens by $[MASK]$ tokens and feed them into the VAB model along with the visual features V, similarly to pre-training. For each decoding iteration $t \in [0, t_T]$, the process involves predicting the token for the remaining masked audio token to obtain $\hat{a}_t$. Subsequently, I compute a confidence score z by incorporating the prediction log-probabilities and introducing temperature-annealed Gumbel noise

$$z(\hat{a}_t) = \log(p(\hat{a}_t)) + \alpha \cdot g_t, \tag{9.2}$$

where g_t represents an independently and identically distributed (i.i.d) sample drawn from

Gumbel noise, and α signifies the temperature that undergoes linear annealing, gradually reducing to 0 as the number of sampling iterations progresses. Following this, I sort the set of sampled tokens based on their confidence scores and determine the k loIst confidence tokens to re-mask in the subsequent sampling iteration. Specifically, the number of tokens to be re-masked, denoted as $k = \gamma(\frac{t}{t_T})N$, where N represents the total number of tokens, and γ follows a cosine schedule so that feIr masked audio tokens are replaced in the early iterations, while more masked audio tokens will be replaced in the later iterations. Additionally, classifier-free guidance [Re146] is also employed to achieve better visual adherence.

As a result, the sampled audio tokens are ready to be decoded by DAC or Encodec to generate audio waveforms. For DAC, the quality of the audio tokens decoded by coarse levels turns out to be insufficient [Re59, Re45, Re26]. Therefore, I train an additional coarse-to-fine model to generate fine-level audio tokens A_f conditioned on coarse tokens A_c. The coarse-to-fine model architecture is a bi-directional transformer encoder as Ill. In comparison to the coarse model in the previous stage, I do not use modal-specific feed-forward networks, and I use all levels of audio tokens instead of coarse-level tokens as inputs. I train the model using masked audio token prediction, but only fine-level audio tokens would be masked and predicted. Formally, training the model θ_{c2f} minimizes the objective

$$l_{\text{c2f}} = -\sum_{\forall a \in A_{f,m}} \log p(a | A_{f,u}, A_c, V, \theta_{\text{c2f}}). \tag{9.3}$$

During inference, generated coarse audio tokens and visual features will be used as the conditions and iterative decoding will be used in coarse-to-fine again for generation. Unlike autoregressive generation, which generates tokens one by one, our approach generates them simultaneously. This significantly reduces the total number of inference steps while still achieving, and in some cases even surpassing, the performance of the autoregressive approach.

Adaptation of VAB for Retrieval After VAB pre-training, the learned representation is amenable for adaptation to a variety of audio-visual tasks through fine-tuning. I therefore proceed to fine-tune the first N_1 modal-specific layers of the VAB model using the contrastive loss, aiming to align audio and visual modalities for retrieval tasks. While it might seem

logical to apply masked prediction and contrastive loss concurrently during the VAB pre-training, our early experiments revealed that such a training strategy led to failure of both tasks and could not converge. Additionally, I observed that initializing contrastive training from scratch on the first N_1 layers required more training epochs to converge to a similar performance as the one initialized from pre-trained VAB with masked audio token prediction. Furthermore, fine-tuning for masked audio token prediction initiated from the contrastive pre-training model did not aid the convergence of masked audio tokens prediction. As a result of these insights, I apply contrastive training as fine-tuning task after the masked prediction pre-training phase. Specifically, I run two forward passes to obtain audio and visual output features from audio tokens and visual features, separately. The audio tokens are not masked at this stage. The resulting output features are then average-pooled and normalized. I fine-tune the model using the standard contrastive loss L_c

$$l_c = -\frac{1}{N}\sum_{i=1}^{N}\log\left[\frac{\exp s_{i,i}/\tau}{\sum_{k\neq i}\exp s_{i,k}/\tau + \exp s_{i,i}/\tau}\right], \tag{9.4}$$

where $s_{i,j} = ||a_i^c||^T||v_j^c||$, a^c and v^c are audio and video representations, and τ is the learnable temperature value initialized with 0.05.

Adaptation of VAB for Classification The latent representation can be adapted to additional tasks, for example, classification. For uni-modal classification tasks, I provide audio tokens or visual features as inputs to the first N_1 modal-specific layers of the VAB model. For audio-visual joint classification, I use both audio tokens and visual features as inputs and concatenate them into a joint sequence as in VAB pre-training. Similar to the contrastive fine-tuning, I do not mask audio tokens. I average-pool the output features of the N_1 layers and add a linear classifier for fine-tuning all classification tasks. While fine-tuned VAB pre-trained model achieves reasonable performances in the classification tasks, I found that the contrastive fine-tuned VAB model could serve as a better initialization for classification fine-tuning. This advantage may stem from the fact that both retrieval and classification tasks do not require masking of audio tokens, thereby enabling a more seamless transfer of knowledge. This effect is particularly evident in audio-only and audio-visual

classification tasks.

9.3 Experiments and Results

	AudioSet Eval Subset			VGGSound Eval Subset			MSR-VTT (Zero-shot)		
	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
Video → Audio									
LanguageBind [Re147]	6.4	20.2	28.3	10.3	30.1	39.7	1.9	6.2	8.8
ImageBind [Re148]	22.1	43.2	52.6	21.6	43.4	52.9	7.0	18.5	25.2
CAV-MAE [Re138]	18.8	39.5	50.1	14.8	34.2	44.0	13.3	29.0	40.5
VAB-DAC (Ours)	35.5	61.8	72.4	30.8	59.6	70.8	**13.8**	30.6	40.1
VAB-Encodec (Ours)	**39.5**	**65.4**	**74.6**	**33.5**	**63.3**	**74.3**	14.2	**31.1**	**42.0**
Audio → Video									
LanguageBind [Re147]	4.4	15.0	22.5	6.5	22.7	33.5	1.2	4.4	6.9
ImageBind [Re148]	20.8	42.6	51.6	20.7	43.2	53.4	6.0	16.9	23.7
CAV-MAE [Re138]	15.1	34.0	43.0	12.8	30.4	40.3	7.6	19.8	30.2
VAB-DAC (Ours)	37.0	61.8	70.8	33.1	**62.7**	**73.8**	**12.0**	**27.3**	**36.2**
VAB-Encodec (Ours)	**37.5**	**64.0**	**73.7**	**34.9**	**62.7**	**73.1**	9.6	23.3	32.9

Table 9.1: Cross-modal retrieval results on AudioSet, VGGSound, and MSR-VTT. Values in bold highlight the best performance. (Table from [XL7])

Audio-Visual Retrieval. In this task, I assess the learned representations of VAB model for both audio-to-visual retrieval and visual-to-audio retrieval. I first fine-tune the VAB model using the contrastive loss, employing the same training dataset utilized during the pre-training phase. To evaluate the retrieval performance, I adopt the CAV-MAE [Re138] methodology for conducting retrieval on audio-visual samples sourced from the AudioSet and VGGSound evaluation set. Furthermore, I extend our evaluation to include zero-shot retrieval on MSR-VTT [Re149] test set. I feed audio tokens and visual features through the VAB model in two separate forward passes. Subsequently, I compute the mean-pooled

and normalized encoder outputs to derive audio and visual representations, respectively. I then calculate retrieval recall metrics at ranks 1, 5, and 10 (R@1, R@5, R@10) based on the cosine similarity of these audio and visual representations. Besides CAV-MAE, I expanded our comparison of retreival performances by comparing VAB with two additional open-source multi-modal alignment models: ImageBind [Re148] and LanguageBind [Re147]. These models primarily utilize images or text, respectively, as the main modality to unify various modalities within the latent space. The comparison results are presented in the Table 9.1. Notably, I found that VAB models performed consistently better on Audioset and VGGSound than all of the baselines by a large margin (x2 improvement on AudioSet and VGGSound). I also discovered that the video-to-audio retrieval is generally better than the audio-to-video retrieval for all datasets. I further observed that while LanguageBind is known for its effective performance in text-centric tasks compared to ImageBind, its effectiveness does not generalize well in correlating audio and visual modalities.

Audio-Visual Event Classification. In this task, I evaluate the quality of VAB representations in the context of audio-visual event classification task. To accomplish this, I employ the contrastive VAB model and fine-tune it on three distinct datasets: 1) AudioSet-20K, 2) AudioSet-2M, and 3) VGGSound. During the classification fine-tuning stage, I retain the first N_1 layers of the model and add a linear classification head. Our model is fine-tuned using audio-only data (A), video-only data (V), and audio-visual data (V+A), enabling us to evaluate both single-modal and multi-modal representation quality. The results of the evaluation are presented in the Table 9.2. Both VAB-DAC and VAB-Encodec exhibit competitive performances across A, V, and V+A classification tasks, with a slight advantage for VAB-Encodec. Notably, visual-only (V) classification outperforms previous methods across three datasets, highlighting the effectiveness of incorporating frame-level CLIP embeddings as our visual features. Additionally, I observed a notable performance gap in the audio-only (A) when compared to the best-performing methods. This outcome can be attributed to the fact that audio tokens represent a lossy compression of the original audio. Without the guidance of visual features, correctly categorizing audio becomes a more challenging task.

Method	VGGSound (Acc)↑			AS-2M (mAP)↑			AS-20K (mAP)↑		
	V+A	V	A	V+A	V	A	V+A	V	A
Audio-visual Models									
G-Blend [Re150]	-	-	-	41.8	18.8	32.4	37.8	22.1	29.1
Perceiver [Re151]	-	-	-	44.2	25.8	38.4	-	-	-
Attn AV [Re152]	-	-	-	44.2	25.7	38.4	-	-	-
CAV-MAE [Re138]	65.5	47.0	59.5	51.2	26.2	46.6	42.0	19.8	37.7
MBT [Re153]	64.1	51.2	52.3	49.6	31.3	41.5	43.9	27.7	31.3
MAViL [Re139]	**67.1**	50.9	**60.8**	**53.3**	30.3	**48.7**	**44.9**	24.8	**41.8**
VAB-DAC (Ours)	63.9	**55.4**	48.2	47.0	33.3	36.2	38.9	28.3	28.8
VAB-Encodec (Ours)	65.2	55.1	51.3	47.7	**33.5**	38.6	38.7	**29.0**	29.0

Table 9.2: Comparison to previous audio-visual models on VGGSound, AS-2M, AS-20K in audio-visual (V+A), video-only (V) and audio-only (A) classification tasks. Values in bold represent the best performance. (Table from [XL7])

This observation aligns with findings in self-supervised learning in the image domain using quantized tokens [Re145]. However, VAB pre-training significantly improves supervised training using visual features and audio tokens from scratch.

Audio-only Classification. To assess the generalization of the acquired audio representations, I further evaluate the pre-trained VAB model by transferring it to other speech-only or audio-only tasks outside its original domain. In particular, I follow MAViL [Re139] and conduct experiments on the Environmental Sound Classification (ESC-50) [Re154] and Speech Commands (SPC-v1) [Re155] datasets. In these experiments, only the audio branch of VAB is fine-tuned. The results, presented in Table 9.3, demonstrate that VAB achieves competitive performance to recent supervised and self-supervised models. These findings underscore the adaptability and transferability of VAB, as it can seamlessly transition from audio-visual self-supervised pre-training to audio-only downstream tasks.

Method	ESC-50	SPC-1
AST [Re2]	88.7	95.5
SS-AST [Re156]	88.8	96.0
Aud-MAE [Re157]	94.1	96.9
MAViL [Re139]	**94.4**	**97.4**
VAB-DAC (Ours)	89.2	95.1
VAB-Encodec (Ours)	91.4	96.1

Table 9.3: Comparison with ESC-50 and SPC-1 audio only classification accuracy. (Table from [XL7])

Chapter 10

CONVERSATIONAL MUSIC RECOMMENDATION FROM VIDEOS

This chapter is based on the following published work: [XL8].

10.1 Motivation

In audio-visual understanding, incorporating natural language as an additional modalities can improve the tasks performance. This is especially the case when the additional text can provide rich contexts that complement audio or visual information. In particular, for video music recommendation system, the textual context can give sufficient description about the user's preference, which can serve as an important cue for music being recommended. Therefore, I propose a system called "MuseChat", a comprehensive conversational music recommendation system for videos. As shown in Figure 10.1, MuseChat consists of two main modules: the music recommendation module and the sentence generator module. Users begin with uploading a video and receive a music recommendation tailored to the video's content. MuseChat enables user-system interactions through dialogues in natural language. At each dialogue turn, users are empowered by the music recommendation module in MuseChat. They can refine recommendations by specifying their preferences in natural language, such as mood, genre, instruments, theme, and artist details. This process continues until they identify their desired music track. Another distinguishing feature is the system's interpretability versus the "black box" nature of conventional music recommendation models. Our sentence generator module can not only justify the recommended music with reasons but also craft personal narratives for users based on the selected music.

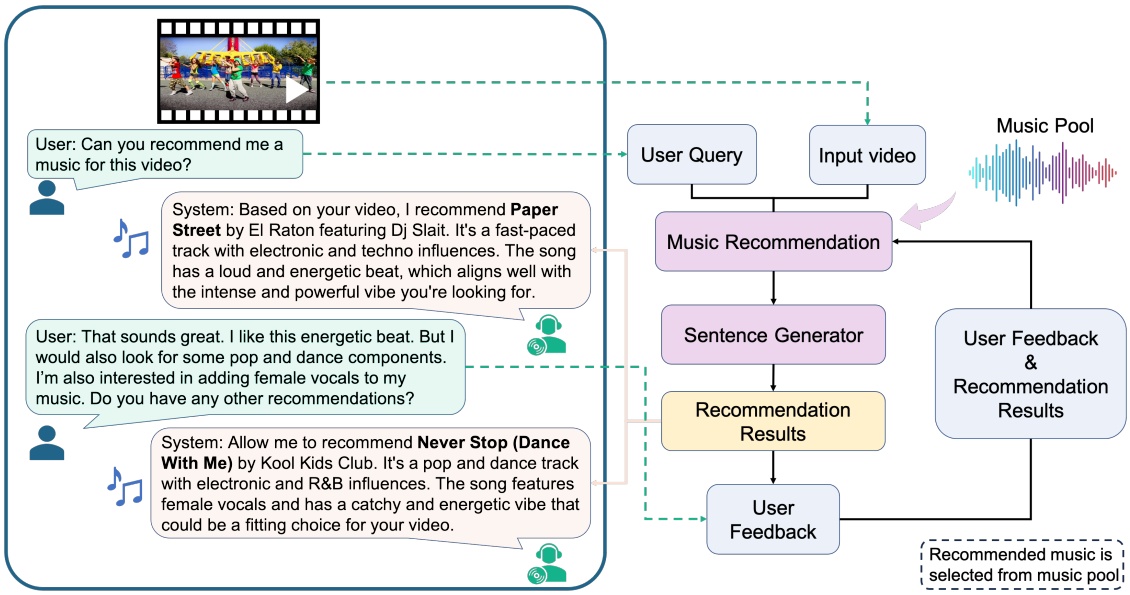

Figure 10.1: MuseChat features two modules: the Music Recommendation Module, which processes either video input alone or in combination with user prompts and past music suggestions, and the Sentence Generator Module, which uses these inputs to create natural language music recommendations. (Figure from [XL8])

10.2 Methods

We propose a novel approach to address the conversational music recommendation task on this dataset, establishing a new baseline performance. There are two main modules involved in the system: music recommendation and sentence generator.

Music Recommendation The goal of the music recommendation module is to select the most relevant music from the music pool using video, candidate music, and user text prompt. Each training sample is defined as a quartet (v, m_c, m_t, t_3), where v is the video, m_c denotes the candidate music track, m_t is the target music track and t_3 is the text of user prompt showing preferences. As illustrated in Figure 10.2, we focus on enhancing the model's ability to transit the recommendation from the existing candidate music m_c to target

music m_t. Intuitively, when user prompt is given as an additional context to the candidate music and original video, the information combined from these modality contexts should arrive at a representation staying as close to the target music as possible. This motivates our formulation of the recommendation module as a metric learning problem: To learn a common embedding space between the tri-modal combination (candidate music + video + user prompt) and music. Towards this end, we propose a novel MVT-Fusion Module, as shown in Figure 10.2 (right), that explores music-text-video fusion in a fine-grained manner to obtain the tri-modal embedding. The embedding then serves as a goal embedding with which the target music is aligned. We utilize contrastive learning for the embedding alignment. We detail MVT-Fusion Module and contrastive formulation respectively.

MVT-Fusion Module takes the candidate music m_c (represented as mel-spectrogram), user prompt t_3, and video frames v as inputs. Each input is encoded via pretrained encoder in their corresponding modality. The candidate music is encoded using an AST encoder [Re2] g^{m_c}. The user prompt is fed into CLIP [Re3] text encoder g^t to obtain contextualized language features (unpooled). Each video frame is transformed into a vector using CLIP image encoder g^v, and is then averaged along the temporal axis to obtain a single vector $\mathbf{x}^{\bar{v}}$, representing the semantics of the video.

To better capture the correlation between the candidate music and the user text prompt, we develop a fusion method for merging the features from music and text modalities. We adopt the "late fusion" strategy, applying several self-attention layers to both output features from the CLIP text encoder and the AST encoder before fusion. The output features from self-attention layers of two branches are $\mathbf{x}^{t_3} \in \mathcal{R}^{n_t \times d}$ and $\mathbf{x}^{m_c} \in \mathcal{R}^{n_m \times d}$, and expanded as:

$$\begin{aligned} \mathbf{x}^{t_3} &= \left[x^{t_3}_{\text{cls}}, x^{t_3}_1, \ldots, x^{t_3}_{(n_t-1)}\right], \\ \mathbf{x}^{m_c} &= \left[x^{m_c}_{\text{cls}}, x^{m_c}_1, \ldots, x^{m_c}_{(n_m-1)}\right], \end{aligned} \tag{10.1}$$

where variable with subscript cls serves as their summary of the respective sequence, along with the other elements capturing detailed features.

Then, we fuse the transformed features by implementing a cross-modal attention layer,

which is defined as: $\text{Att}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$, where d_k is the dimensionality of key vectors. Q and K, V are from two different modalities.

We add the video vector $\mathbf{x}^{\bar{v}}$ to obtain the tri-modal fusion vector representation $\mathbf{x}^f$:

$$\mathbf{x}^f = \mathbf{x}^{\bar{v}} + \text{Att}(x^{t_3}_{\text{cls}}, \mathbf{x}^{m_c}, \mathbf{x}^{m_c}) + \text{Att}(x^{m_c}_{\text{cls}}, \mathbf{x}^{t_3}, \mathbf{x}^{t_3}) \tag{10.2}$$

Contrastive Formulation Once $\mathbf{x}_f$ is obtained as the tri-modal fusion feature, it is necessary to align the representation of the target music with it. To achieve this, we first transform music from the pool into a vector space using a separate AST encoder g^{m_t}. Then we propose a contrastive learning approach to learn a common vector space between tri-modal fusion vector and music vector. Specifically, we use the hidden state of the last layer corresponding to the *cls* token as the music vector. We denote the target music vector as $\mathbf{x}_+^{m_t^{cls}}$, and any other music vector as $\mathbf{x}_-^{m_t^{cls}}$. Then, we formulate a contrastive loss aiming at keeping the vector distance between $\mathbf{x}_f$ and $\mathbf{x}_+^{m_t^{cls}}$ closer while pushing away $\mathbf{x}_-^{m_t^{cls}}$ from any other music, as illustrated in Figure 10.2 (left). Specifically, we use the Contrastive Multiview Coding Loss [Re158], a cross-modal variant of InfoNCE [Re159]. For each batch B, we have:

$$\mathcal{L}_{\mathcal{R}} = -\sum_{i=1}^{B}\left[\log \frac{h\left(\mathbf{x}^f_{(i)}, \mathbf{x}^{m_t^{cls}}_{(i)}\right)}{\sum_{j\neq i} h\left(\mathbf{x}^f_{(i)}, \mathbf{x}^{m_t^{cls}}_{(j)}\right) + h\left(\mathbf{x}^f_{(i)}, \mathbf{x}^{m_t^{cls}}_{(i)}\right)}\right], \tag{10.3}$$

where $\mathbf{x}^f_{(i)}$ and $\mathbf{x}^{m_t^{cls}}_{(i)}$ are i-th fusion vectors and target music vectors in the batch respectively. $h(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{\mathbf{x}^\top \mathbf{y}}{\tau}\right)$ is a discriminating function, with τ being a trainable temperature hyperparameter.

Sentence Generator To equip MuseChat with reasoning capability, we propose a sentence generator to provide justification for the recommended music. Towards this end, we design a multi-modal LLM, as illustrated in Figure 10.3, using Vicuna-7B [Re160] (derived from fine-tuning Llama2-7B [Re89] model) as the backbone. Each training instance comprises a music representation, denoted as $\mathbf{x}^{\bar{m}_t}$, which is the average of the music embedding $\mathbf{x}^{m_t} = [x^{m_t}_{\text{cls}}, x^{m_t}_1, \ldots, x^{m_t}_{(n_m-1)}]$, derived from the previously fine-tuned AST Encoder g^{m_t}. The

instance also includes the corresponding recommendation reasoning statement, t_4, from the simulated conversations. Notably, in our settings, each piece of music, whether as a candidate or target, has a corresponding reasoning statement in a conversation. Therefore, we do not specifically use the reasoning statement of candidate music in the first turn, t_2, because when this candidate music serves as the target in the other conversation, its reasoning statement is already utilized. We apply a linear layer f_l to project the averaged music features $\mathbf{x}^{\bar{m}_t}$ onto the text embedding space. To increase training efficiency, we leverage LoRA [Re127] to fine-tune the attention and output layers of Vicuna. We use next token prediction task aiming at minimize the negative log-likelihood of response tokens conditioned on the recommended music:

$$\mathcal{L}_{\mathcal{G}}\left(\mathbf{y};\theta\right) = -\sum_{i=1}^{n} log\left[p_\theta\left(y_i \mid \left[f_l\left(\mathbf{x}^{\bar{m}_t}\right), y_{<i}\right]; \theta\right)\right], \tag{10.4}$$

where y_i is the i-th token in the response $\mathbf{y}$, and θ is the trainable parameters.

Although we train the sentence generator using only music embeddings, during inference, we also input the title of suggested music from the music recommendation module. This is necessary because the model cannot generate accurate names of unseen music tracks.

10.3 Experiments and Results

Ranking Evaluation Following the setup for measuring music retrieval with free-form natural language tasks as described in [Re163], we randomly create non-overlapping music pools using test data, each containing 500 tracks. Each pool has only one target music track for each video. For the track-level testing, we calculate embeddings for all 12 segments of each 120-second video and music track. We then take the average of these 12 embeddings to create a single representative embedding for each video and each music track. Using these averaged embeddings, we evaluate the performance of the music recommendation module. In the first turn, music is suggested based solely on video features, as we assume that the user does not provide any specific preferences at this point. In the second turn, we include the user's text

Model	MR ↓	R@1 ↑	R@5 ↑	R@10 ↑	SR ↑
VM-NET [Re161]	28	5.31	18.14	28.78	-
MVPt [Re162]	7	20.71	48.89	63.14	-
ViML [Re163]	5	22.61	52.43	66.56	-
Sum-Fusion	17	10.58	28.47	40.19	21.70
Self-Attn Fusion	5	21.40	50.83	65.29	28.37
Cross-Attn Fusion	3	25.71	56.97	71.07	31.48
MuseChat (ours)	**2**	**32.79**	**63.92**	**76.53**	**40.49**
Chance	250	0.20	1.00	2.00	0.40

Table 10.1: Music retrieval results for baseline models and MuseChat. (Table from [XL8])

prompt and candidate music along with the video features. This setup allows us to evaluate the system's ability to modify its initial recommendations based on the new information. For both turns, we rank music tracks by calculating the cosine similarity between the features of the music in the pool and the fusion features. We use various metrics such as Recall@K for K = 1, 5, 10, and Median Rank (MR) to evaluate the performance of the recommendation. We also measure the "success rate" (abbreviated as SR), defined as the percentage of videos whose target music track appears at least once as the top of the recommended list within two turns. Since the MVP model cannot accommodate the user prompt as an input for the second turn, its SR is same as R@1. We report the average performance for each of these metrics across all test music pools.

Baselines Comparison We introduce six baseline models with various structures and modalities to assess the effectiveness of MuseChat in ranking. (1) **VM-NET** [Re161]. We re-implement it in PyTorch and replace the audio and vision backbone with AST [Re2] and CLIP [Re3] respectively for better performance. (2) **MVPt** [Re162] does not have publicly available source code. Thus, we replace their DeepSim [Re164] audio encoders with the publicly available AST [Re2] model. (3) **ViML** [Re163] also lacks publicly available source code. We replace their DeepSim audio encoders with the publicly available AST model and use their proposed text dropout strategy. Text and video features are fused

using Transformer blocks, following the settings described in their paper. (4) **Sum-Fusion** retains the architecture of MuseChat but employs a different fusion mechanism by summing up the vectors of the video, music, and text directly. We use the mean-pooled vector for both candidate music encoded by AST and text features extracted by CLIP text encoder. (5) **Self-Attn Fusion** extends the summation approach by including self-attention layers for text and music modalities before fusion, capturing intra-modal dynamics. (6) **Cross-Attn Fusion** removes self-attention layers and keep the cross-attention layer between music and text branch. We train the baseline models using the same music video pools and loss function as MuseChat. As shown in Table 10.1, MuseChat outperforms all baseline models. In particular, in R@1 metric, our model achieves a significant gain of **+7.0%** against "Cross-Attn Fusion" and **+10.1%** against the ViML model, showing the effectiveness of our fusion approach as well as the use of all three modalities as inputs. Notably, for models that could take video, candidate music and user prompt as inputs, we report the recall and MR for the second turn to show the effectiveness of our model when dealing with user preferences. The first turn results for these models are detailed in the supplementary materials. Additionally, we explore the contributions of various modalities to the retrieval performance in our model, with these specifics also included in the supplementary materials.

Modality Ablation Studies To further show the necessities of combining all three modalities for retrieval, we conduct ablation studies by removing video, candidate music, and text branch in turn during training. When excluding candidate music or text features, we remove the related cross-attention layers, retaining only the self-attention layers, with the *cls* token used for the summation of modality features. As Table 10.2 indicates, the model without visual branch, which is only valid for the second turn, exhibits the worst performance. This reveals the significance of visual information. Additionally, when comparing the MVPt [Re162] model with MuseChat without candidate music, we observe that user prompt helps to retrieve music more accurately.

Reasoning Evaluation We introduce two baseline models to highlight the significance of training our sentence generator module using both music embeddings and music titles as

Model	Modality	MR ↓	R@1 ↑	R@5 ↑	R@10 ↑	SR ↑
MVPt [Re162]	Video → Music	7	20.71	48.89	63.14	20.71
MuseChat w/o Video	(Music, Text) → Music	19	8.12	24.53	37.11	8.12
MuseChat w/o Candidate Music	(Video, Text) → Music	5	22.67	51.53	65.42	26.02
MuseChat (ours)	(Video, Music, Text) → Music	**2**	**32.79**	**63.92**	**76.53**	**40.49**
Chance	-	250	0.20	1.00	2.00	0.40

Table 10.2: Ablation Studies: Comparing MuseChat's Performance Without Certain Modality Branches and training from scratch. (Table from [XL8])

Model	Input Modality	BertScore [Re6] (f1) ↑	AB Divergence [Re7] ↓	$\mathcal{L}_2$ Distance ↓	Fisher-Rao Distance [Re7] ↓
Vicuna-7B	Music Title	0.9453	3.93	0.382	2.11
Vicuna w/ Music	Music Embeddings	0.9526	2.68	0.279	2.02
MuseChat (ours)	Music Title + Embeddings	**0.9676**	**1.51**	**0.208**	**1.47**

Table 10.3: Comparison of semantic similarity between output and simulated conversations using various metrics. BERTScore [Re6] assesses token-level similarity, while AB Divergence, $\mathcal{L}_2$ Distance, and Fisher-Rao Distance are derived based on InfoLM [Re7]. (Table from [XL8])

inputs. The first baseline employs the frozen Vicuna-7B [Re160] model, which is fine-tuned on the Llama2-7B [Re89] weights. Since this model cannot process music embeddings, we only present it with the music title. The second baseline (Vicuna w/ Music) uses the same architecture as our sentence generator module but takes only music embeddings as input. We employ various common NLG metrics [Re6, Re7] to evaluate the performance of the models on simulated conversations. As shown in Table 10.3, the Vicuna-7B model performs the worst. It struggles to capture the musicality of the given track, as it is a text-only model. As for Vicuna w/ Music, while capturing the musicality of the recommended track due to its training on both music and text, it fails to identify the correct music name and artist name based on audio information only. In contrast, by taking both audio information and music

title as inputs, our sentence generator module achieves the best performance on reasoning.

We also evaluate our model and two baselines via human assessment, employing a 5-point MOS scale to gauge correctness (music and artist identification), musicality (description of music traits), and clarity (response understandability and coherence). Table 10.4 shows the human evaluation results. Since Vicuna-7B only takes the music title as input, it achieves a better score in correctness compared to Vicuna w/ Music, which struggles to identify the title and artist of an unknown piece of music based solely on music characteristics. In contrast, the Vicuna w/ Music model captures musicality better than Vicuna-7B, which relies solely on its knowledge base for musicality assessment. MuseChat, however, achieves the best overall performance by both correctly identifying music and artist name and understanding music in the context of video and user preference.

Model	Correctness	Musicality	Clarity	Overall
Vicuna-7B	3.07	2.54	3.60	3.07
Vicuna w/ Music	1.24	3.50	4.05	2.93
MuseChat (ours)	**4.63**	**4.22**	**4.54**	**4.46**

Table 10.4: Human evaluation scores for music reasoning outputs. (Table from [XL8])

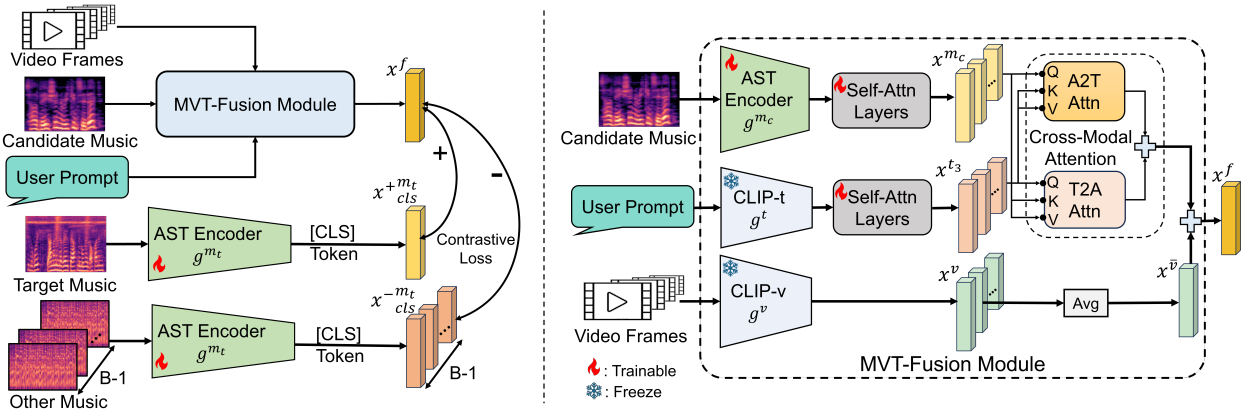

Figure 10.2: The music recommendation module combines video, the candidate music (1st round result), and user prompt (2nd round input) to retrieve a music in a common embedding space, trained using multi-modal contrastive loss (left). The candidate music corrected by user prompt along with the original video should result in a representation "closer" to the target music than other music. MVT-Fusion Module (right) is designed to combine the 3 modalities into an embedding space: i). The candidate music is encoded using the Audio Spectrogram Transformer (AST) [Re2]. ii). User prompt is fed into CLIP [Re3] text encoder (freeze) to get unpooled features. iii). Average pooling is performed on CLIP (freeze) vector of each video frame to obtain the video representation. iv). To foster fusion between candidate music and user prompt, self-attention layers and cross-modal attention (A2T and T2A, 'T' - text, 'A' - audio) are added to obtain the fused music-text vector. (Figure from [XL8])

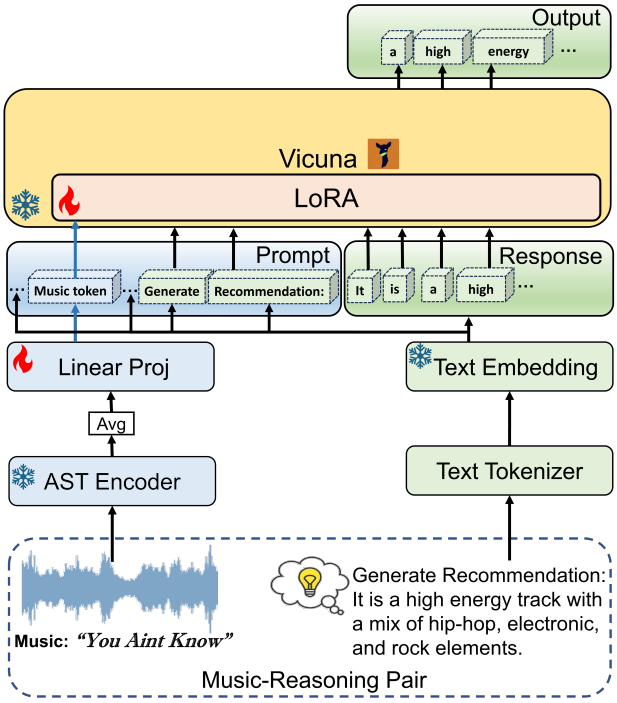

Figure 10.3: Illustration of sentence generator. During training, we only train the linear projection layer and the additional LoRA weights while keeping the parameters of Vicuna-7B and AST encoder frozen. And the prompt input is "*### Recommender: Music feature: <Music> [music token] </Music>; Generate Recommendation:*". During inference, we give the recommended music title as additional information, and the prompt input is "*### Recommender: Music title: [title]; Music feature: <Music> [music token] </Music>; Generate Recommendation:*". "*[music token]*" corresponds to the embedding vector projected from AST encoder output. (Figure from [XL8])

Chapter 11

UNBIASED AUDIO-VISUAL QUESTION ANSWERING BENCHMARK

This chapter is based on the following published work: [XL9].

11.1 Motivation

In recent years, there has been a growing emphasis on the intersection of audio, vision, and text modalities, driving forward the advancements in multi-modal learning research. More specifically, in the field of audio-visual learning, the interplay between audio and visual information provides a rich avenue for understanding dynamic scenarios. One particular task that embodies this synergy is Audio-Visual Question Answering (AVQA). As Figure 11.1 shows, given a musical instrument performance video and associated music audio, models are expected to answer questions that are related to them or the relationships therein. Unlike existing popular Visual Question Answering (VQA) task, which only tackles two modalities - vision and language, AVQA is designed to bridge and reason through all three modalities - vision, language and audio, which stimulates the merge of a new dataset. MUSIC-AVQA dataset, proposed to cater for this task, acting as an important benchmark, facilitate the research progress in this field.

The dataset consists of 3 major question categories by modality: Audio-Visual, Visual and Audio questions, respectively. Across all the categories, 5 question aspects are considered, including “Existential" - e.g. Is there a voiceover?, “Temporal" - e.g. Which violin makes the sound first?, “Counting" - e.g. How many sounding violins in the video?, “location" - e.g. Where is the performance?, and "Comparative" - e.g. Which object makes the sound first?.

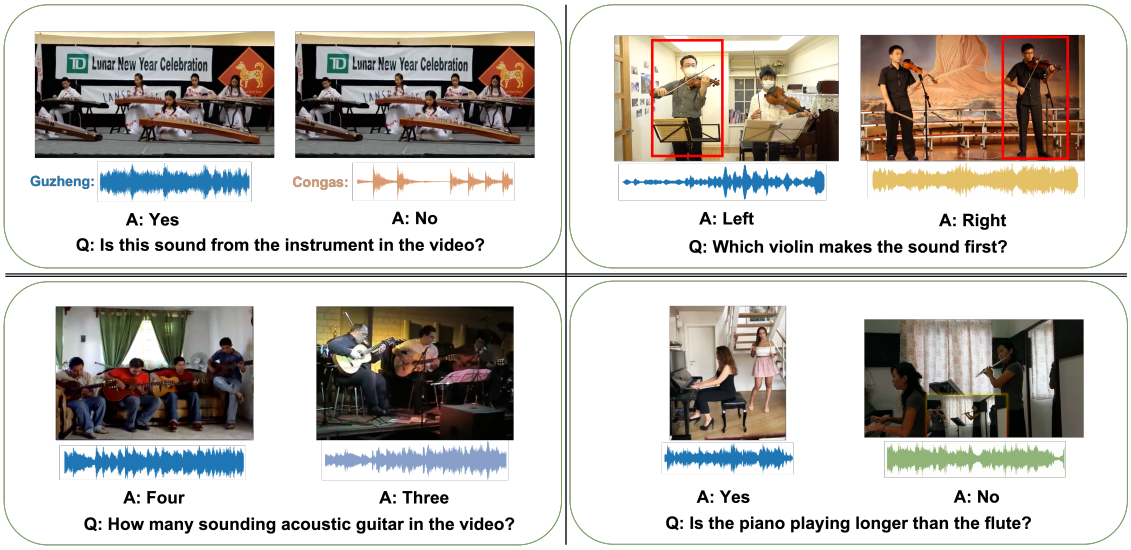

Figure 11.1: An overview of *MUSIC-AVQA v2.0 QA* samples. (a) showcases two videos with the same visuals but different audio; (b) both videos display two violinists in different orders; (c) videos present acoustic guitar ensembles of different counts; (d) features a piano-flute duet where the piano plays longer in the first video. To answer accurately, models must consider these audio-visual nuances rather than just language priors. (Figure from [XL9])

By joining the modality types with question aspects, the dataset ends up with 33 question templates, in which one can change the instrument name from one to another based on the video. Each question template contains a fix set of answers, ranging from binary answers (e.g. "yes" and "no") to counting answers (e.g. "1", "2", "3" etc.) and so on. Finally, MUSIC-AVQA dataset contains 9,290 (7,423 real, 1,867 synthetic) videos and 45,867 corresponding questions.

However, strong bias that exists in any modality can lead to the model neglecting the others. Consequently, the model's ability to effectively reason across these diverse modalities is compromised, impeding further advancement. We notice there is a strong bias exists in the dataset, which results in undermining the reliability of this dataset as a credible benchmark. For example, in a particular question category asking about whether or not the audio track

comes from the instrument played in the video, Over 90% of answer in the dataset is "yes". In audio-visual temporal question, when asking which instrument in the video sounds first, nearly 80% answer is “Simultaneously”. In counting questions, answers of small number like 1 and 2 alone could take up more than 50%. These imbalance exhibit across question aspects of “Existence", “Counting", “Temporal", “Location" and “Comparative”. Such outstanding bias could impact the model training negatively and thus lowering the ability of the underlying model, as model will be trained to favor towards to the most common answers in the training set and ignoring the importance of video and audio, as well as the reasoning between three of them. This problem has been explored in other tasks, for example, VQA [Re165] task, on which several datasets and models [Re166, Re167, Re168, Re169] are proposed to tackle it. But in AVQA, this problem is still remain untouched. In light of these observations, we endeavor to systematically address the issue and propose an improved version of the data, called MUSIC-AVQA v2.0.

Our initial step was to evaluate the answer distribution for each question template. Through this analysis, we identified specific templates that displayed a skewed answer distribution. After identifying these skewed templates, we proceed to select the question templates that have minority answers, designating them as our primary targets for balance. Recognizing the significance of holistic data representation, we take the additional step of manually collecting videos that corresponded with these specific question template and answer pairs. This ensures that the dataset is more representative and balanced.

Existing works have proposed models to the task of AVQA, which were trained on the public MUSIC-AVQA dataset. On the evaluation of the balanced dataset, we show that the strong data bias does misleading the model training, resulting in inferior performance. Moreover, we also contribute another model that has the ability to learning the connections across all the three different modalities. The model extends existing methods by: (i). adding an additional pretrained Audio-Spectrogram-Transformer (AST) branch for audio-visual grounding, and (ii). designing a cross-modal pixel-wise attention between audio and visual spatial maps.

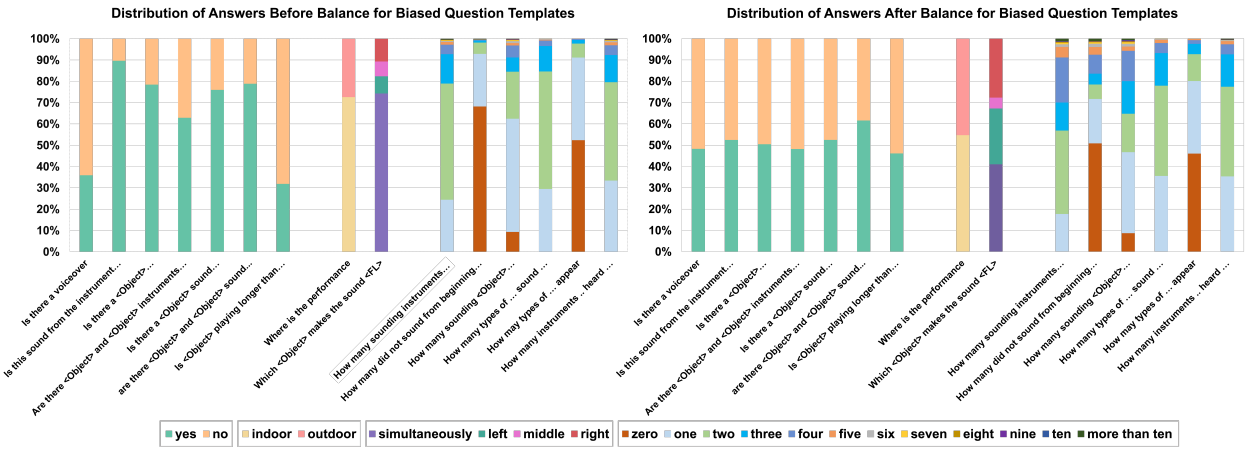

Figure 11.2: An overview of answer distribution in bias question templates before and after balance. (Figure from [XL9])

11.2 Dataset

We describe our approach to balance the dataset on the public MUSIC-AVQA dataset. This process entails two phases: pinpointing biased questions and then achieving balance. First, we scrutinize the distribution of answers for each question template. Specifically, we review questions templates under all the 9 questions types. Our criteria for identifying bias is when a single answer represents over 60% of responses for binary questions or exceeds 50% for multi-class questions (with more than two possible answers). Out of the 9 question types evaluated, 7 exhibited a skewed answer distribution for at least one of their templates. These include: audio-visual existential, audio-visual counting , audio-visual temporal, visual location, visual counting, audio counting and audio comparative questions. Within these types, multiple templates could exhibit biases, amounting to 15 out of 33 templates in total. A detailed comparison of the answer distribution before and after our balancing for each biased template is illustrated in Figure 11.2. We present the balancing process below.

Audio-visual Existential Questions: Binary questions dominate this question type, where the answers are either “yes" or “no". In this case, we simply pick the most frequent

answer and collect complementary pairs for it. We take two questions as an example: **Is this sound from the instrument in the video?** 90% of data samples in this question template are answered “yes". To create QA pairs whose answers are “no", we replace the audio track from the video with audio of another instrument type. To make the QA pairs non-trivial, we cluster the set of instruments into “string instrument", “woodwind instrument", “brass instrument" and “percussion instrument". When replacing the audio track, by 50% chance the audio track is replaced with a different instrument of the same cluster, while another 50% chance the audio track is replaced with instrument music belonging to other clusters. Using this method, we create 794 videos paired with non-matching audio segments. **Is there a voiceover?** Originally, this question is severely imbalanced where 79.6% of answers are “no". However, after carefully delving deep into the video data, we found that different individuals have different definitions to the concept of “voiceover", which led to inconsistencies in the labels. For example, some labelers define “voiceover" as human voice appearing on top of the instrument sound, while others define it as any general “off-screen" sound. To fix such inconsistency, we define it as **any “off-screen sounds”**. After manually checking 1,278 video-audio-question pairs from the training set, we corrected 169 mislabeled entries (13%). Despite this correction, a significant imbalance remained with 68% of labels still being no". To address this, we add another 456 QA pairs from our collected videos where “voiceover" presents (answered “yes"), resulting in a balanced distribution with 51.7% yes" and 48.3% no" answers.

Audio-Visual Counting Questions: This type of question asks about counting aspect of instruments in the video. The questions are structured using 4 templates that address the following aspect of counting: (i). the total number of sounding instruments (T1). (ii). the number of instrument types which are sounding (T2), (iii). the number of a specific sounding instrument type (T3), iv). and number of instruments that did not sound from the beginning to the end (T4). The answers are restricted to “0-10" and “more than ten". Upon analyzing the dataset, we observe a significant imbalance in the answers. Specifically, “0", “1" and “2" dominates the answers. For all four templates, the most frequent answer

exceeded 50% of the total, with one template even reaching 60%.

To balance audio-visual counting questions, we manually collect musical ensemble performance videos. Our goal is to gather videos of musical ensemble performances where at least one answer to the aforementioned question templates exceeded 2. We sourced potential videos from the YouTube8M [Re170] dataset, specifically targeting those tagged with terms like "musical ensemble," "string quartet," or specific instrument names. For each instrument type, we selected videos tagged with that type and combined them with videos tagged as "musical ensemble," "string quartet," "quartet ensemble," or a specific instrument name, which helps narrow down potential candidates. From this set, we manually filtered out videos that were of very low quality, had static scenes like album covers, or had ambiguous content. For each selected video, we annotated: (i). Total number of instruments, (ii). Number of distinct instrument types, (iii). Count of each instrument type, (iv). Number of sounding instruments, (v). Number of distinct sounding instrument types, (vi). Count of the most frequently appearing instrument that also produces sound, (vii). Number of instruments that did sound from the beginning to the end of the video. In total, we collected 591 videos for audio-visual counting questions. Using the annotations from these videos, we generated additional QA pairs for each template: 572 (+39%) for T1, 502 (+25.4%) for T2, 815 (+40.1%, 350 from originally unlabeled videos for the question template in MUSIC-AVQA dataset, 465 from our collected videos) for T3, and 413 (+30.3%) for T4. These new pairs have answers from the less frequent answer categories. After adding these pairs, imbalance issue from all 4 question templates is sufficiently mitigated, with the most frequent answer percentages decreasing by 16%, 17%, 15%, 13% for the 4 templates, respectively.

Audio-Visual Temporal Questions: This category of questions probes the order in which instruments play during a performance. The candidate answers range from 22 instrument categories, as well as positional indicators like “left", “middle", “right" that specify an instrument’s location. Additionally, the term“Simultaneously" denotes that instruments play at the same time. Among 3 question templates in this question category, the question “Which `<Object>` makes the sound first/last?" shows a strong imbalance: 74% of answers

are “simultaneously". To address this imbalance in a multi-class setting, we labeled QA pairs with the answers “left," “right," or “middle" to diminish the dominance of the “simultaneously" category. For example, consider a video where three violinists are performing. If the violinist on the left initiates the performance, followed by the middle and right violinists, we can formulate a QA pair as: "Q: Which violin starts playing first? A: left." In addition, we augment the video by horizontally flipping it. This transforms the QA pair to: "Q: Which violin starts playing first? A: right." Following the above procedure, we collected 203 additional targeted videos from YouTube for creating QA pairs. After augmentation, we end up creating 713 (+81.1%) additional QA pairs with answers rather than “simultaneously", reducing the most frequent answer percentage by 33% from 74% to 41%.

Visual Counting Questions: Unlike audio-visual questions, these questions rely solely on visual information. The first two templates, “Is there `<Object>` in the entire video” and “Are there `<Object>` and `<Object>` instruments in the video” determine the presence of specific instruments. For these templates, the majority of answers are "yes", constituting 78.4% and 62.7% respectively. To counter this bias, we generated 794 and 423 QA pairs with the answer "no" for each template. These pairs were created using labels from our collected videos.

The third template focuses on counting the types of instruments present in the video. There’s a notable imbalance here, with the answers “1" and “2" making up 91% of all responses. To address this, we used labels from the 591 videos from our audio-visual counting collection. We specifically chose 204 videos where the answer to this question exceeded 2. This selection helped reduce the dominance of the top two answer categories, bringing their combined percentage down from 91% to 80%.

Visual Location Questions: This category of questions pertains to the location of the performance and the specific positioning of a performer. It seeks answers to whether the performance is indoor or outdoor and inquires about the relative position of instruments in the video. Out of the four question templates in this category, the template “Where is the performance?" with answers either “indoor” or “outdoor”, exhibits an outstanding imbalance.

To address this, we collect 456 QA pairs whose answer is “outdoor”, from 456 videos that are not labeled for this question template from original MUSIC-AVQA dataset, resulting in reduction of QA pairs with the dominant category (“indoor”) by 17.9% from 72.6% to 54.7%.

Audio Counting Questions: The first two question templates, "Is there a `<Object>` sound?" and "Are there `<Object1>` and `<Object2>` sounds?", focus on the audio aspect, determining the presence of specific sounds. For these templates, the majority of answers are "yes", accounting for 76.0% and 78.8% respectively. To address this imbalance, we created 794 and 423 QA pairs with the answer “no" for each template, using labels from videos where the sounding instruments were identified. This reduced the dominance of the "yes" answer to 52.5% and 61.5% for the two templates, respectively. The third template queries the total number of distinct instrument sounds heard throughout the video. Notably, the answers "1" and "2" represent over 89% of all responses, highlighting a significant imbalance. To address the issue, we labeled 572 QA pairs from our collection of ensemble performance videos, specifically choosing videos where the answer to this question was neither "1" nor "2". After the balancing, We reduced the combined dominance of the "1" and "2" answers by 28% to 61%.

Audio Comparative Questions: This type of question compares different instruments sounding in the video in terms of loudness, duration and rhythm aspect. Among 3 question templates, a question template asking which instrument playing longer has a strong imbalance, where the answer “no” (indicating that neither instrument plays significantly longer) represents 68% of the data. To address this imbalance, we curated 182 QA pairs from previously unlabeled videos in the original dataset, ensuring that the first `<Object>` plays longer than the second `<Object>`. This reduces the dominance of “no” answer to 53.9%.

11.3 Methods

In addition to data balancing, we introduce a new model designed to set a robust baseline for MUSIC-AVQA v2.0. This model integrates existing audio-visual learning and QA components: (i). LAVISH [Re171], a 2-tower pretrained Swin-Transformer V2 [Re172] with

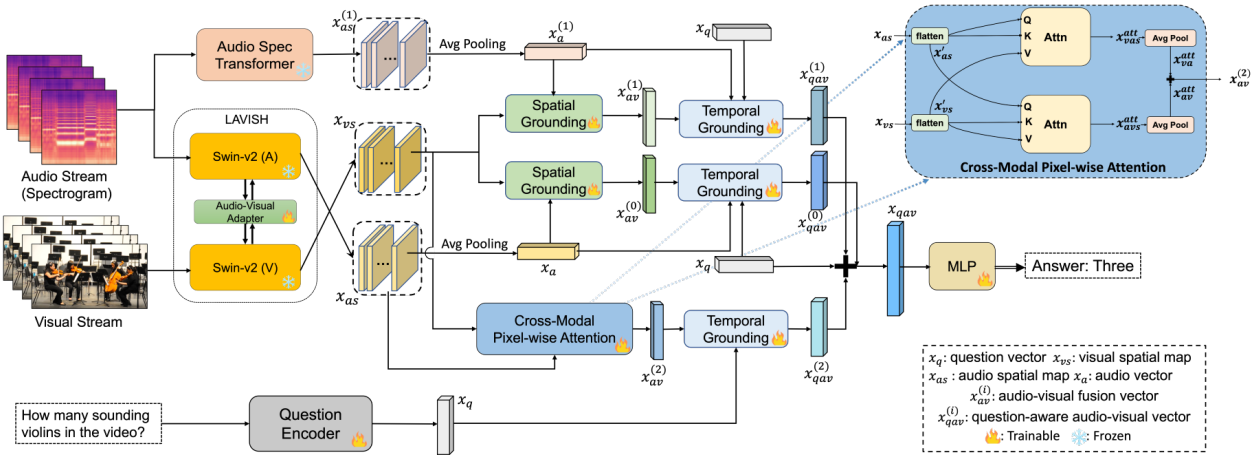

Figure 11.3: An overview of the new baseline model: (i). A pre-trained Audio-Spectrogram-Transformer (AST) branch is incorporated as an additional audio feature branch. (ii). A cross-modal pixel-wise attention module between audio and visual feature maps is designed to better capture the audio-visual correspondence at a granular level. (Figure from [XL9])

"Audio-Visual Adapter" for audio-visual fusion. (ii). A spatial & temporal grounding module from AVST [Re96]. Our extension to the existing model include: (i). A new audio-visual fusion branch, the "AST branch". This extracts audio feature from a pretrained Audio-Spectrogram-Transformer (AST) and merges them with the visual feature branch from Swin-v2 backbone, leveraging audio-visual spatial and temporal grounding [Re96]. (ii). A cross-modal pixel-wise attention module, further refining audio-visual fusion at a granular level. A detailed overview of the model is illustrated in Figure 11.3.

AST branch: Incorporating the pretrained audio feature as an auxiliary branch enables our model to capture richer semantic audio information compared to the vision pretrained transformer applied to the audio spectrogram in LAVISH. Specifically, we extract the final hidden output of the AST model, which is a spatial feature map $x_{as}^{(1)}$. We then apply the same operations as those used for the LAVISH [Re171] branch outputs, namely spatial and temporal grounding as introduced by AVST [Re96] (please refer to it for details). The feature map $x_{as}^{(1)}$ undergoes average pooling to produce a vector $x_{a1}^{(1)}$, which subsequently computes

spatial attention in relation to the visual map, x_{vs} , output from LAVISH's vision branch. The resulting spatial grounding module output is a vector in each frame, $x_{av}^{(1)}$, capturing visual features attended by audio. Following this, temporal grounding is performed. $x_{av}^{(1)}$ is concatenated with audio features, which are then attended by the question vector produced by a LSTM encoder, x_q, along the temporal axis. The output of temporal grounding module is a question-aware audio-visual fusion feature $x_{qav}^{(1)}$. This feature is concatenated with outputs from other branches and subsquently passed to a MLP for classification.

Cross-modal pixel-wise attention module: Existing spatial grounding module [Re96] uses a mean-pooled audio vector to compute attention with visual spatial maps. This attention approach: i) Losing the spatial details of spectrogram features. ii) is uni-directional, where only the audio vector serves as a query to the visual maps without any reciprocal interaction. To address these limitations, we propose a refined pixel-wise cross attention between the visual and audio maps, aiming to capture the correspondence between these two modalities more effectively. Specifically, given $x_{vs} \in \mathbb{R}^{H\times W\times C}$ representing frame-level visual map output from LAVISH, and $x_{as} \in \mathbb{R}^{H\times W\times C}$ representing spectrogram frame-level feature output from LAVISH, we compute two pixel-wise audio-visual attentions between two maps. We first flatten the spatial dimension of both feature maps to be $(HW)\times C$, resulting in x'_{vs} and x'_{as}. Then we compute mutual cross-attention between these two flattened maps, where each map attends to the other :

$$x_{vas}^{att} = x'_{as} + \text{Softmax}(\frac{x'_{vs}x'^{T}_{as}}{HW})x'_{as}, \tag{11.1}$$

$$x_{avs}^{att} = x'_{vs} + \text{Softmax}(\frac{x'_{as}x'^{T}_{vs}}{HW})x'_{vs} \tag{11.2}$$

The obtained x_{vas} and x_{avs} represent the pixel-wise fusion maps for audio and vision branch. We then normalize both maps and average pool their spatial dimensions to produce two vectors, x_{av} and x_{va}. These are concatenated to yield the final module output, $x_{av}^{(2)}$. $x_{av}^{(2)}$ is same dimension as $x_{av}^{(0)}$, $x_{av}^{(1)}$ - outputs of the spatial grounding modules from the other 2 branches. As a result, subsequent operations, such as temporal grounding, remain consistent with the other two branches.

11.4 Experiments and Results

11.4.1 Models Evaluation on Bias and Balanced Dataset

To further verify the data bias existing in the MUSIC-AVQA dataset and its negative impact to the model performance, we evaluate AVST and LAVISH models on two newly created datasets (biased and balanced datasets) here. We first create a balanced test set by sampling 20% from MUSIC-AVQA v2.0, with balanced answer distribution in every question categories using stratified sampling. Then within this balanced test set, we sample a bias test set by keeping the same QA distribution as MUSIC-AVQA dataset [Re96] (with our corrected QA pairs in “voiceover" category). Both the biased and balanced set test are used to evaluate all models.

After reserving the balanced test set from MUSIC-AVQA v2.0, the residual data remains balanced. To ensure a fair comparison, it is imperative that the training sets for both the bias and balanced datasets are of equal size. To achieve this, we first extract a bias subset from the leftover balanced data. To maximize the training samples in this bias set, any QA pair with an answer belonging to the most frequent answer category within its question template is incorporated into the biased subset.

Once the balanced test set is held out from MUSIC-AVQA v2.0, the remaining part is still a balanced dataset. To guarantee the fairness of comparison, we need to keep the size of training set same for the bias and the balanced set. To achieve this, we first sample the bias subset from the remaining balanced set. In order to maximize the number of training samples for the bias set, any QA pair whose answer is the most frequent answer category in its question template is included into the bias subset. For QA pairs in remaining answer categories in the question template, the numbers are determined by: $N_{most} \times \frac{N^{'}_{other}}{N^{'}_{most}}$, where N_{most} is the number of QA pairs whose answer is the most frequent answer category of the question template in our remaining balanced set, $N^{'}_{most}$ is the number of QA pairs whose answer is the most frequent answer category of the question template in MUSIC-AVQA training dataset, and $N^{'}_{other}$ is the number of QA pairs whose answer is from another less

Table 11.1: Evaluate Existing Models on Balanced Test set. Highlights are results where both models trained on balanced set consistently outperform trained on bias set. (Ext: Existential. Cnt: Counting. Temp: Temporal. Comp: Comparative.) (Table from [XL9])

Model	Train set	Total	Audio-Visual					Visual		Audio	
			Ext	Temp	Cnt	Loc	Comp	Cnt	Loc	Cnt	Comp
AVST [Re96]	bias	69.40	70.16	61.94	62.99	63.26	63.94	75.02	78.85	77.86	63.61
	balanced	**71.15**	**71.38**	59.98	**68.85**	63.48	65.40	**77.61**	77.80	**82.13**	62.05
LAVISH [Re171]	bias	70.39	68.11	60.08	66.72	65.11	63.12	78.14	81.46	78.80	60.96
	balanced	**73.35**	**72.04**	63.19	**71.95**	67.07	63.40	**80.40**	82.66	**83.57**	63.14

Table 11.2: Evaluate Existing Models on Bias Test set (Ext: Existential. Cnt: Counting. Temp: Temporal. Comp: Comparative.) (Table from [XL9])

Model	Train set	Total	Audio-Visual					Visual		Audio	
			Ext	Temp	Cnt	Loc	Comp	Cnt	Loc	Cnt	Comp
AVST [Re96]	bias	**73.07**	**85.68**	66.02	69.97	63.26	63.94	77.50	77.89	83.56	65.57
	balanced	72.01	75.36	64.68	70.82	63.48	65.40	77.91	76.75	84.44	62.11
LAVISH [Re171]	bias	**74.59**	**84.79**	67.84	73.53	65.11	63.12	80.77	81.14	84.54	63.59
	balanced	74.00	75.36	68.33	73.37	67.07	63.40	80.36	81.79	84.83	63.92

frequent category of the question template in MUSIC-AVQA training dataset. Once we create the bias subset, we sample another balanced subset by keeping the same number of samples as the bias subset. In the last step, we reserve 1/8 for validation, with the remaining 7/8 forming our biased and balanced training sets.

By following the outlined procedure, we create 2 datasets, a bias and a balanced set. Each dataset contains 31,513 training samples, mirroring the size of the original MUSIC-AVQA training set (approximately 31k). Additionally, both datasets share 4,502 validation samples, 10,819 balanced test samples, and 9,119 biased test samples. We then assess the performance

Table 11.3: Evaluation Results on Balanced Test Set: Our new baselines v.s existing models. (Ext: Existential. Cnt: Counting. Temp: Temporal. Comp: Comparative.) (Table from [XL9])

Model	Total	Audio-Visual					Visual		Audio	
		Ext	Temp	Cnt	Loc	Comp	Cnt	Loc	Cnt	Comp
AVST [Re96]	71.02	72.44	59.36	68.22	65.54	63.31	77.48	77.88	82.34	60.81
LAVISH [Re171]	73.18	73.83	60.81	73.28	65.00	63.49	81.99	80.57	84.37	58.48
LAST	74.85	74.08	59.15	75.17	**69.02**	**66.12**	83.19	83.41	85.75	61.59
LAST-Att	**75.44**	**76.21**	**60.60**	**75.23**	68.91	65.60	**84.12**	**84.01**	**86.03**	**62.52**

Table 11.4: Evaluation Results on Bias Test Set: Our new baselines v.s existing models. (Ext: Existential. Cnt: Counting. Temp: Temporal. Comp: Comparative.) (Table from [XL9])

Model	Total	Audio-Visual					Visual		Audio	
		Ext	Temp	Cnt	Loc	Comp	Cnt	Loc	Cnt	Comp
AVST	71.92	75.36	64.81	70.51	65.54	63.31	77.74	76.67	84.74	61.78
LAVISH [Re171]	73.51	74.92	66.5	75.08	65	63.49	82.08	79.59	85.32	59.14
LAST	75.24	75.58	65.78	76.16	**69.02**	**66.12**	83.63	82.36	**86.20**	61.78
LAST-Att	**75.45**	**76.47**	**66.75**	**76.20**	68.91	65.60	**83.86**	**83.09**	85.71	**63.10**

of 2 existing open-source models, the baseline AVST model [Re96] and the state-of-the-art LAVISH [Re171] model. Both models are evaluated on the biased and balanced test sets respectively, adhering strictly to their original training guidelines without any modifications. To validate the integrity of our code and data, we trained, validated, and tested both models using the original MUSIC-AVQA dataset. The AVST model achieved a total accuracy of 71.25% on the test set (compared to the reported 71.51%), while the LAVISH model achieved 77.17% (close to the reported 77.20%). These results align closely with the numbers reported in their publications.

We proceeded with training both models on the bias and balanced training sets, resulting in 4 distinct models. For models trained on the bias set, we use the total accuracy on the bias validation set to select the best checkpoint across epochs. Similarly, for models trained on the balanced set, we use the total accuracy on the balanced validation set for checkpoint selection. After training, we evaluated all four models on both the bias and balanced test splits. The evaluation results are presented in Table 11.1 for the balanced test set and in Table 11.2 for the bias test set. As shown in Table 11.1, both models trained using the balanced set achieve higher total accuracy on the balanced test set, with gains of **+1.75%** for AVST and **+2.96%** for LAVISH. Specifically, in question types where severe answer imbalance exists, such as audio-visual counting and existential questions, models trained on our balanced data consistently achieve higher accuracy on the balanced test set. For instance, the LAVISH model showed improvements of **+3.93%** and **+5.23%** respectively. Conversely, as shown in Table 11.2, all models trained on the bias set surpassed those trained on the balanced set when evaluated on the biased test set. These results suggest the data bias in original MUSIC-AVQA. Models trained from the data overfit to the biases, thereby undermining their ability to generalize well.

11.4.2 New Baseline Evaluation

To establish new baselines, we evaluate our proposed models using MUSIC-AVQA v2.0, our entire balanced dataset, comprising 36.7k QA pairs for training, 5,250 for validation,

and 10,819 for testing. We compare two model variants, 'LAST' and 'LAST-Att', against existing models. The LAST variant integrates the existing LAVISH model with the AST branch, as detailed in Section 11.3. LAST-Att represents our full model, incorporating both the AST branch and cross-modal pixel-wise attention. Further implementation specifics for both models can be found in the Supplementary Material. As shown in Table 11.3 and Table 11.4, both of our proposed baselines surpass the performance of LAVISH and AVST on both test sets. Notably, on the balanced test set, LAST-Att achieves a performance boost of **+2.26%** over LAVISH and **+4.42%** over AVST.

Chapter 12

CONVERSATIONAL AUDIO-VISUAL EMBODIED NAVIGATION

This chapter is based on the following published work: [XL10].

12.1 Motivation

The advent of powerful deep neural networks and sophisticated language models have led to significant advancements in building conversational agents that can collaborate with humans in solving challenging reasoning tasks [Re173, Re174, Re175, Re176]. While, much effort has been expended on tasks that are predominantly in the language domain, such progress is yet to percolate into real world problems that need complex reasoning over multiple modalities of perception. One such task that we exclusively explore in this paper is that of audio-visual navigation of an embodied robotic agent where the goal is to localize a sound producing source in a realistic, complex, and never-seen before environment when the sound is noisy, intermittent, sporadic, and mixed with other sounds — a situation even humans may find hard to tackle. As can be easily imagined, the applications needing such an *audio goal* capability are enormous; for example, at one end, we may think of a robotic disaster and emergency response agent that may need to move through huge rubble to localize victims who may cry for help and on the other, one may consider a home robotic vacuum repurposed to be vigilant to strange sounds.

While the task of navigating to the audio goal, has witnessed some attention in the recent years ([Re177]), we consider a variant of this task, dubbed audio-visual-language embodied navigation (AVLEN) ([Re178]), where the agent has the ability to interact with a

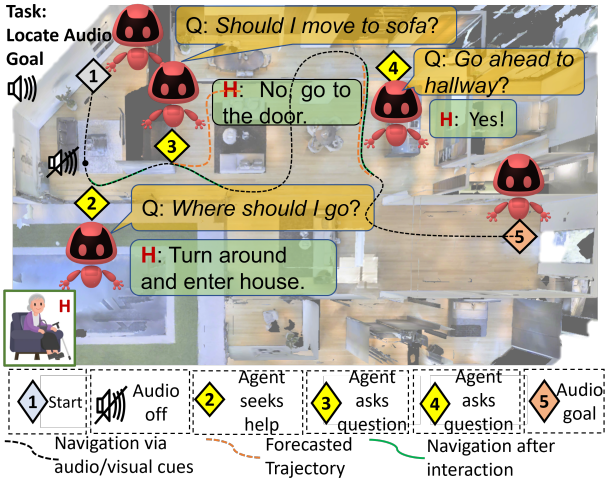

Figure 12.1: An illustrative CAVEN interaction: The agent starts at ① guided by the audio event at ⑤. At ②, the agent decides to seek help from the human/oracle H (e.g., because the audio stopped). The oracle then provides a short natural language instruction for the agent to follow. At locations ③ and ④, the agent decides to ask questions to the oracle using the forecasted trajectories (orange) and gets feedback, finally reaching the audio goal at ⑤. (Figure from [XL10])

human/oracle when it is unable to solve the task by itself and potentially query an oracle for task-specific guidance. However, the interactive abilities of the agent in AVLEN is limited in several aspects. In particular, the AVLEN agent could ask only a fixed question (e.g., “Help me!"), while the (human) oracle could provide a natural language response for guiding the agent to the goal. This technique of querying, while useful to some extent, does not cover the full scope of bi-directional interactions. As we know, back and forth interaction in natural language simulates a human-like conversation, allowing for better expressivity towards sharing ideas effectively. For instance, let’s assume for a moment that the agent is a 5 year old child who needs help in finding a sounding toy at a secret location. While the parents (oracle) could suggest: “look inside the wooden trunk” (as in [Re178]), the child might not know what a ‘trunk’ is. Instead, isn’t it better if the child had asked: "Should I look

next to the large brown box?" and the parents say: "yes"? or suggest "No, look inside it"? It is not only easy to respond with a 'yes'/'no' answer (if possible), but this also avoids the need to know what a 'trunk' is (and ask more questions or make wrong inferences). Engaging in conversations to resolve such ambiguities is of importance in several time-critical real-world circumstances, e.g., the sound of wheezing in an elderly care or a thud in a medical facility.

Our goal in this paper is to build a fully-conversational robotic agent, which we call CAVEN – Conversational Audio-Visual Embodied Navigation, with the capabilities as described above, that can engage in bidirectional interactions with an oracle in natural language towards solving the audio goal task in a complex realistic visual environment. Specifically, CAVEN can either use the audio-visual cues for its navigation (as in prior works [Re179, Re180, Re181]) or in case the agent is uncertain of which navigation step to take, it can interact with the oracle in two distinct modes: (i) a *question mode*, in which the agent forecasts a plausible trajectory based on audio-goal belief, using which it frames a natural language question to be posed to the oracle, and subsequently interpreting the oracle's response to the question, and (ii) a *query mode*, where the agent is unsure of what question to even phrase (e.g., when there are no useful cues in the scene) or completely uncertain about its current situation, and therefore directly seeks the oracle's guidance. Figure 12.1 illustrates a typical conversation between a human and our agent.

There are several challenges to tackle when designing the learning and inference model for CAVEN. Specifically, (i) when should the agent use language? (ii) what type of language interaction should the agent use (question or query)? (iii) how should the agent phrase the question? (iv) how to make the oracle understand the agent's question?, (v) how should the oracle respond to the agent's question? and (vi) how frequently should the agent be allowed to ask questions (budget)? Note that, some of these challenges are partially addressed in prior works [Re182, Re183, Re184] such as (v) and (vi). However in CAVEN, we tackle all these challenges within a single framework, by proposing a novel budget-aware partially observable semi-Markov decision process (POSMDP), using a reinforcement learning framework by introducing novel learning rewards.

To empirically assess the performance of CAVEN, we conduct extensive experiments on the semantic audio-goal navigation task [Re177] in the SoundSpaces simulator, under various challenging scenarios, each having intermittent sounds emanating from a source. One key difficulty to train the CAVEN model is the absence of any large scale dataset that includes language instructions in an audio-visual navigation setting. To this end, we introduce *AVN-Instruct* – a novel audio-visual-language navigation sub-instruction dataset with 41.5k pairs of audio-goal, trajectory, and language instructions. Our experimental results using the above setup clearly bring out the benefits of enabling the agent to converse with the oracle, demonstrating a solid gain of nearly 12% over competing approaches on the success rate.

12.2 Method

Task Setup: We assume the standard embodied audio goal problem setup [Re179], where the agent is equipped with an RGBD camera and a binaural microphone and at any time step can take one of four navigation actions: $\{\text{stop}, \text{move\_forward}, \text{turn\_right}, \text{turn\_left}\}$ in a densely-sampled 3D grid with the goal of locating the audio source. As in [Re179], we assume the sound is semantically unique and is produced by a static object, however the audio could be noisy, sporadic, or mixed up with other environmental sounds. An audio goal navigation episode is deemed successful if the agent calls the stop action within a given proximity to the goal.

Beyond the standard problem setup above, CAVEN can also seek language-based guidance from an oracle. Practically, the oracle could be a human who has higher level information about the scene, e.g., a remote operator controlling several such agents and intervening whenever needed, or a home owner who is notified about the situation and is sought to provide guidance. To incorporate the language modality into the audio goal setup, we follow AVLEN [Re178] in which the agent can query the oracle for help and the oracle responds via a short message describing a pathlet towards the audio goal. However, the interaction in AVLEN is only uni-directional and the agent cannot ask questions. Our CAVEN goes beyond this shortcoming and can phrase a question in free-form natural language using cues

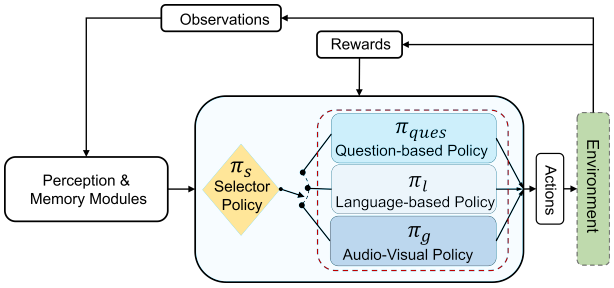

Figure 12.2: Architecture of our CAVEN model. We show the reinforcement learning policies, namely a selector policy π_s and three option policies π_g, π_l, and π_{ques}. (Figure from [XL10])

from the audio-visual context. Further, we assume the oracle after receiving this question, will either give a *"yes"* response if the oracle's interpretation of the question in its own state space results in actions that match its estimate of the actions along the ground truth geodesic to the goal. Otherwise, the oracle responds with a *"no"* followed by a short sentence guiding the agent to the goal. Note that the oracle in AVLEN has access to the 3D space of the full environment and thus can provide plausible instructions for navigation, however CAVEN has only a *very restricted view* of the scene in its vicinity, thus making this task of creating a question at the agent's side entirely different from that of the oracle's. In our new problem setup, we also assume that the number of times an agent can receive direct navigation instructions from the oracle (as a result of a wrong question or when it directly queries) is limited by a budget so that the agent only seeks help when necessary.

12.2.1 CAVEN Learning and Inference Framework

As we envisage CAVEN to incorporate various modules with diverse temporal spans, it is natural to consider a partially observable semi-Markov decision process (POSMDP) as our control module [Re185]. A POSMDP is essentially a partially observable Markov decision process (POMDP) with macro actions and is characterized by the tuple $(\mathcal{S}, \mathcal{A}, T, R, \Omega, \mathcal{Z}, \gamma)$ where $\mathcal{S}$, $\mathcal{A}$, T, R, and γ are the state space, action space, transition function, reward

function, and discount factor, respectively, while Ω and $\mathcal{Z}$ are the observation space and observation model. In a partially observable setup, the agent maintains a belief distribution b over $\mathcal{S}$, which is used to compute the expected reward. While in a POMDP setup, the agent maintains a policy $\pi : \mathbb{R}^{|\mathcal{S}|} \times \mathcal{A} \rightarrow [0, 1]$ that maximizes the expected reward, in POSMDP the agent maintains multiple low level 'options' as temporal abstractions, denoted Ξ, and a high level selector policy π_s to select the options from Ξ. An option $\xi \in \Xi$ is defined by the triplet $(\mathcal{S}^{\xi}, \pi^{\xi}, \beta^{\xi})$, where $\mathcal{S}^{\xi}$ is the set of valid states, π^{ξ} is the policy, β^{ξ} is the termination condition. In our setup, we disentangle agents' interactive audio goal navigation process into three low level temporal abstractions (i.e., options): i) audio-visual navigation ξ_g, ii) instruction-guided navigation ξ_l, iii) bi-directional question-answer navigation ξ_{ques}. We use π_g, π_ℓ, and π_{ques} to denote the respective policies of ξ_g, ξ_l, and ξ_{ques} and R'_g, R'_l, and R'_{ques} as corresponding immediate rewards. In our case, instead of using the termination condition for each option, we allow the audio-visual navigation option ξ_g to take a single step, while the interaction-based options (ξ_l and ξ_{ques}) are allowed a fixed span of ν steps (unless stop action is executed by these options). These options are assumed valid in any state of the environment, i.e., $\mathcal{S}_{\xi_g}, \mathcal{S}_{\xi_l}, \mathcal{S}_{\xi_{ques}} \in \mathcal{S}$.

Although the agent always has access to three option policies, it should maintain its autonomy and should only engage in a limited number of language interactions to mitigate its uncertainty. Further, in our setup, we have different levels of engagement of the oracle with the agent for varied language interactions (e.g., bi-directional conversations with question and answer, querying for language instructions) and a system should favor asking correct questions based on its audio-visual cues over relying on oracle instructions to reduce the oracle's effort. To consider all of these scenarios, we formulate option policies with dynamically adjusted constraints. These constraints are realized by penalties associated with the reward functions of each option policy. The audio-visual navigation policy $\pi_g : \mathbb{R}^{|\mathcal{S}| \times |M|} \times G \times |\mathcal{A}| \rightarrow [0, 1]$ chooses the navigation actions $a \in \mathcal{A}$ based on the audio-visual features. Here, M is a memory module storing a fixed number of past observations, and G is a set of audio goal estimates. Since, π_g is fully autonomous and

does not require oracle interaction, we encourage selecting this option by defining an unconstrained reward, $R'_g(b_t, a_t) = \mathbb{E}\left[\sum_{i=t}^{\infty} \gamma^{i-t} R'_g(b_i, a_i)\right]$. The instruction guided navigation policy $\pi_\ell : \mathbb{R}^{|\mathcal{S}|\times\nu} \times \mathcal{I} \times G \times |\mathcal{A}| \rightarrow [0, 1]$ navigates based on the received natural language instruction. Here, $\mathcal{I}$ is the set of all natural language instructions. Since, π_ℓ is entirely dependent on the oracle instruction, we penalize such interactions using ζ_ℓ, i.e., $R'_\ell(b_t, a_t) = \mathbb{E}\left[\sum_{i=t}^{t+\nu-1} \gamma^{i-t} R'_\ell(b_i, a_i)\right] - \zeta_\ell(t)$. The bi-directional conversational navigation policy $\pi_{ques} : \mathbb{R}^{|\mathcal{S}|\times\nu} \times \mathcal{Q} \times \mathcal{I} \times G \times |\mathcal{A}| \rightarrow [0, 1]$ navigates based on asking a question and receiving an answer. Here, $\mathcal{Q}$ is the set of all natural language questions. Specifically, π_{ques} consists of multiple novel components and the policy module can be divided in three submodules based on the functionality: i) question generator $\mathcal{G}^q$, ii) question evaluator $\mathcal{E}$, and iii) instruction generator $\mathcal{G}^i$. The output of π_{ques} depends on the interplay between these submodules. Question generator $\mathcal{G}^q$ is used to generate questions. Then, the question evaluator $\mathcal{E}$ evaluates on the oracle side if the question is correct. If the question is incorrect then the instruction generator $\mathcal{G}^i$ (which mimics the oracle) generates instructions for navigation. Since, asking correct question results in minimal oracle effort in producing a response, we define a dynamic penalty based on the question by, $R'_{ques}(b_t, a_t) = \mathbb{E}\left[\sum_{i=t}^{t+\nu-1} \gamma^{i-t} R'_{ques}(b_i, a_i)\right] - \zeta_{ques}(t, \mathcal{E}(q))$, where $q \in \mathcal{Q}$ and $\mathcal{E}(q)$ is an indicator function that checks whether the question q asked by the agent falls within the range of the estimated navigation direction by the oracle, and no penalty will incur when $\mathcal{E}(q) = 1$.

Putting it all together, our objective to learn these policies $\pi = \left\{\pi_s, \pi_g, \pi_\ell, \pi_{ques}\right\}$ is via maximizing the value function $V^\pi(b_0)$, i.e., $\arg\max_\pi V^\pi(b_0)$, where

$V^\pi(b) = \pi_s(\xi_g|b)\Big[R'_g + \sum_{o'\in\Omega} \mathcal{Z}'(o'|b, \xi_g) V^\pi(b')\Big] + \pi_s(\xi_\ell|b)\Big[R'_\ell + \sum_{o'\in\Omega} \mathcal{Z}(o'|b, \xi_\ell) V^\pi(b')\Big] + \pi_s(\xi_{ques}|b)\Big[R'_{ques} + \sum_{o'\in\Omega} \mathcal{Z}'(o'|b, \xi_{ques}) V^\pi(b')\Big]$.

Here, b' is the updated belief and $\mathcal{Z}'$ is the multi-time transition function [Re186] given by: $\mathcal{Z}'(o'|b, \xi) = \sum_{j=1}^{\infty} \sum_{s'} \sum_{s} \gamma^j \mathcal{Z}(s', o', j|s, \xi) b(s)$. Below, we detail the architecture of each of these policies.

Bi-directional Question-Answer Policy Module: Bi-directional question-answer policy consists of three components: (i) *TrajectoryNet* (forecasting short navigation steps), (ii)

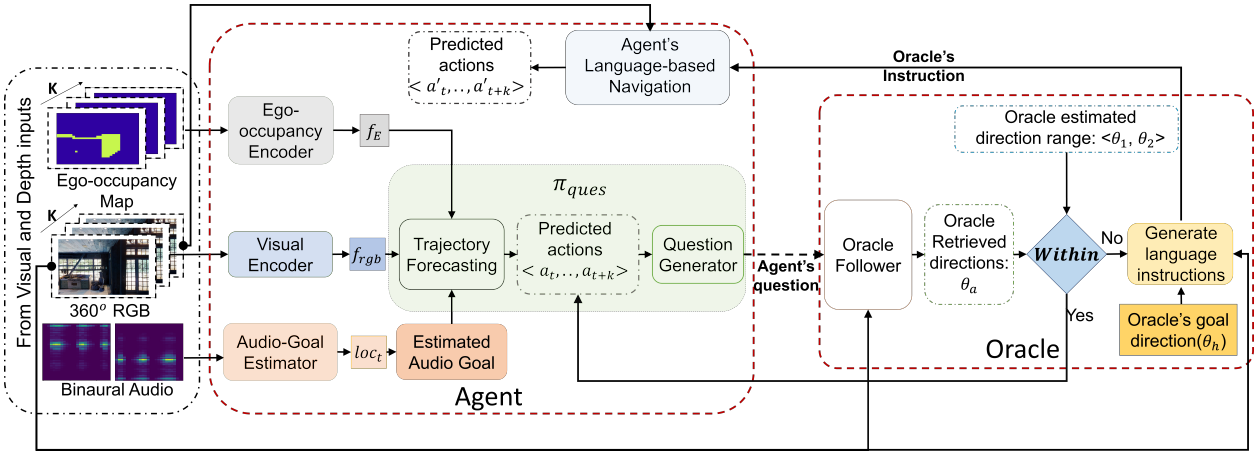

Figure 12.3: Architecture of our question policy module and the control flow within it. Here, θ_a is oracle-interpreted agent's direction to take, while θ_1 and θ_2 represent the lower and upper bounds of oracle's estimated direction range to the goal.

QuestionNet (generates natural language questions using trajectories), and (iii) *FollowerNet* (interprets the question on oracle side). These components detailed below are illustrated in Figure 12.3. They are used to enable the functionalities within the π_{ques} policy as: i) question generator (*TrajectoryNet* + *QuestionNet*), ii) question evaluator (*FollowerNet*), and iii) instruction generator (*QuestionNet*).

(i) TrajectoryNet: In order to forecast the steps of a trajectory, the agent needs to have a clear observation of its surroundings. Towards this end, we allow the agent to have a panoramic view at its current location. With the full view of its surroundings and an estimate of the audio-goal, the agent *forecasts* a sequence of l-step actions, denoted by $\mathcal{F}_a$. This is achieved by *TrajectoryNet* – a transformer encoder-decoder network which takes as input a sequence of ego occupancy maps E_t of four disjoint views (separated by 90-degrees) and the goal vector g_t predicted by a binaural audio encoder, to predict a sequence of actions $\mathcal{F}_a = \langle f_{a_1}, f_{a_2} \ldots, f_{a_l} \rangle$ (auto-regressively). The ego occupancy map is calculated by transforming depth images into point clouds and projecting them onto the ground plane.

(ii) QuestionNet: The action sequences defined in SoundSpaces [Re179] are discrete, e.g,

move_forward implies moving forward by $1m$. However, the language produced from these actions by itself may be ambiguous (since it is a higher level construct) and thus does not explicitly reflect the granularity of these discrete actions. Further, as will be explained in the Experiments section, while the trajectories are forecasted using the SoundSpaces grid (which uses 90 degree angles for turning), the language instructions are produced using a model trained on the LandmarkRxR dataset [Re187], that uses panoramic images as input. To compensate for these mismatches, we propose to first gather the view of the agent at the end of the forecasted trajectory, which we call g_{view}, and the corresponding displacement vector $g_{sub} := [d_f, cos(\theta_f), sin(\theta_f)]$, where d_f is the distance between the agent's location and the trajectory end point and θ_f is the angle difference between the direction of d_f and the agent's facing direction.

Next, we capture the panorama around the agent using 12 equiangular views, as RGB images as well as the corresponding occupancy maps to abstract the 3D scene geometry. ResNet-152 features are then extracted from these RGB images using an ImageNet pre-trained model, while the ego-occupancy maps are encoded using a 2D-CNN; both the features are fused with position embeddings and passed through a transformer encoder. In order to fuse these panoramic views with the forecasted agent views (in the SoundSpaces grid), we propose to use a transformer decoder, which takes the output of the encoder and a fusion of ResNet-152 features from g_{view}, coupled with the position encoding of g_{sub}, and the embeddings of hitherto produced words in the question (e.g., GloVe [Re188]), and proceeds to generate the next word in the question autoregressively.

(iii) FollowerNet: After the question is asked, the oracle needs to verify if it can be correctly translated into a direction that falls within the oracle's own estimation of the direction range to the goal. To this end, we incorporate *FollowerNet* at the oracle, which is assumed to have knowledge of the agent's location and its audio-visual context, and can convert the question back to the oracle's space of the view angles.

Language-based Policy Module: There can be situations when an agent cannot produce a question to ask the oracle; e.g., when there are no useful landmarks to base the question.

To cater to such cases, we equip the agent to directly query the oracle for language-based instructions. When invoked, the agent receives instructions, similar to when a wrong question is posed to the oracle.

Audio-Visual Navigation Policy Module: This policy is modeled as a transformer [Re23] based encoder-decoder as in [Re177]. The encoder takes as input the current and previous observations in the memory M, the output of which is combined with the goal descriptor g and decoded by the decoder to produce a feature vector defining the belief state of the agent b. Next, a single-layer actor-critic neural network learns a policy, π_g, that transforms this belief b to predict the distribution on the navigation actions, which the agent samples to take a step in the environment.

Selector Policy: This module, denoted π_s, decides when to navigate using audio-visual cues (i.e., use π_g), when to query the oracle for instructions directly (i.e., use π_ℓ); or when to pose a question to the oracle, (i.e., use π_{ques}). Instead of directly using model uncertainty (as is common in prior works [Re189]), we use our proposed RL framework to train this policy in an end-to-end manner, guided by the reward design ζ described below.

12.3 Experiments and Results

labelsec:expts **Datasets:** The CAVEN agent is trained and evaluated on the SoundSpaces platform [Re179]. It uses MatterPort3D environment scans [Re190]. We use the the semantic audio-visual navigation dataset from [Re179] to benchmark our experiments.

AVN-Instruct Dataset: For pre-training and evaluation of the language interaction modules (i.e., *QuestionNet*, *FollowerNet*), we use the Landmark RxR dataset [Re187], which contains 150k well-annotated sub-trajectories and their corresponding language sub-instructions grounded on scenes captured using the MatterPort3D simulator. Then we adopt the pre-trained QuestionNet to synthesize a dataset called AVN-Instruct, which contains a total of 41.5k dense pairs of sub-instructions, audio-goal, and visual scene under the state space of Soundspace simulator. Before integrating the modules into the RL framework, we fine-tune the whole question module end-to-end on AVN-Instruct with a set of 500 and 1000 samples

	Help	Heard Sound							Unheard Sound						
		SR↑	SPL↑	SNA↑	DTG↓	SWS↑	SNI↑	SNO↑	SR↑	SPL↑	SNA↑	DTG↓	SWS↑	SNI↑	SNO↑
Rand Nav.	✗	1.4	3.5	1.2	17.0	1.4	–	–	1.4	3.5	1.2	17.0	1.4	–	–
ObjGoal-RL	✗	1.5	0.8	0.6	16.7	1.1	–	–	1.5	0.8	0.6	16.7	1.1	–	–
Gan et al.	✗	29.3	23.7	23.0	11.3	14.4	–	–	15.9	12.3	11.6	12.7	8.0	–	–
Chen et al.	✗	21.6	15.1	12.1	11.2	10.7	–	–	18.0	13.4	12.9	12.9	6.9	–	–
AV-WaN	✗	20.9	16.8	16.2	10.3	8.3	–	–	17.2	13.2	12.7	11.0	6.9	–	–
SMT+Audio	✗	22.0	16.8	16.0	12.4	8.7	–	–	16.7	11.9	10.0	12.1	8.5	–	–
SAVi	✗	33.9	24.0	18.3	8.8	21.5	–	–	24.8	17.2	13.2	9.9	14.7	–	–
AVLEN	Language	36.1	24.6	19.7	8.5	23.1	–	21.8	26.2	17.6	14.2	9.2	15.8	–	15.9
AVLEN	GT-Actions	48.2	34.3	26.7	7.5	36.0	–	29.1	36.7	24.1	18.7	8.3	26.6	–	22.3
CAVEN	Noisy Language	**45.2**	**32.9**	**28.8**	**7.5**	**32.3**	17.9	**31.4**	**38.2**	**27.6**	**24.1**	**8.2**	**25.9**	15.0	**23.1**
CAVEN	Language	**48.4**	**35.8**	**31.0**	**6.9**	**34.2**	21.5	**33.4**	**42.0**	**30.0**	**26.5**	**7.6**	**30.9**	16.7	**27.9**
CAVEN	GT-Actions	**54.8**	**41.4**	**35.9**	**6.5**	**39.9**	24.3	**37.8**	**49.7**	**37.3**	**32.7**	**6.7**	**37.2**	19.8	**33.0**

Table 12.1: Comparison of CAVEN performances against the state of the art under heard and unheard sound settings. (Table from [XL10])

for validation and testing.

Evaluation Metrics: We follow the standard metrics defined in SAVi [Re177] to evaluate the navigation performance, namely SR, SPL, SNA, DTG and SWS. In addition, we introduce two new metrics for assessing navigation performance that also considers the number of language-based oracle interactions, namely: (a) success rate weighted by the inverse number of language interactions (SNI) – which is the ratio of the success rate to the average number of times either direct instructions are sought from the oracle or a question is posed to it per episodes, and (b) success rate weighted by inverse number of oracle instructions (SNO) – which is the ratio of the success rate to the average number of times either direct instructions are sought from the oracle or a *wrong* question is posed to it. These additional metrics help explain the performance gain under conversational settings.

Experimental Results and Analysis: Here, we compare our proposed formulation against state-of-the-art semantic audio-visual navigation approaches, namely [Re181, Re179, Re180,

	Help	**SR ↑**	**SPL ↑**	**SNA ↑**	**DTG ↓**	**SWS ↑**	**SNI ↑**	**SNO ↑**
Chen	✗	4.0	2.4	2.0	14.7	2.3	-	-
WaN	✗	3.0	2.0	1.8	14.0	1.6	-	-
SMTA	✗	4.2	2.9	2.1	14.9	2.8	-	-
SAVi	✗	11.8	7.4	5.0	13.1	8.4	-	-
AVLEN	Lang	14.0	8.4	5.9	12.8	11.1	-	8.5
Rand	Bi	16.9	10.6	7.9	11.9	11.1	7.2	9.4
Ufm	Bi	16.9	10.5	7.6	11.9	11.6	7.1	9.5
MU	Bi	19.6	12.4	8.9	11.4	14.0	7.8	10.2
CAVEN	Bi	**21.3**	**13.9**	**11.7**	**11.6**	**14.5**	8.4	**11.6**

Table 12.2: Comparison of CAVEN performances with different approaches in the *presence of distractor sound.* WaN: AV-WaN, STMA: STM+Audio, Rand: Random, Ufm: Uniform, MU: Model Uncertainty, Lang: Language, Bi: Bi-directional Interaction. (Table from [XL10])

Re191, Re177]. Using the same protocol as in AVLEN, we evaluate our performances on two different settings: (i) heard and (ii) unheard sound, both in unseen environments with sporadic sources. To ensure fair comparisons, we control CAVEN to have a similar number of oracle feedbacks as in [Re178]. Table 12.1 provides the results of our experiments using heard and unheard sounds. The table shows that our full model –CAVEN (language), achieves significant improvements across all metrics. CAVEN exhibits a **12% gain** on the newly introduced **SNO** metric over [Re178], our closest competitor, in both heard and unheard cases. This clearly shows that the agent benefits much more from both our novel language components. Given the budget on directly receiving instructions from the oracle, we find that CAVEN poses a correct question about 40% of the time, thereby incurring less penalty. Even with a noisy oracle, i.e., *Noisy Language* in Table 12.1, we achieve better performances compared to [Re178], showing the robustness of our framework. To induce noise, we either ground the generated oracle's instructions on random trajectories or switch 'yes'/'no' responses, both with a chance of 25%.

Param δ_{ques}	SR ↑	SPL ↑	SNA ↑	DTG ↓	SWS ↑
1.0	32.1	23.1	19.4	8.0	20.8
0.5	36.5	26.9	24.6	8.2	21.1
0.0 (ours)	**42.0**	**30.0**	**26.5**	**7.6**	**30.9**

Table 12.3: Ablation of the reward parameter δ_{ques} of CAVEN's question module under unheard sound settings. (Table from [XL10])

Navigation Under Distractor Sounds: We also evaluate the performance of CAVEN in the presence of distractor sounds, in the unheard setting. Since this environment presents a mixture of sounds, therefore to disambiguate, a one hot encoding of the target sounding object is also provided as an input to the agent . The presence of distractor sounds adversely affects the estimation of the audio-goal, which results in more uncertainty in the agent's decision-making. Under this setting, the conversations between the agent and oracle becomes even more critical. Even under such challenging circumstances, as shown in Table 12.2, we notice a 5.5% and a 3.1% **gain** on SPL and SNO, respectively against our closest competitor.

Ablation on Selector Policy: In Tables 12.1, 12.2, we compare various strategies instead of learning the selector policy, π_s. In *Random*, the agent randomly selects a navigation policy, while in *Uniform*, the agent chooses a policy every 3 steps, alternating between the three policies. In Model Uncertainty, the audio-goal uncertainty estimated by the selector policy is used to decide which policy to invoke; i.e., if the audio-goal uncertainty is above 66.7%, the language-based policy is invoked; if the uncertainty is between 33.3% and 66.7%, question policy is invoked; otherwise, the audio-goal policy is invoked. Our results show learning of π_s is better.

Analysis of Policy Dynamics: To study the situations when the agent invokes the various navigation policies, we record the confidence of audio-goal estimated by selector policy π_s, when each of the option policies is invoked and compute its distribution using all test set episodes. As shown in Figure 12.4, the audio-goal is invoked when the agent is highly

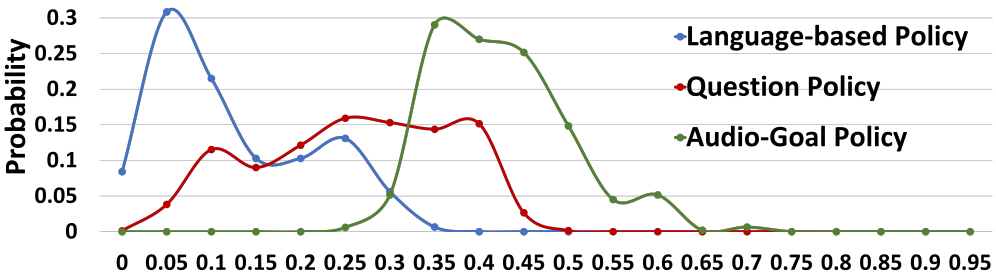

Figure 12.4: Distribution of estimated audio goal confidence when each policy is invoked. (Figure from [XL10])

confident and the language-based policy is invoked when agent's confidence is low. It is note-worthy that the question policy is invoked more often when the agent is moderately confident. Though it potentially risks being penalized by asking wrong questions, it benefits from seeking confirmation from the oracle using its own beliefs to alleviate uncertainty, thus navigating efficiently.

Insights into Differential Rewarding: In Table 12.3, we report the CAVEN performances on varying the penalty parameter δ_{ques}. Note that our differential rewarding scheme gives no penalty when the agent makes a correct question $\delta_{ques} = 0$, however penalizes heavily for mistakes. Thus, the *gap* between the two penalties act as an incentive for the agent to make more number of correct trajectory predictions than in a case where this penalty gap is lower (e.g., $\delta_{ques} = 0.5, 1.0$ in which case it is similar to the penalty it receives for the wrong question). The success rate is higher suggesting that the incentive the agent receives in making a correct question influences learning of the trajectory forecasting significantly.

Chapter 13

CONCLUSION

In this thesis, I have explored and developed throughout my PhD research two major categories of multi-modal interactive systems: Multi-modal audio generation systems and Multi-modal task solving systems. Within the domain of Multi-modal audio generation, our work and proposed methods progressed from being able to address specialized tasks, such as generation of music from musician video or rhythmic music generation, toward more general applications. Furthermore, the works progressed from music generation to wide-ranging audio generation. Chapter 3 introduced a video-to-music generation system focused on piano performances, where a direct visual-to-musical mapping exists. Chapter 4 expanded this to systems that infer rhythm from subtle human body movements. Chapter 5 generalized the approach by introducing a latent-space model, removing the need for explicit visual-audio mappings and enabling broader applications. In Chapter 6, earlier methodologies were significantly enhanced through the integration of advanced techniques such as multi-modal large language models (LLMs) and neural audio codecs, enabling generalized audio generation across diverse content types with text-based conditioning. Chapter 7 further addressed immersive audio by proposing a spatial rendering technique that captures realistic environmental acoustics, improving the realism of generated sounds.

From Chapter 8 onward, the focus shifted to Multi-modal task solving systems, where generation models were adapted for understanding-based tasks. Self-supervised learning methods were employed to align audio-visual representations in a robust way. Chapter 9 expanded the modality space by introducing language, significantly improving audio-visual alignment and enabling more challenging downstream applications. A conversational, video-based music recommendation system was developed to better capture user preferences, improving both

accuracy and interpretability. Chapter 10 presented a new benchmark to address biases in existing audio-visual question-answering datasets, alongside a model specifically designed for unbiased evaluation in complex multi-modal contexts. In Chapter 12, exploration of integrating language into embodied AI was presented through the development of a dialogue-driven audio-visual navigation agent trained with reinforcement learning to operate effectively under sparse human feedback.

Overall, this thesis presents a series of landmark contributions to the design of multi-modal systems that bridge audio, visual, and other modalities through developing deep learning techniques. The innovations presented in my works and this thesis enable the construction of robust representations for both generation and task solving systems and pave the way toward foundational AI systems capable of perceiving, reasoning, and generating realistic multi-modal signals in complex real-world environments.

XIULONG LIU'S PUBLICATIONS

[XL1] Kun Su, Xiulong Liu, and Eli Shlizerman. Audeo: Audio generation for a silent performance video. *Advances in Neural Information Processing Systems*, 33:3325–3337, 2020.

[XL2] Kun Su, Xiulong Liu, and Eli Shlizerman. How does it sound? *Advances in Neural Information Processing Systems*, 34:29258–29273, 2021.

[XL3] Xiulong Liu, Kun Su, and Eli Shlizerman. Let the beat follow you - creating interactive drum sounds from body rhythm. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 7187–7197, January 2024.

[XL4] Kun Su, Xiulong Liu, and Eli Shlizerman. Multi-instrumentalist net: Unsupervised generation of music from body movements. *arXiv preprint arXiv:2012.03478*, 2020.

[XL5] Xiulong Liu, Kun Su, and Eli Shlizerman. Tell what you hear from what you see–video to audio generation through text. *arXiv preprint arXiv:2411.05679*, 2024.

[XL6] Xiulong Liu, Ruohan Gao, Eli Shlizerman, et al. Hearing anywhere in any environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[XL7] Kun Su, Xiulong Liu, and Eli Shlizerman. From vision to audio and beyond: A unified model for audio-visual representation and generation. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine*

Learning, volume 235 of *Proceedings of Machine Learning Research*, pages 46804–46822. PMLR, 21–27 Jul 2024.

[XL8] Zhikang Dong, Xiulong Liu, Bin Chen, Pawel Polak, and Peng Zhang. Musechat: A conversational music recommendation system for videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12775–12785, June 2024.

[XL9] Xiulong Liu, Zhikang Dong, and Peng Zhang. Tackling data bias in music-avqa: Crafting a balanced dataset for unbiased question-answering. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4478–4487, 2024.

[XL10] Xiulong Liu, Sudipta Paul, Moitreya Chatterjee, and Anoop Cherian. Caven: An embodied conversational agent for efficient audio-visual navigation in noisy environments. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(4):3765–3773, Mar. 2024.

[XL11] Kun Su, Xiulong Liu, and Eli Shlizerman. Predict & cluster: Unsupervised skeleton based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9631–9640, 2020.

BIBLIOGRAPHY

[Re1] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):172–186, 2019.

[Re2] Yuan Gong, Yu-An Chung, and James Glass. Ast: Audio spectrogram transformer. *arXiv preprint arXiv:2104.01778*, 2021.

[Re3] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[Re4] Sagnik Majumder, Changan Chen, Ziad Al-Halah, and Kristen Grauman. Few-shot audio-visual learning of environment acoustics. *Advances in Neural Information Processing Systems*, 35:2522–2536, 2022.

[Re5] Mason Long Wang, Ryosuke Sawata, Samuel Clarke, Ruohan Gao, Shangzhe Wu, and Jiajun Wu. Hearing anything anywhere. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11790–11799, 2024.

[Re6] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

[Re7] Pierre Jean A Colombo, Chloé Clavel, and Pablo Piantanida. Infolm: A new metric to evaluate summarization & data2text generation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 10554–10562, 2022.

[Re8] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr, 2013.

[Re9] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315, 2017.

[Re10] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2021.

[Re11] Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438*, 2022.

[Re12] Robert A Moog. Midi: Musical instrument digital interface. *Journal of the Audio Engineering Society*, 34(5):394–404, 1986.

[Re13] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer: Generating music with long-term structure. In *International Conference on Learning Representations*, 2018.

[Re14] Yu-Siang Huang and Yi-Hsuan Yang. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1180–1188, 2020.

[Re15] Jon Gillick, Adam Roberts, Jesse Engel, Douglas Eck, and David Bamman. Learning to groove with inverse sequence transformations. In *International Conference on Machine Learning (ICML)*, 2019.

[Re16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[Re17] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[Re18] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[Re19] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 203–213, 2020.

[Re20] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, pages 813–824, 2021.

[Re21] Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[Re22] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[Re23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[Re24] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model. *arXiv preprint arXiv:1612.07837*, 2016.

[Re25] Xinhao Mei, Varun Nagaraja, Gael Le Lan, Zhaoheng Ni, Ernie Chang, Yangyang Shi, and Vikas Chandra. Foleygen: Visually-guided audio generation. *arXiv preprint arXiv:2309.10537*, 2023.

[Re26] Hugo Flores Garcia, Prem Seetharaman, Rithesh Kumar, and Bryan Pardo. Vampnet: Music generation via masked acoustic token modeling. *arXiv preprint arXiv:2307.04686*, 2023.

[Re27] Zalán Borsos, Matt Sharifi, Damien Vincent, Eugene Kharitonov, Neil Zeghidour, and Marco Tagliasacchi. Soundstorm: Efficient parallel audio generation. *arXiv preprint arXiv:2305.09636*, 2023.

[Re28] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[Re29] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. Gansynth: Adversarial neural audio synthesis. *arXiv preprint arXiv:1902.08710*, 2019.

[Re30] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, 2019.

[Re31] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. *arXiv preprint arXiv:1904.09324*, 2019.

[Re32] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11315–11325, 2022.

[Re33] Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T Freeman, Michael Rubinstein, et al. Muse: Text-to-image generation via masked generative transformers. *arXiv preprint arXiv:2301.00704*, 2023.

[Re34] Alon Ziv, Itai Gat, Gael Le Lan, Tal Remez, Felix Kreuk, Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. Masked audio generation using a single non-autoregressive transformer. *arXiv preprint arXiv:2401.04577*, 2024.

[Re35] Chuang Gan, Deng Huang, Peihao Chen, Joshua B Tenenbaum, and Antonio Torralba. Foley music: Learning to generate music from videos. *ECCV*, 2020.

[Re36] Ye Zhu, Yu Wu, Kyle Olszewski, Jian Ren, Sergey Tulyakov, and Yan Yan. Discrete contrastive diffusion for cross-modal music and image generation. In *International Conference on Learning Representations (ICLR)*, 2023.

[Re37] Andrew Owens, Phillip Isola, Josh McDermott, Antonio Torralba, Edward H. Adelson, and William T. Freeman. Visually indicated sounds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[Re38] Yipin Zhou, Zhaowen Wang, Chen Fang, Trung Bui, and Tamara L Berg. Visual to sound: Generating natural sound for videos in the wild. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[Re39] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[Re40] Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. High-fidelity audio compression with improved rvqgan. *Advances in Neural Information Processing Systems*, 36, 2024.

[Re41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[Re42] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[Re43] Vladimir Iashin and Esa Rahtu. Taming visually guided sound generation. *arXiv preprint arXiv:2110.08791*, 2021.

[Re44] Roy Sheffer and Yossi Adi. I hear your true colors: Image guided audio generation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[Re45] Kun Su, Judith Yue Li, Qingqing Huang, Dima Kuzmin, Joonseok Lee, Chris Donahue, Fei Sha, Aren Jansen, Yu Wang, Mauro Verzetti, et al. V2meow: Meowing to the visual beat via music generation. *arXiv preprint arXiv:2305.06594*, 2023.

[Re46] Simian Luo, Chuanhao Yan, Chenxu Hu, and Hang Zhao. Diff-foley: Synchronized video-to-audio synthesis with latent diffusion models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 48855–48876. Curran Associates, Inc., 2023.

[Re47] Heng Wang, Jianbo Ma, Santiago Pascual, Richard Cartwright, and Weidong Cai. V2a-mapper: A lightweight solution for vision-to-audio generation by connecting foundation models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 15492–15501, 2024.

[Re48] Shentong Mo, Jing Shi, and Yapeng Tian. Diffava: Personalized text-to-audio generation with visual alignment. *arXiv preprint arXiv:2305.12903*, 2023.

[Re49] Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. In *International Conference on Machine Learning*, pages 13916–13932. PMLR, 2023.

[Re50] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. AudioLDM: Text-to-audio generation with latent diffusion models. *Proceedings of the International Conference on Machine Learning*, 2023.

[Re51] Haohe Liu, Qiao Tian, Yi Yuan, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Yuping Wang, Wenwu Wang, Yuxuan Wang, and Mark D Plumbley. Audioldm 2:

Learning holistic audio generation with self-supervised pretraining. *arXiv preprint arXiv:2308.05734*, 2023.

[Re52] Hao-Wen Dong, Xiaoyu Liu, Jordi Pons, Gautam Bhattacharya, Santiago Pascual, Joan Serrà, Taylor Berg-Kirkpatrick, and Julian McAuley. Clipsonic: Text-to-audio synthesis with unlabeled videos and pretrained language-vision models. In *2023 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 1–5. IEEE, 2023.

[Re53] Qingqing Huang, Daniel S Park, Tao Wang, Timo I Denk, Andy Ly, Nanxin Chen, Zhengdong Zhang, Zhishuai Zhang, Jiahui Yu, Christian Frank, et al. Noise2music: Text-conditioned music generation with diffusion models. *arXiv preprint arXiv:2302.03917*, 2023.

[Re54] Flavio Schneider, Ojasv Kamal, Zhijing Jin, and Bernhard Schölkopf. Mo\ˆ usai: Text-to-music generation with long-context latent diffusion. *arXiv preprint arXiv:2301.11757*, 2023.

[Re55] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn 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.

[Re56] Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. Audiogen: Textually guided audio generation. *arXiv preprint arXiv:2209.15352*, 2022.

[Re57] Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation. *Advances in Neural Information Processing Systems*, 36, 2024.

[Re58] Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, et al. Audiolm: a language modeling approach to audio generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[Re59] Andrea Agostinelli, Timo I Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, et al. Musiclm: Generating music from text. *arXiv preprint arXiv:2301.11325*, 2023.

[Re60] Nikhil Singh, Jeff Mentch, Jerry Ng, Matthew Beveridge, and Iddo Drori. Image2reverb: Cross-modal reverb impulse response synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 286–295, 2021.

[Re61] Anton Ratnarajah, Shi-Xiong Zhang, Meng Yu, Zhenyu Tang, Dinesh Manocha, and Dong Yu. Fast-rir: Fast neural diffuse room impulse response generator. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 571–575. IEEE, 2022.

[Re62] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

[Re63] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[Re64] Kun Su, Mingfei Chen, and Eli Shlizerman. Inras: Implicit neural representation for audio scenes. *Advances in Neural Information Processing Systems*, 35:8144–8158, 2022.

[Re65] Andrew Luo, Yilun Du, Michael Tarr, Josh Tenenbaum, Antonio Torralba, and Chuang Gan. Learning neural acoustic fields. *Advances in Neural Information Processing Systems*, 35:3165–3177, 2022.

[Re66] Anton Ratnarajah, Zhenyu Tang, Rohith Aralikatti, and Dinesh Manocha. Mesh2ir: Neural acoustic impulse response generator for complex 3d scenes. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 924–933, 2022.

[Re67] Alexander Richard, Peter Dodds, and Vamsi Krishna Ithapu. Deep impulse responses: Estimating and parameterizing filters with deep networks. pages 3209–3213. IEEE, 2022.

[Re68] Susan Liang, Chao Huang, Yapeng Tian, Anurag Kumar, and Chenliang Xu. Av-nerf: Learning neural fields for real-world audio-visual scene synthesis. *Advances in Neural Information Processing Systems*, 36:37472–37490, 2023.

[Re69] Anton Ratnarajah, Sreyan Ghosh, Sonal Kumar, Purva Chiniya, and Dinesh Manocha. Av-rir: Audio-visual room impulse response estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27164–27175, 2024.

[Re70] Swapnil Bhosale, Haosen Yang, Diptesh Kanojia, Jiankang Deng, and Xiatian Zhu. Av-gs: Learning material and geometry aware priors for novel view acoustic synthesis. *arXiv preprint arXiv:2406.08920*, 2024.

[Re71] Ziyang Chen, Israel D Gebru, Christian Richardt, Anurag Kumar, William Laney, Andrew Owens, and Alexander Richard. Real acoustic fields: An audio-visual room acoustics dataset and benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21886–21896, 2024.

[Re72] Byeongjoo Ahn, Karren Yang, Brian Hamilton, Jonathan Sheaffer, Anurag Ranjan, Miguel Sarabia, Oncel Tuzel, and Jen-Hao Rick Chang. Novel-view acoustic synthesis from 3d reconstructed rooms. *arXiv preprint arXiv:2310.15130*, 2023.

[Re73] Mingfei Chen and Eli Shlizerman. AV-cloud: Spatial audio rendering through audio-visual cloud splatting. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[Re74] Zitong Lan, Chenhao Zheng, Zhiwei Zheng, and Mingmin Zhao. Acoustic volume rendering for neural impulse response fields. *arXiv preprint arXiv:2411.06307*, 2024.

[Re75] Rishabh Garg, Ruohan Gao, and Kristen Grauman. Geometry-aware multi-task learning for binaural audio generation from video. In *British Machine Vision Conference (BMVC)*, 2021.

[Re76] Pedro Morgado, Nuno Nvasconcelos, Timothy Langlois, and Oliver Wang. Self-supervised generation of spatial audio for 360 video. *Advances in neural information processing systems*, 2018.

[Re77] Rishabh Garg, Ruohan Gao, and Kristen Grauman. Visually-guided audio spatialization in video with geometry-aware multi-task learning. In *International Journal of Computer Vision (IJCV)*, 2023.

[Re78] Ruohan Gao and Kristen Grauman. 2.5d visual sound. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[Re79] Dingzeyu Li, Timothy R Langlois, and Changxi Zheng. Scene-aware audio for 360 videos. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018.

[Re80] Jesper Haahr Christensen, Sascha Hornauer, and X Yu Stella. Batvision: Learning to see 3d spatial layout with two ears. In *ICRA*. IEEE, 2020.

[Re81] Ruohan Gao, Changan Chen, Ziad Al-Halah, Carl Schissler, and Kristen Grauman. Visualechoes: Spatial visual representation learning through echolocation. In *European Conference on Computer Vision (ECCV)*, 2020.

[Re82] Ziyang Chen, Xixi Hu, and Andrew Owens. Structure from silence: Learning scene structure from ambient sound. *Conference on Robot Learning (CoRL)*, 2021.

[Re83] Mason Wang, Samuel Clarke, Jui-Hsien Wang, Ruohan Gao, and Jiajun Wu. Soundcam: a dataset for finding humans using room acoustics. *Advances in Neural Information Processing Systems*, 36, 2024.

[Re84] Changan Chen, Ruohan Gao, Paul Calamia, and Kristen Grauman. Visual acoustic matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18858–18868, 2022.

[Re85] Tingle Li, Renhao Wang, Po-Yao Huang, Andrew Owens, and Gopala Anumanchipalli. Self-supervised audio-visual soundscape stylization. In *ECCV*, 2024.

[Re86] Changan Chen, Alexander Richard, Roman Shapovalov, Vamsi Krishna Ithapu, Natalia Neverova, Kristen Grauman, and Andrea Vedaldi. Novel-view acoustic synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6409–6419, 2023.

[Re87] Arjun Somayazulu, Changan Chen, and Kristen Grauman. Self-supervised visual acoustic matching. *Advances in Neural Information Processing Systems*, 36, 2024.

[Re88] Sanjoy Chowdhury, Sreyan Ghosh, Dasgupta Subhrajyoti, Anton Ratnarajah, Utkarsh Tyagi, and Dinesh Manocha. Adverb: Visually guided audio dereverberation. *ICCV*, 2023.

[Re89] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[Re90] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

[Re91] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023.

[Re92] Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding. *arXiv preprint arXiv:2306.02858*, 2023.

[Re93] Yuan Gong, Hongyin Luo, Alexander H Liu, Leonid Karlinsky, and James Glass. Listen, think, and understand. *arXiv preprint arXiv:2305.10790*, 2023.

[Re94] Soham Deshmukh, Benjamin Elizalde, Rita Singh, and Huaming Wang. Pengi: An audio language model for audio tasks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 18090–18108. Curran Associates, Inc., 2023.

[Re95] Sanjoy Chowdhury, Sayan Nag, Subhrajyoti Dasgupta, Jun Chen, Mohamed Elhoseiny, Ruohan Gao, and Dinesh Manocha. Meerkat: Audio-visual large language model for grounding in space and time. *arXiv preprint arXiv:2407.01851*, 2024.

[Re96] Guangyao Li, Yake Wei, Yapeng Tian, Chenliang Xu, Ji-Rong Wen, and Di Hu. Learning to answer questions in dynamic audio-visual scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19108–19118, 2022.

[Re97] Gengze Zhou, Yicong Hong, and Qi Wu. Navgpt: Explicit reasoning in vision-and-language navigation with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 7641–7649, 2024.

[Re98] Zineng Tang, Ziyi Yang, Chenguang Zhu, Michael Zeng, and Mohit Bansal. Any-to-any generation via composable diffusion. *Advances in Neural Information Processing Systems*, 36, 2024.

[Re99] Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua. Next-gpt: Any-to-any multimodal llm. *arXiv preprint arXiv:2309.05519*, 2023.

[Re100] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and frames: Dual-objective piano transcription. *arXiv preprint arXiv:1710.11153*, 2017.

[Re101] Zheng Yan, Weiwei Liu, Shiping Wen, and Yin Yang. Multi-label image classification by feature attention network. *IEEE Access*, 7:98005–98013, 2019.

[Re102] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[Re103] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[Re104] Bryan Wang and Yi-Hsuan Yang. Performancenet: Score-to-audio music generation with multi-band convolutional residual network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1174–1181, 2019.

[Re105] Daniel Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984.

[Re106] Mert Bay, Andreas F Ehmann, and J Stephen Downie. Evaluation of multiple-f0 estimation and tracking systems. In *ISMIR*, pages 315–320, 2009.

[Re107] A Sophia Koepke, Olivia Wiles, Yael Moses, and Andrew Zisserman. Sight to sound: An end-to-end approach for visual piano transcription. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1838–1842. IEEE, 2020.

[Re108] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. *arXiv preprint arXiv:1801.07455*, 2018.

[Re109] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[Re110] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

[Re111] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Counting out time: Class agnostic video repetition counting in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10387–10396, 2020.

[Re112] Florian Krebs, Sebastian Böck, and Gerhard Widmer. An efficient state-space model for joint tempo and meter tracking. In *ISMIR*, pages 72–78, 2015.

[Re113] Peter Grosche, Meinard Müller, and Frank Kurth. Cyclic tempogram—a mid-level tempo representation for musicsignals. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5522–5525. IEEE, 2010.

[Re114] Abe Davis and Maneesh Agrawala. Visual rhythm and beat. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2532–2535, 2018.

[Re115] Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

[Re116] Fabrizio Pedersoli and Masataka Goto. Dance beat tracking from visual information alone. *ISMIR*, 2020.

[Re117] Shuhei Tsuchida, Satoru Fukayama, Masahiro Hamasaki, and Masataka Goto. Aist dance video database: Multi-genre, multi-dancer, and multi-camera database for dance information processing. In *ISMIR*, pages 501–510, 2019.

[Re118] Colin Raffel. *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. PhD thesis, Columbia University, 2016.

[Re119] Matthew EP Davies, Norberto Degara, and Mark D Plumbley. Evaluation methods for musical audio beat tracking algorithms. *Queen Mary University of London, Centre for Digital Music, Tech. Rep. C4DM-TR-09-06*, 2009.

[Re120] Eitan Richardson and Yair Weiss. On gans and gmms. In *Advances in Neural Information Processing Systems*, pages 5847–5858, 2018.

[Re121] Li-Chia Yang and Alexander Lerch. On the evaluation of generative models in music. *Neural Computing and Applications*, 32(9):4773–4784, 2020.

[Re122] Ruilong Li, Shan Yang, David A Ross, and Angjoo Kanazawa. Ai choreographer: Music conditioned 3d dance generation with aist++. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 13401–13412, 2021.

[Re123] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[Re124] Bochen Li, Xinzhao Liu, Karthik Dinesh, Zhiyao Duan, and Gaurav Sharma. Creating a multitrack classical music performance dataset for multimodal music analysis: Challenges, insights, and applications. *IEEE Transactions on Multimedia*, 21(2):522–535, 2018.

[Re125] Juan F Montesinos, Olga Slizovskaia, and Gloria Haro. Solos: A dataset for audio-visual music analysis. *arXiv preprint arXiv:2006.07931*, 2020.

[Re126] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.

[Re127] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[Re128] Wouter Kool, Herke Van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International Conference on Machine Learning*, pages 3499–3508. PMLR, 2019.

[Re129] Khaled Koutini, Jan Schlüter, Hamid Eghbal-Zadeh, and Gerhard Widmer. Efficient training of audio transformers with patchout. *arXiv preprint arXiv:2110.05069*, 2021.

[Re130] Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fr\'echet audio distance: A metric for evaluating music enhancement algorithms. *arXiv preprint arXiv:1812.08466*, 2018.

[Re131] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[Re132] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[Re133] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575, 2015.

[Re134] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[Re135] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[Re136] D. Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984.

[Re137] Manfred R Schroeder. New method of measuring reverberation time. *The Journal of the Acoustical Society of America*, 37(6_Supplement):1187–1188, 1965.

[Re138] Yuan Gong, Andrew Rouditchenko, Alexander H Liu, David Harwath, Leonid Karlinsky, Hilde Kuehne, and James Glass. Contrastive audio-visual masked autoencoder. *arXiv preprint arXiv:2210.07839*, 2022.

[Re139] Po-Yao Huang, Vasu Sharma, Hu Xu, Chaitanya Ryali, Haoqi Fan, Yanghao Li, Shang-Wen Li, Gargi Ghosh, Jitendra Malik, and Christoph Feichtenhofer. Mavil: Masked audio-video learners. *arXiv preprint arXiv:2212.08071*, 2022.

[Re140] Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation. *arXiv preprint arXiv:2306.05284*, 2023.

[Re141] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

[Re142] Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. High-fidelity audio compression with improved rvqgan. *arXiv preprint arXiv:2306.06546*, 2023.

[Re143] Hangbo Bao, Wenhui Wang, Li Dong, Qiang Liu, Owais Khan Mohammed, Kriti Aggarwal, Subhojit Som, Songhao Piao, and Furu Wei. Vlmo: Unified vision-language pre-training with mixture-of-modality-experts. *Advances in Neural Information Processing Systems*, 35:32897–32912, 2022.

[Re144] Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, et al. Image as a foreign language: Beit pretraining for all vision and vision-language tasks. *arXiv preprint arXiv:2208.10442*, 2022.

[Re145] Tianhong Li, Huiwen Chang, Shlok Mishra, Han Zhang, Dina Katabi, and Dilip Krishnan. Mage: Masked generative encoder to unify representation learning and image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2142–2152, 2023.

[Re146] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[Re147] Bin Zhu, Bin Lin, Munan Ning, Yang Yan, Jiaxi Cui, HongFa Wang, Yatian Pang, Wenhao Jiang, Junwu Zhang, Zongwei Li, et al. Languagebind: Extending video-language pretraining to n-modality by language-based semantic alignment. *arXiv preprint arXiv:2310.01852*, 2023.

[Re148] Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. Imagebind: One embedding space to bind them all. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15180–15190, 2023.

[Re149] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5288–5296, 2016.

[Re150] Weiyao Wang, Du Tran, and Matt Feiszli. What makes training multi-modal classification networks hard? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12695–12705, 2020.

[Re151] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR, 2021.

[Re152] Haytham M Fayek and Anurag Kumar. Large scale audiovisual learning of sounds with weakly labeled data. *arXiv preprint arXiv:2006.01595*, 2020.

[Re153] Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. Attention bottlenecks for multimodal fusion. *Advances in Neural Information Processing Systems*, 34:14200–14213, 2021.

[Re154] Karol J Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1015–1018, 2015.

[Re155] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[Re156] Yuan Gong, Cheng-I Lai, Yu-An Chung, and James Glass. Ssast: Self-supervised audio spectrogram transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10699–10709, 2022.

[Re157] Po-Yao Huang, Hu Xu, Juncheng Li, Alexei Baevski, Michael Auli, Wojciech Galuba, Florian Metze, and Christoph Feichtenhofer. Masked autoencoders that listen. *Advances in Neural Information Processing Systems*, 35:28708–28720, 2022.

[Re158] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 776–794. Springer, 2020.

[Re159] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[Re160] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.

[Re161] Sungeun Hong, Woobin Im, and Hyun S Yang. Cbvmr: content-based video-music retrieval using soft intra-modal structure constraint. In *Proceedings of the 2018 ACM on international conference on multimedia retrieval*, pages 353–361, 2018.

[Re162] Dídac Surís, Carl Vondrick, Bryan Russell, and Justin Salamon. It's time for artistic correspondence in music and video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10564–10574, 2022.

[Re163] Daniel McKee, Justin Salamon, Josef Sivic, and Bryan Russell. Language-guided music recommendation for video via prompt analogies. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14784–14793, 2023.

[Re164] Jongpil Lee, Nicholas J Bryan, Justin Salamon, Zeyu Jin, and Juhan Nam. Disentangled multidimensional metric learning for music similarity. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6–10. IEEE, 2020.

[Re165] Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Dhruv Batra, and Devi Parikh. Vqa: Visual question answering. 2016.

[Re166] Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Yin and yang: Balancing and answering binary visual questions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5014–5022, 2016.

[Re167] Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. Don't just assume; look and answer: Overcoming priors for visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4971–4980, 2018.

[Re168] Remi Cadene, Corentin Dancette, Matthieu Cord, Devi Parikh, et al. Rubi: Reducing unimodal biases for visual question answering. *Advances in neural information processing systems*, 32, 2019.

[Re169] Gouthaman Kv and Anurag Mittal. Reducing language biases in visual question answering with visually-grounded question encoder. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*, pages 18–34. Springer, 2020.

[Re170] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

[Re171] Yan-Bo Lin, Yi-Lin Sung, Jie Lei, Mohit Bansal, and Gedas Bertasius. Vision transformers are parameter-efficient audio-visual learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2299–2309, 2023.

[Re172] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12009–12019, 2022.

[Re173] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*, 2023.

[Re174] Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, et al. Conversational ai: The science behind the alexa prize. *arXiv preprint arXiv:1801.03604*, 2018.

[Re175] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[Re176] Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14953–14962, 2023.

[Re177] Changan Chen et al. Semantic audio-visual navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15516–15525, 2021.

[Re178] Sudipta Paul et al. Avlen: Audio-visual-language embodied navigation in 3d environments. *arXiv preprint arXiv:2210.07940*, 2022.

[Re179] Changan Chen, Unnat Jain, Carl Schissler, Sebastia Vicenc Amengual Gari, Ziad Al-Halah, Vamsi Krishna Ithapu, Philip Robinson, and Kristen Grauman. Soundspaces: Audio-visual navigation in 3d environments. In *European Conference on Computer Vision*, pages 17–36. Springer, 2020.

[Re180] Changan Chen et al. Learning to set waypoints for audio-visual navigation. In *International Conference on Learning Representations*, 2021.

[Re181] Chuang Gan, Yiwei Zhang, Jiajun Wu, Boqing Gong, and Joshua B Tenenbaum. Look, listen, and act: Towards audio-visual embodied navigation. In *2020 IEEE*

International Conference on Robotics and Automation (ICRA), pages 9701–9707. IEEE, 2020.

[Re182] Santosh Kesiraju, Oldřich Plchot, Lukáš Burget, and Suryakanth V Gangashetty. Learning document embeddings along with their uncertainties. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2319–2332, 2020.

[Re183] Aditya Siddhant and Zachary C Lipton. Deep bayesian active learning for natural language processing: Results of a large-scale empirical study. *arXiv preprint arXiv:1808.05697*, 2018.

[Re184] Yijun Xiao and William Yang Wang. Quantifying uncertainties in natural language processing tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7322–7329, 2019.

[Re185] Tuyen P Le et al. A deep hierarchical reinforcement learning algorithm in partially observable markov decision processes. *Ieee Access*, 6:49089–49102, 2018.

[Re186] Richard S Sutton et al. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[Re187] Keji He et al. Landmark-rxr: Solving vision-and-language navigation with fine-grained alignment supervision. In M. Ranzato and etal, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 652–663. Curran Associates, Inc., 2021.

[Re188] Jeffrey Pennington et al. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[Re189] Ta-Chung Chi, Minmin Shen, Mihail Eric, Seokhwan Kim, and Dilek Hakkani-tur. Just ask: An interactive learning framework for vision and language navigation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2459–2466, 2020.

[Re190] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017.

[Re191] Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. Scene memory transformer for embodied agents in long-horizon tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 538–547, 2019.