UNIVERSITY OF CALIFORNIA SAN DIEGO

Generative AI for Music and Audio

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Hao-Wen Dong

Committee in charge:

Professor Taylor Berg-Kirkpatrick, Co-Chair
Professor Julian McAuley, Co-Chair
Professor Shlomo Dubnov
Professor Lawrence Saul

2024

Copyright
Hao-Wen Dong, 2024
All rights reserved.

The Dissertation of Hao-Wen Dong is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2024

DEDICATION

To my wife Eltha, my parents Nancy and David, and my pooh bear,
without whom nothing would have been possible.

EPIGRAPH

Without music, life would be a mistake.

—Friedrich Nietzsche

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost I want to express my deepest gratitude to my advisors Julian McAuley and Taylor Berg-Kirkpatrick for their guidance and support throughout my PhD years. While my PhD topic is not in their core fields of expertise, Julian and Taylor have always been providing me insightful and constructive advice on my research. I am fortunate to have such amazing advisors who trust and push me to pursue my own research agenda, and give me much freedom on doing internships elsewhere frequently. For the past few years, Julian and Taylor have greatly shaped my attitude towards research and life, which I will carry on to my future endeavors. In particular, I want to thank Julian for providing me invaluable advice on my career and job search, and I want to thank Taylor for guiding me how to think about and approach my research.

I would also like to thank Shlomo Dubnov for serving on my doctoral committee and for his thoughtful advice and genuine support on my research and career. I also thank Lawrence Saul for serving on my doctoral committee and revamping my writing skills in his amazing Scientific Teaching course. I also want to thank Yi-Hsuan Yang for bringing me into this field and for his continued guidance along my career.

I want to extend my sincere gratitude to my wife, my parents, my pooh bear and my family, without whom nothing would have been possible. My warmest thanks to my wife Eltha Teng for her love, support and companionship. It is almost like a dream to have done our PhD together at UC San Diego, and I am fortunate to have gone through all the ups and downs with you. It is you that make this journey joyful and memorable. My heartfelt thanks to my parents Nancy Yen and David Dong for their unconditional love and support that have shaped me into the person I am today. My sweetest thanks to my pooh bear for being my best buddy for 26 years and accompanying me through all the deadlines.

I would also like to thank all my colleagues and friends I met at UC San Diego. Huge thanks to Ke Chen for collaborating on several projects and being my go-to person whenever I want to discuss some new ideas. Big thanks to Zachary Novack for co-organizing the AI Music reading group and collaborating on several projects. Special thanks to Chris Donahue for introducing me to Julian and Taylor when I just started my PhD. Thanks to Weihan Xu,

Haven Kim, Sanjayan Sreekala, Chris Francis, Sachinda Edirisooriya, Wanning Lu, Hoang Phan, Diego Reyes, Phillip Long, Junda Wu and Zhouhang Xie for collaborating with me on several projects. Thanks to Zhankui He and Noveen Sachdeva for co-maintaining our lab machines and fixing them in the noisy server room at SDSC once in a while. Thanks to my labmates in McAuley Lab and Berg Lab for their helpful feedback on my work and all the fun we had. Thank you to my friends Yueh-Hua Wu, I-Lin Yeh, Chun-Jhen Lai, Kai-En Lin, Chih-Chun Hsu, I-Da Chiang and Fang-Chi Leong (just to name a few) for enriching my life outside of research.

Part of this dissertation was done during internships at NVIDIA, Adobe, Dolby, Amazon and Sony. I want to express my sincere gratitude to my internship mentors (in chronological order): Cong Zhou, Naoya Takahashi, Wenbo Zhao, Gunnar Sigurdsson, Xiaoyu Liu, Justin Salamon, Oriol Nieto, Siddharth Gururani and Ming-Yu Liu for their guidance and support during my internships as well as the time and effort they devoted in my project. I would also like to thank all my colleagues and friends I met during my internships. Special thanks to Julia Wilkins, Jordi Pons and Chenyang Tao for collaborating and offering useful advice on my projects. Thanks to Junghyun Koo, Tung-Cheng Wu, Vincent Cheung, Yun-Hsuan Chen, Siddharth Nijhawan, Kin Wai Cheuk, Ilaria Manco, Jiawen Huang and Marco Martínez for all the fun we had in Tokyo during my internship at Sony.

Thank you to all my collaborators (in alphabetical order): Taylor Berg-Kirkpatrick, Ke Chen, Tagyoung Chung, Chris Donahue, Shlomo Dubnov, Sachinda Edirisooriya, Chris Francis, Satoru Fukayama, Benjamin Genchel, Arpit Gupta, Siddharth Gururani, Jing Huang, Dasaem Jeong, Jiun-Yu Kao, Haven Kim, Tetsuro Kitahara, Yu-Hsiang Lin, Hao-Min Liu, Xiaoyu Liu, Ming-Yu Liu, Phillip Long, Julian McAuley, Yuki Mitsufuji, Anjali Narayan-Chen, Oriol Nieto, Zachary Novack, Nanyun Peng, Jordi Pons, Keijiro Saino, Justin Salamon, Joan Serrà, Gunnar Sigurdsson, Sanjayan Sreekala, Naoya Takahashi, Chih-Pin Tan, Chenyang Tao, Julia Wilkins, Junda Wu, Wen-Yi Xiao, Zhouhang Xie, Weihan Xu, Li-Chia Yang, Yi-Hsuan Yang, Yin-Cheng Yeh, Wenbo Zhao and Cong Zhou. It has been wonderful working with you, and I look forward to collaborating with you again in the future.

Finally, I want to thank UCSD ECE Department, J. Yang and Family Foundation, and Taiwan Ministry of Education for their generous support for my PhD study.

* * *

This dissertation contains material from the following publications:

- *Chapter 2, in full, is a reprint of the material as it appears in "MusPy: A Toolkit for Symbolic Music Generation" by Hao-Wen Dong, Ke Chen, Julian McAuley and Taylor Berg-Kirkpatrick, which was published in the Proceedings of the International Society for Music Information Retrieval Conference (ISMIR) in 2020. The dissertation author was the primary investigator and author of this paper.*
- *Chapter 3, in full, is a reprint of the material as it appears in "Multitrack Music Transformer" by Hao-Wen Dong, Ke Chen, Shlomo Dubnov, Julian McAuley and Taylor Berg-Kirkpatrick, which was published in the Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) in 2023. The dissertation author was the primary investigator and author of this paper.*
- *Chapter 4, in full, is a reprint of the material as it appears in "Towards Automatic Instrumentation by Learning to Separate Parts in Symbolic Multitrack Music" by Hao-Wen Dong, Chris Donahue, Taylor Berg-Kirkpatrick and Julian McAuley, which was published in the Proceedings of the International Society for Music Information Retrieval Conference (ISMIR) in 2021. The dissertation author was the primary investigator and author of this paper.*
- *Chapter 5, in full, is a reprint of the material as it appears in "Deep Performer: Score-to-Audio Music Performance Synthesis" by Hao-Wen Dong, Cong Zhou, Taylor Berg-Kirkpatrick and Julian McAuley, which was published in the Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) in 2022. The dissertation author was the primary investigator and author of this paper.*
- *Chapter 6, in full, is a reprint of the material as it appears in "CLIPSep: Learning Text-queried Sound Separation with Noisy Unlabeled Videos" by Hao-Wen Dong, Naoya Takahashi, Yuki Mitsufuji, Julian McAuley and Taylor Berg-Kirkpatrick, which was published in the Proceedings of the International Conference on Learning Representations (ICLR) in 2023. The dissertation author was the primary investigator and author of this paper.*

- *Chapter 7, in full, is a reprint of the material as it appears in "CLIPSonic: Text-to-Audio Synthesis with Unlabeled Videos and Pretrained Language-Vision Models" by Hao-Wen Dong, Xiaoyu Liu, Jordi Pons, Gautam Bhattacharya, Santiago Pascual, Joan Serrà, Taylor Berg-Kirkpatrick and Julian McAuley, which was published in the Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA) in 2023. The dissertation author was the primary investigator and author of this paper.*

VITA

Education

2019 – 2024 *Doctor of Philosophy in Computer Science*, University of California San Diego

2019 – 2021 *Master of Science in Computer Science*, University of California San Diego

2013 – 2017 *Bachelor of Science in Electrical Engineering*, National Taiwan University

Professional Experience

Sep 2023 – Dec 2023 *Research Intern*, NVIDIA

May 2023 – Sep 2023 *Research Scientist/Engineer Intern*, Adobe

Jan 2023 – Apr 2023 *Speech/Audio Deep Learning Intern*, Dolby

Sep 2022 – Jan 2023 *Applied Scientist Intern*, Amazon

May 2022 – Sep 2022 *Student Intern*, Sony

Jun 2021 – Sep 2021 *Deep Learning Audio Intern*, Dolby

May 2019 – Aug 2019 *Research Intern*, Yamaha

Jul 2017 – Apr 2019 *Research Assistant*, Academia Sinica

Publications

Weihan Xu, Julian McAuley, Shlomo Dubnov, and Hao-Wen Dong, "Equipping Pretrained Unconditional Music Transformers with Instrument and Genre Controls," *IEEE Big Data Workshop on AI Music Generation (AIMG)*, 2023.

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," *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2023.

Hao-Wen Dong, Gunnar A. Sigurdsson, Chenyang Tao, Jiun-Yu Kao, Yu-Hsiang Lin, Anjali Narayan-Chen, Arpit Gupta, Tagyoung Chung, Jing Huang, Nanyun Peng, and Wenbo Zhao, "CLIPSynth: Learning Text-to-audio Synthesis from Videos using CLIP and Diffusion Models," *CVPR Workshop on Sight and Sound (WSS)*, 2023.

Hao-Wen Dong, Ke Chen, Shlomo Dubnov, Julian McAuley, and Taylor Berg-Kirkpatrick, "Multitrack Music Transformer," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

Hao-Wen Dong, Naoya Takahashi, Yuki Mitsufuji, Julian McAuley, and Taylor Berg-Kirkpatrick, "CLIPSep: Learning Text-queried Sound Separation with Noisy Unlabeled Videos," *International Conference on Learning Representations (ICLR)*, 2023.

Ke Chen, Hao-Wen Dong, Yi Luo, Julian McAuley, Taylor Berg-Kirkpatrick, Miller Puckette, and Shlomo Dubnov, "Improving Choral Music Separation through Expressive Synthesized Data from Sampled Instruments," *International Society for Music Information Retrieval Conference (ISMIR)*, 2022.

Hao-Wen Dong, Cong Zhou, Taylor Berg-Kirkpatrick, and Julian McAuley, "Deep Performer: Score-to-Audio Music Performance Synthesis," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

Hao-Wen Dong, Chris Donahue, Taylor Berg-Kirkpatrick, and Julian McAuley, "Towards Automatic Instrumentation by Learning to Separate Parts in Symbolic Multitrack Music," *International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

Sachinda Edirisooriya, Hao-Wen Dong, Julian McAuley, and Taylor Berg-Kirkpatrick, "An Empirical Evaluation of End-to-End Polyphonic Optical Music Recognition," *International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

Yin-Cheng Yeh, Wen-Yi Hsiao, Satoru Fukayama, Tetsuro Kitahara, Benjamin Genchel, Hao-Min Liu, Hao-Wen Dong, Yian Chen, Terence Leong, and Yi-Hsuan Yang, "Automatic Melody Harmonization with Triad Chords: A Comparative Study," *Journal of New Music Research (JNMR)*, 50(1):37–51, 2021.

Hao-Wen Dong, Ke Chen, Julian McAuley, and Taylor Berg-Kirkpatrick, "MusPy: A Toolkit for Symbolic Music Generation," *International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

Hao-Wen Dong and Yi-Hsuan Yang, "Convolutional Generative Adversarial Networks with Binary Neurons for Polyphonic Music Generation," *International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

Hao-Wen Dong, Wen-Yi Hsiao, and Yi-Hsuan Yang, "Pypianoroll: Open Source Python Package for Handling Multitrack Pianorolls," *ISMIR Late-Breaking Demos*, 2018.

Hao-Wen Dong[*], Wen-Yi Hsiao[*], Li-Chia Yang, and Yi-Hsuan Yang, "MuseGAN: Multi-Track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment," *AAAI Conference on Artificial Intelligence (AAAI)*, 2018. ([*]equal contribution)

Hao-Wen Dong[*], Wen-Yi Hsiao[*], Li-Chia Yang, and Yi-Hsuan Yang, "MuseGAN: Demonstration of a Convolutional GAN Based Model for Generating Multi-track Piano-rolls," *ISMIR Late-Breaking Demos*, 2017. ([*]equal contribution)

ABSTRACT OF THE DISSERTATION

Generative AI for Music and Audio

by

Hao-Wen Dong

Doctor of Philosophy in Computer Science

University of California San Diego, 2024

Professor Taylor Berg-Kirkpatrick, Co-Chair
Professor Julian McAuley, Co-Chair

Generative AI has been transforming the way we interact with technology and consume content. In the next decade, AI technology will reshape how we create audio content in various media, including music, theater, films, games, podcasts, and short videos. In this dissertation, I introduce the three main directions of my research centered around *generative AI for music and audio*: 1) multitrack music generation, 2) assistive music creation tools, and 3) multimodal learning for audio and music. Through my research, I aim to answer the following two fundamental questions: 1) *How can AI help professionals or amateurs create music and audio content?* 2) *Can AI learn to create music in a way similar to how humans learn music?* My long-term goal is to lower the barrier of entry for music composition and democratize audio content creation.

Chapter 1

Introduction

AI is the study of how to make computers do things
at which, at the moment, people are better.
—Elaine Rich and Kevin Knight

Generative AI has been transforming the way we interact with technology and consume content. The recent success of large language model-based chatbots (e.g., OpenAI's ChatGPT[1] and Google's Gemini[2]), AI assistants (e.g., GitHub and Microsoft Copilot) and text-to-image generation systems (e.g., Adobe Firefly,[3] Midjourney[4] and Stable Diffusion[5]) showcases how AI-powered technology can be integrated into professional workflows and boost human productivity. In the next decade, generative AI technology will also reshape how we create audio content in the $2.3 trillion global entertainment industry, including the music, film, TV, podcast and gaming sectors. Take AI-powered music creation for example: On one hand, we have witnessed major progress in automatic music composition (Briot et al., 2017; Huang et al., 2020), which has long been considered as a grand challenge of AI. On the other hand, our expectations of *AI Music* today has expanded to cover the whole music creation process—from composition, arrangement, sound production, recording to mixing (De Man et al., 2019). With a growing momentum in both academia and industry, AI-powered audio creation has been gaining attention in the broader AI community.

[1] https://chat.openai.com/
[2] https://gemini.google.com/
[3] https://firefly.adobe.com/
[4] https://www.midjourney.com/
[5] https://stability.ai/stable-image

Figure 1.1. Looking back to the past, how music and technology interacts has always been a two-way process. (Left) the violin-making industry grows with the classical music, and together create the golden age of classical music. (Right) the invention and development of synthesizers and drum machines helped popularize electronic music. Image sources (from left to right): 1) Mozart83, Public domain, via Wikimedia Commons, 2) Hildegard Dodel, Public domain, via Wikimedia Commons, 3) yan, CC BY-SA 4.0, via Wikimedia Commons, and 4) taken at Hamamatsu Museum of Musical Instruments, August 2019.

My research springs from two fundamental questions: 1) *How can AI help professionals or amateurs create music and audio content?* 2) *Can AI learn to create music in a way similar to how humans learn music?* From a musical perspective, technology has always been a driving factor of music evolution. For example, the study of acoustics and musical instrument making fostered the development of classical music; the invention of synthesizers and drum machines helped popularize electronic music. I am thus interested in exploring how the latest AI technology can empower artists to create novel contents. From a technical perspective, music possesses a unique complexity in that music follows rules and patterns while being creative and expressive at the same time. I am thus fascinated about the idea of building intelligent systems that can learn, create and play music like humans do. I envision the future development of AI Music to be a two-way process—*new technology creates new music; new music inspires new technology*.

Motivated by this belief, I study a wide range of topics centered around *Generative AI for Music and Audio*, including multitrack music generation (Dong et al., 2018a; Dong et al., 2017; Dong and Yang, 2018; Dong et al., 2023a; Xu et al., 2023; Liu et al., 2018a), automatic instrumentation (Dong et al., 2021), automatic arrangement (Dong et al., 2018a; Liu et al., 2018a), automatic harmonization (Yeh et al., 2021), music performance synthesis (Dong et al.,

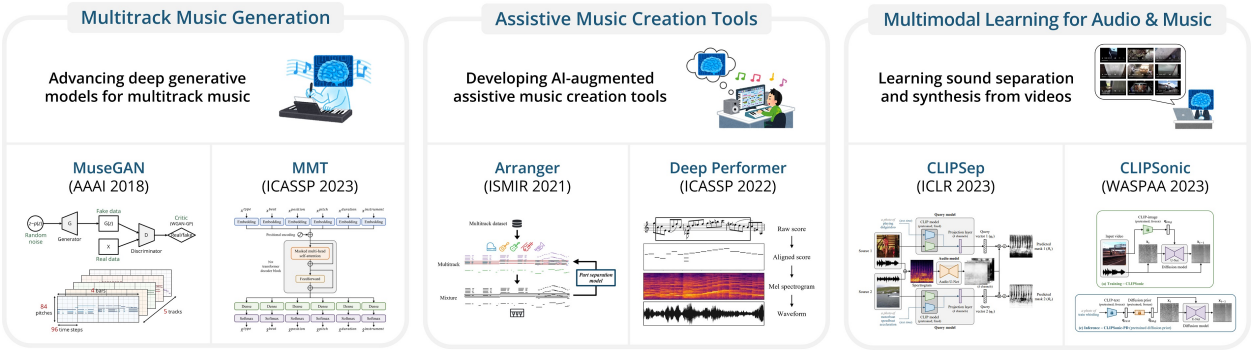


Figure 1.2. An overview of the three main directions of my research.

2022), text-queried sound separation (Dong et al., 2023d), text-to-audio synthesis (Dong et al., 2023c; Dong et al., 2023b) and symbolic music processing software (Dong et al., 2018b; Dong et al., 2020).

My research can be categorized into three main directions, as shown in Figure 1.2: 1) **multitrack music generation**—*advancing deep generative models for multitrack music*, 2) **assistive music creation tools**—*developing AI tools that can help musicians and amateurs create music*, and 3) **multimodal learning for audio and music**—*learning sound separation and synthesis from noisy videos*.

1.1 Multitrack Music Generation

Researchers have been working on automatic music composition for decades, and it has long been viewed as a grand challenge of AI. I started this thread of research on multitrack music generation in 2017. Back then, prior work on deep learning-based music generation had focused on generating melodies, lead sheets (i.e., melodies and chords) or four-part chorales Briot et al., 2017. However, modern pop music often consists of multiple instruments or tracks. To make deep learning technology applicable in modern music production workflow, it is important to modernize deep learning models for multitrack music generation.

Witnessing the lack of infrastructure for symbolic music generation when conducting these research projects, I developed libraries for processing symbolic music to consolidate the infrastructure of music generation research (Dong et al., 2018b; Dong et al., 2020). The Python toolkits I developed to process symbolic music for machine learning applications

have been widely used in the field. With these libraries, researchers can easily download commonly used datasets programmatically and be liberated from reimplementing tedious data processing routines. The toolkits also allowed me to conduct *the first large-scale experiment that measures the cross-dataset generalizability of deep neural networks for music* (Dong et al., 2020). This work will be presented in Chapter 2.

Moreover, I study efficient music generation model with an eye to enabling real time improvisation or near real time creative applications. In Dong et al., 2023a, I proposed the Multitrack Music Transformer (MMT) model that achieves comparable performance with state-of-the-art systems, landing in between two recently proposed models in a subjective listening test, while achieving substantial speedups and memory reductions over both. Further, I presented *the first systematic analysis of musical self-attention*, where I showed that the trained model learns a relative attention in certain aspects of music. With the great success brought by large-scale generative pretraining in natural language processing, my ongoing work aims to scale this method up using the largest ever symbolic music dataset containing more than one million scores, which I collected and compiled from the MuseScore forum. This work will be presented in Chapter 3.

1.2 Assistive Music Creation Tools

Music creation today is still largely limited to professional musicians for it requires a certain level of knowledge in music theory, music notation and music production tools. Apart from generating new music content from scratch, another line of my research focuses on developing AI-augmented tools to assist amateurs to create and perform music. My long-term goal along this research direction is to lower the entry barrier of music composition and make music creation accessible to everyone.

For example, in (Dong et al., 2021), I developed *the first deep learning model for automatic instrumentation*. Instrumentation refers to the process where a musician arranges a solo piece for a certain ensemble such as a string quartet or a rock band. This can be challenging for amateur composers as it requires domain knowledge of each target instrument. In this work, I proposed a new machine learning model that can produce convincing instrumentation for a solo piece by framing this problem as a sequential multi-class classification

problem. Such an automatic instrumentation system can suggest potential instrumentation for amateur composers, especially useful when arranging for an unfamiliar ensemble. Further, the proposed model can empower a musician to play multiple instruments on a single keyboard at the same time. This work will be presented in Chapter 4.

Another example is my work on music performance synthesis (Dong et al., 2022). While synthesizers play a critical role and are intensively used in modern music production, existing synthesizers either requires an input with expressive timing or allows only monophonic inputs. In light of the similarities between text-to-speech (TTS) and score-to-audio synthesis, I showed in this work that we can adapt a state-of-the-art TTS model for music performance synthesis. Moreover, I proposed a novel mechanism to enable polyphonic music synthesis. This work represents *the first deep learning based polyphonic synthesizer that can synthesize a score into a natural, expressive performance*. This work will be presented in Chapter 5.

1.3 Multimodal Learning for Audio and Music

The third line of my research focuses on multimodal learning for audio and music. Sound is an integral part of movies, dramas, documentaries, podcasts, games, short videos and audiobooks. In these media, audio and music production tools need to interact with inputs from other modalities such as text and images, and thus multimodal models are critical in enabling controllable creation tools for music and audio in these applications.

Along this direction, I have worked on text-queried sound separation (Dong et al., 2023d) and text-to-audio synthesis (Dong et al., 2023c; Dong et al., 2023b). Unlike existing work that relies on a large amount of paired audio-text data, I explore a new direction of approaching bimodal learning for text and audio through leveraging the visual modality as a bridge. The key idea behind my study is to combine the naturally-occurring audio-visual correspondence in videos and the multimodal representation learned by contrastive language-vision pretraining (CLIP). Based on this idea, I developed *the first text-queried sound separation model that can be trained without any text-audio pairs* (Dong et al., 2023d). Text-queried sound separation aims to separate a specific sound out from a mixture of sounds given a text query, which has many downstream applications in audio post-production

such as editing and remixing. I showed that the proposed model can successfully learn text-queried sound separation using only noisy unlabeled videos, and it even achieves competitive performance against a supervised model in some settings. Moreover, I built *the first text-to-audio synthesis model that requires no text-audio pairs during training* (Dong et al., 2023c; Dong et al., 2023b). The proposed model learns to synthesize audio given text queries, which can find applications in video and audio editing software. One of the key benefits of the approach studied in my work lies in its scalability to large video datasets in the wild as we only need unlabeled videos for training. This work will be presented in Chapters 6 and 7.

1.4 Dissertation Organization

The rest of this dissertation is organized as follows: Chapters 2 to 7 are reprints of six conference papers published during my PhD. Specifically, Chapter 2 is a reprint of *"MusPy: A Toolkit for Symbolic Music Generation"* (Dong et al., 2020) published in ISMIR 2020. Chapter 3 is a reprint of *"Multitrack Music Transformer"* (Dong et al., 2023a) published in ICASSP 2023. Chapter 4 is a reprint of *"Towards Automatic Instrumentation by Learning to Separate Parts in Symbolic Multitrack Music"* (Dong et al., 2021) published in ISMIR 2021. Chapter 5 is a reprint of *"Deep Performer: Score-to-Audio Music Performance Synthesis"* (Dong et al., 2022) published in ICASSP 2022. Chapter 6 is a reprint of *"CLIPSep: Learning Text-queried Sound Separation with Noisy Unlabeled Videos"* (Dong et al., 2023d) published in ICLR 2023. Chapter 7 is a reprint of *"CLIPSonic: Text-to-Audio Synthesis with Unlabeled Videos and Pretrained Language-Vision Models"* (Dong et al., 2023b) published in WASPAA 2023. Finally, Chapter 8 concludes this dissertation and discusses my future research directions.

Think, Think, Think

—Winnie the Pooh

Chapter 2

MusPy: A Toolkit for Symbolic Music Generation

Abstract

In this paper, we present MusPy, an open source Python library for symbolic music generation. MusPy provides easy-to-use tools for essential components in a music generation system, including dataset management, data I/O, data preprocessing and model evaluation. In order to showcase its potential, we present statistical analysis of the eleven datasets currently supported by MusPy. Moreover, we conduct a cross-dataset generalizability experiment by training an autoregressive model on each dataset and measuring held-out likelihood on the others—a process which is made easier by MusPy's dataset management system. The results provide a map of domain overlap between various commonly used datasets and show that some datasets contain more representative cross-genre samples than others. Along with the dataset analysis, these results might serve as a guide for choosing datasets in future research. Source code and documentation are available at https://github.com/salu133445/muspy.

2.1 Introduction

Recent years have seen progress on music generation, thanks largely to advances in machine learning (Briot et al., 2017). A music generation pipeline usually consists of several steps—data collection, data preprocessing, model creation, model training and model

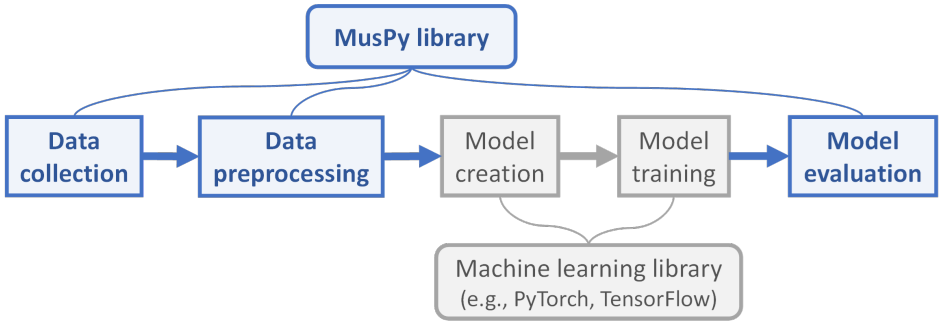


Figure 2.1. An example of a learning-based music generation system. MusPy provides basic routines specific to music as well as interfaces to machine learning frameworks.

evaluation, as illustrated in Figure 2.1. While some components need to be customized for each model, others can be shared across systems. For symbolic music generation in particular, a number of datasets, representations and metrics have been proposed in the literature (Briot et al., 2017). As a result, an easy-to-use toolkit that implements standard versions of such routines could save a great deal of time and effort and might lead to increased reproducibility. However, such tools are challenging to develop for a variety of reasons.

First, though there are a number of publicly-available symbolic music datasets, the diverse organization of these collections and the various formats used to store them presents a challenge. These formats are usually designed for different purposes. Some focus on playback capability (e.g., MIDI), some are developed for music notation softwares (e.g., MusicXML (Good, 2001) and LilyPond (*LilyPond* n.d.)), some are designed for organizing musical documents (e.g., Music Encoding Initiative (MEI) (Hankinson et al., 2011)), and others are research-oriented formats that aim for simplicity and readability (e.g., MuseData (Hewlett, 1997) and Humdrum (Huron, 1997). Oftentimes researchers have to implement their own preprocessing code for each different format. Moreover, while researchers can implement their own procedures to access and process the data, issues of reproducibility due to the inconsistency of source data have been raised in (Bittner et al., 2019) for audio datasets.

Second, music has hierarchy and structure, and thus different levels of abstraction can lead to different representations (Dannenberg, 1993). Moreover, a number of music representations designed specially for generative modeling of music have also been proposed in prior art, for example, as a sequence of pitches (Mozer, 1994; Eck and Schmidhuber, 2002; Boulanger-Lewandowski et al., 2012; Roberts et al., 2018), events (Oore et al., 2020; Huang

et al., 2019; Donahue et al., 2019; Huang and Yang, 2020), notes (Mogren, 2016) or a time-pitch matrix (i.e., a piano roll) (Yang et al., 2017; Dong et al., 2018a).

Finally, efforts have been made toward more robust objective evaluation metrics for music generation systems (Yang and Lerch, 2018) as these metrics provide not only an objective way for comparing different models but also indicators for monitoring training progress in machine learning-based systems. Given the success of mir_eval (Raffel et al., 2014) in evaluating common MIR tasks, a library providing implementations of commonly used evaluation metrics for music generation systems could help improve reproducibility.

To manage the above challenges, we find a toolkit dedicated for music generation a timely contribution to the MIR community. Hence, we present in this paper a new Python library, MusPy, for symbolic music generation. It provides essential tools for developing a music generation system, including dataset management, data I/O, data preprocessing and model evaluation.

With MusPy, we provide a statistical analysis on the eleven datasets currently supported by MusPy, with an eye to unveiling statistical differences between them. Moreover, we conduct three experiments to analyze their relative diversities and cross-dataset domain compatibility of the various datasets. These results, along with the statistical analysis, together provide a guide for choosing proper datasets for future research. Finally, we also show that combining multiple heterogeneous datasets could help improve generalizability of a music generation system.

2.2 Related Work

Few attempts, to the best of our knowledge, have been made to develop a dedicated library for music generation. The Magenta project (*Magenta* n.d.) represents the most notable example. While MusPy aims to provide fundamental routines in data collection, preprocessing and analysis, Magenta comes with a number of model instances, but is tightly bound with TensorFlow (Abadi et al., 2016). In MusPy, we leave the model creation and training to dedicated machine learning libraries, and design MusPy to be flexible in working with different machine learning frameworks.

There are several libraries for working with symbolic music. music21 (Cuthbert and

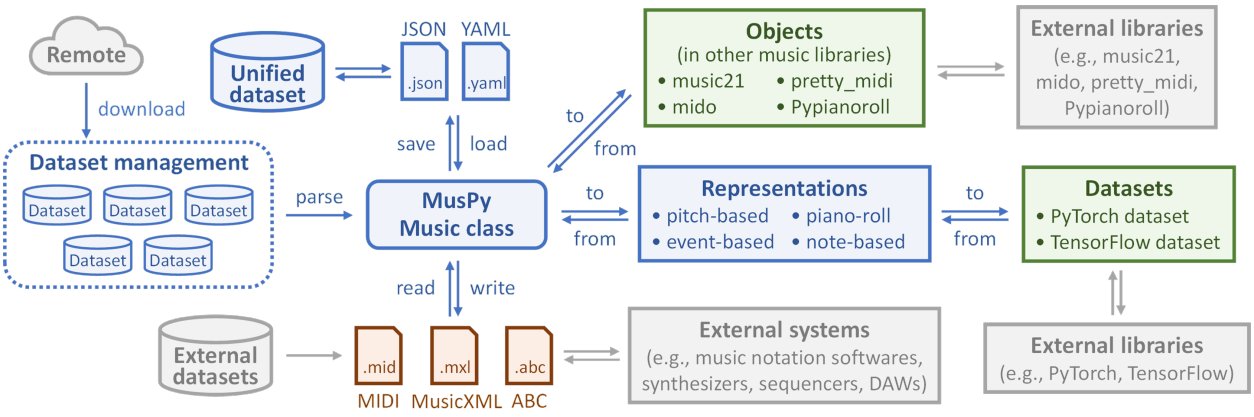


Figure 2.2. System diagram of MusPy. The MusPy Music object at the center is the core element of MusPy.

Ariza, 2010) is one of the most representative toolkits and targets studies in computational musicology. While music21 comes with its own corpus, MusPy does not host any dataset. Instead, MusPy provides functions to download datasets from the web, along with tools for managing different collections, which makes it easy to extend support for new datasets in the future. jSymbolic (Mckay and Fujinaga, 2006) focuses on extracting statistical information from symbolic music data. While jSymbolic can serve as a powerful feature extractor for training supervised classification models, MusPy focuses on generative modeling of music and supports different commonly used representations in music generation. In addition, MusPy provides several objective metrics for evaluating music generation systems.

Related cross-dataset generalizability experiments (Donahue et al., 2019) show that pretraining on a cross-domain data can improve music generation results both qualitatively and quantitatively. MusPy's dataset management system makes it easier for us to thoroughly verify this hypothesis by examining pairwise generalizabilities between various datasets.

2.3 MusPy

MusPy is an open source Python library dedicated for symbolic music generation. Figure 2.2 presents the system diagram of MusPy. It provides a core class, MusPy Music class, as a universal container for symbolic music. Dataset management system, I/O interfaces and model evaluation tools are then built upon this core container. We provide in Figure 2.3 examples of data preparation and result writing pipelines using MusPy.

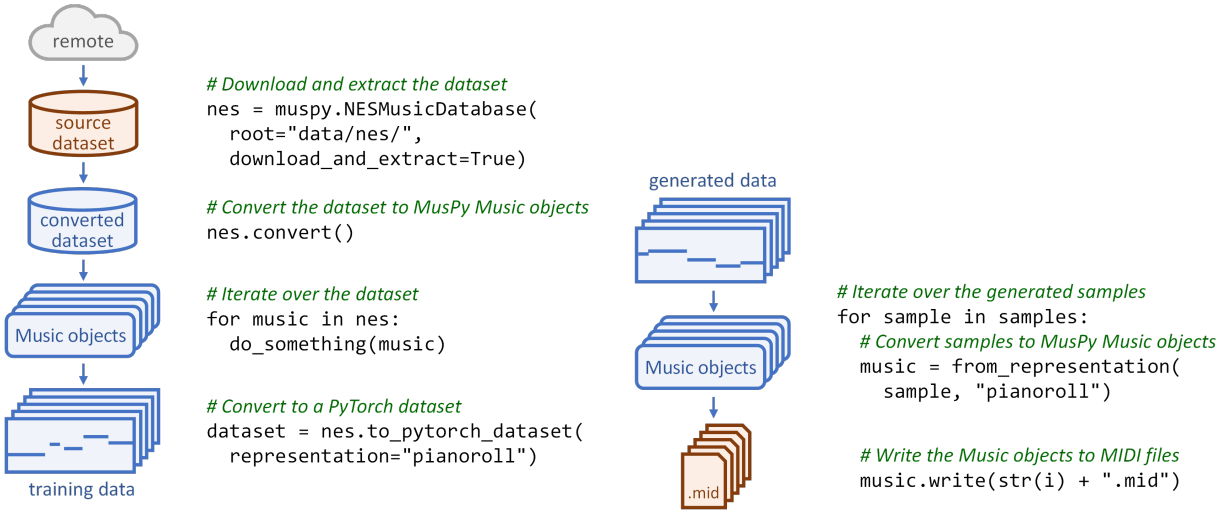


Figure 2.3. Examples of (a) training data preparation and (b) result writing pipelines using MusPy.

2.3.1 MusPy Music class and I/O interfaces

We aim at finding a middle ground among existing formats for symbolic music and design a unified format dedicated for music generation. MIDI, as a communication protocol between musical devices, uses velocities to indicate dynamics, beats per minute (bpm) for tempo markings, and control messages for articulation, but it lacks the concepts of notes, measures and symbolic musical markings. In contrast, MusicXML, as a sheet music exchanging format, has the concepts of notes, measures and symbolic musical markings and contains visual layout information, but it falls short on playback-related data. For a music generation system, however, both symbolic and playback-specific data are important. Hence, we follow MIDI's standard for playback-related data and MusicXML's standard for symbolic musical markings.

In fact, the MusPy Music class naturally defines a universal format for symbolic music, which we will refer to as the MusPy format, and can be serialized into a human-readable JSON/YAML file. Table 2.1 summarizes the key differences among MIDI, MusicXML and the proposed MusPy formats. Using the proposed MusPy Music class as the internal representation for music data, we then provide I/O interfaces for common formats (e.g., MIDI, MusicXML and ABC) and interfaces to other symbolic music libraries (e.g., music21 (Cuthbert and Ariza, 2010), mido (*Mido: MIDI Objects for Python* n.d.), pretty_midi (Raffel

Table 2.1. Comparisons of MIDI, MusicXML and the proposed MusPy formats. Triangle marks indicate optional or limited support.

	MIDI	MusicXML	MusPy
Sequential timing	✓		✓
Playback velocities	✓	△	✓
Program information	✓	△	✓
Layout information		✓	
Note beams and slurs		✓	
Song/source meta data	△	✓	✓
Track/part information	△	✓	✓
Dynamic/tempo markings		✓	✓
Concept of notes		✓	✓
Measure boundaries		✓	✓
Human readability		△	✓

and Ellis, 2014) and Pypianoroll (Dong et al., 2018b)). Figure 2.3(b) provides an example of result writing pipeline using MusPy.

2.3.2 Dataset management

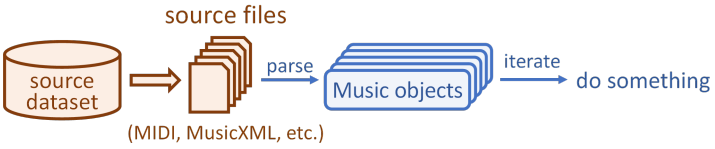
MusPy provides an easy-to-use dataset management system similar to torchvision datasets (Marcel and Rodriguez, 2010) and TensorFlow Dataset (*TensorFlow Datasets* n.d.). Table 2.2 presents the list of datasets currently supported by MusPy and their comparisons. Each supported dataset comes with a class inherited from the base MusPy Dataset class. The modularized and flexible design of the dataset management system makes it easy to handle local data collections or extend support for new datasets in the future. Figure 2.4 illustrates the two internal processing modes when iterating over a MusPy Dataset object. In addition, MusPy provides interfaces to PyTorch (Paszke et al., 2019) and TensorFlow (Abadi et al., 2016) for creating input pipelines for machine learning (see Figure 2.3(a) for an example).

2.3.3 Representations

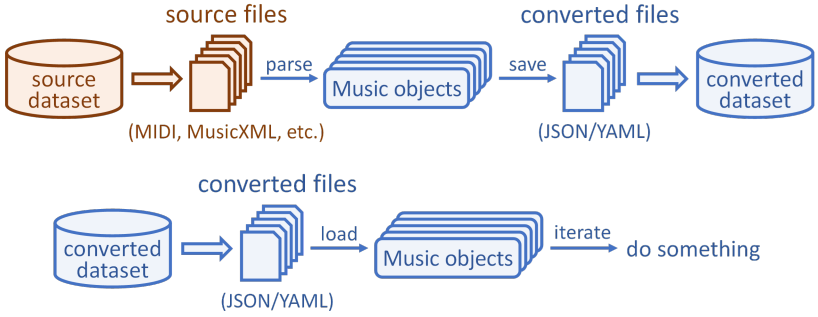
Music has multiple levels of abstraction, and thus can be expressed in various representations. For music generation in particular, several representations designed for generative modeling of symbolic music have been proposed and used in the literature (Briot et al., 2017). These representations can be broadly categorized into four types—the pitch-based (Mozer, 1994; Eck and Schmidhuber, 2002; Boulanger-Lewandowski et al., 2012; Roberts et al.,

Table 2.2. Comparisons of datasets currently supported by MusPy. Triangle marks indicate partial support. Note that, in this version, only MusicXML and MIDI files are included for the music21 Corpus.

Dataset	Format	Hours	Songs	Genre	Melody	Chords	Multitrack
Lakh MIDI Dataset (LMD) (Raffel, 2016)	MIDI	>9000	174,533	misc	△	△	△
MAESTRO Dataset (Hawthorne et al., 2019)	MIDI	201.21	1,282	classical			
Wikifonia Lead Sheet Dataset (*Wikifonia* n.d.)	MusicXML	198.40	6,405	misc	✓	✓	
Essen Folk Song Database (*Essen Folk Song Database* n.d.)	ABC	56.62	9,034	folk	✓	✓	
NES Music Database (Donahue et al., 2018)	MIDI	46.11	5,278	game	✓		✓
Hymnal Tune Dataset (*Hymnal* n.d.)	MIDI	18.74	1,756	hymn	✓		
Hymnal Dataset (*Hymnal* n.d.)	MIDI	17.50	1,723	hymn			
music21 Corpus (Cuthbert and Ariza, 2010)	misc	16.86	613	misc	△		△
Nottingham Database (NMD) (*Nottingham Database* n.d.)	ABC	10.54	1,036	folk	✓	✓	
music21 JSBach Corpus (Cuthbert and Ariza, 2010)	MusicXML	3.46	410	classical			✓
JSBach Chorale Dataset (Boulanger-Lewandowski et al., 2012)	MIDI	3.21	382	classical			✓



(a) on-the-fly mode



(b) preconverted mode

Figure 2.4. Two internal processing modes for iterating over a MusPy Dataset object.

Table 2.3. Comparisons of representations supported by MusPy. T and N denote the numbers of time steps and notes, respectively. Note that the configurations can be modified to meet specific requirements and use cases.

Representation	Shape	Values	Default configurations
Pitch-based	$T \times 1$	$\{0, 1, \ldots, 129\}$	128 note-ons, 1 hold, 1 rest (*support only monophonic music*)
Event-based	$T \times 1$	$\{0, 1, \ldots, 387\}$	128 note-ons, 128 note-offs, 100 time shifts, 32 velocities
Piano-roll	$T \times 128$	$\{0, 1\}$ or $\mathbb{R}^+$	$\{0, 1\}$ for binary piano rolls; $\mathbb{R}^+$ for piano rolls with velocities
Note-based	$N \times 4$	$\mathbb{N}$ or $\mathbb{R}^+$	List of $(time, pitch, duration, velocity)$ tuples

2018), the event-based (Oore et al., 2020; Huang et al., 2019; Donahue et al., 2019; Huang and Yang, 2020), the note-based (Mogren, 2016) and the piano-roll (Yang et al., 2017; Dong et al., 2018a) representations. Table 2.3 presents a comparison of them. We provide in MusPy implementations of these representations and integration to the dataset management system. Figure 2.3(a) provides an example of preparing training data in the piano-roll representation from the NES Music Database using MusPy.

2.3.4 Model evaluation tools

Model evaluation is another critical component in developing music generation systems. Hence, we also integrate into MusPy tools for audio rendering as well as score and piano-roll visualizations. These tools could also be useful for monitoring the training progress or demonstrating the final results. Moreover, MusPy provides implementations of several objective metrics proposed in the literature (Mogren, 2016; Dong et al., 2018a; Wu and Yang, 2020). These objective metrics, as listed below, could be used to evaluate a music generation system by comparing the statistical difference between the training data and the generated samples, as discussed in (Yang and Lerch, 2018).

- *Pitch-related metrics*—polyphony, polyphony rate, pitch-in-scale rate, scale consistency, pitch entropy and pitch class entropy.
- *Rhythm-related metrics*—empty-beat rate, drum-in-pattern rate, drum pattern consistency and groove consistency.

2.3.5 Summary

To summarize, MusPy features the following:

- Dataset management system for commonly used datasets with interfaces to PyTorch and TensorFlow.
- Data I/O for common symbolic music formats (e.g., MIDI, MusicXML and ABC) and interfaces to other symbolic music libraries (e.g., music21, mido, pretty_midi and Pypianoroll).
- Implementations of common music representations for music generation, including the pitch-based, the event-based, the piano-roll and the note-based representations.
- Model evaluation tools for music generation systems, including audio rendering, score and piano-roll visualizations and objective metrics.

All source code and documentation can be found at https://github.com/salu133445/muspy.

2.4 Dataset Analysis

Analyzing datasets is critical in developing music generation systems. With MusPy's dataset management system, we can easily work with different music datasets. Below we compute the statistics of three key elements of a song—length, tempo and key using MusPy, with an eye to unveiling statistical differences among these datasets. First, Figure 2.5 shows the distributions of song lengths for different datasets. We can see that they differ greatly in their ranges, medians and variances.

Second, we present in Figure 2.6 the distributions of initial tempo for datasets that come with tempo information. We can see that all of them are generally bell-shaped but with different ranges and variances. We also note that there are two peaks, 100 and 120 quarter notes per minute (qpm), in Lakh MIDI Dataset (LMD), which is possibly because these two values are often set as the default tempo values in music notation programs and MIDI editors/sequencers. Moreover, in Hymnal Tune Dataset, only around ten percent of songs have an initial tempo other than 100 qpm.

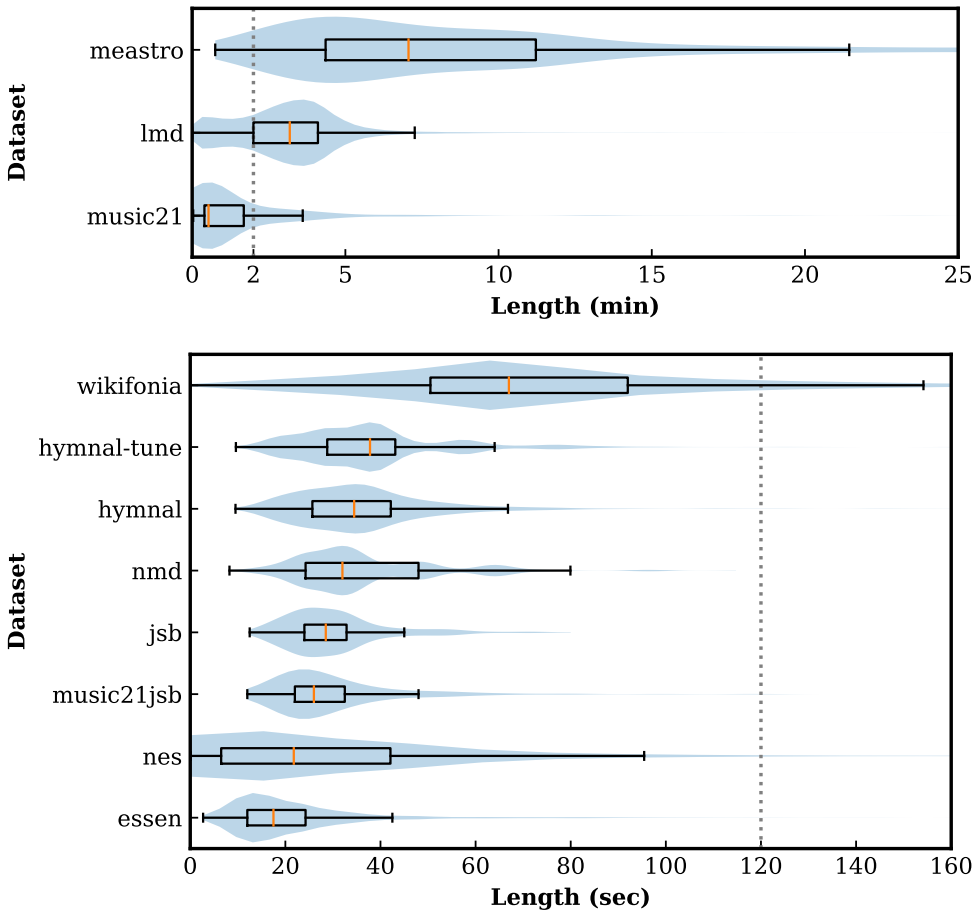


Figure 2.5. Length distributions for different datasets.

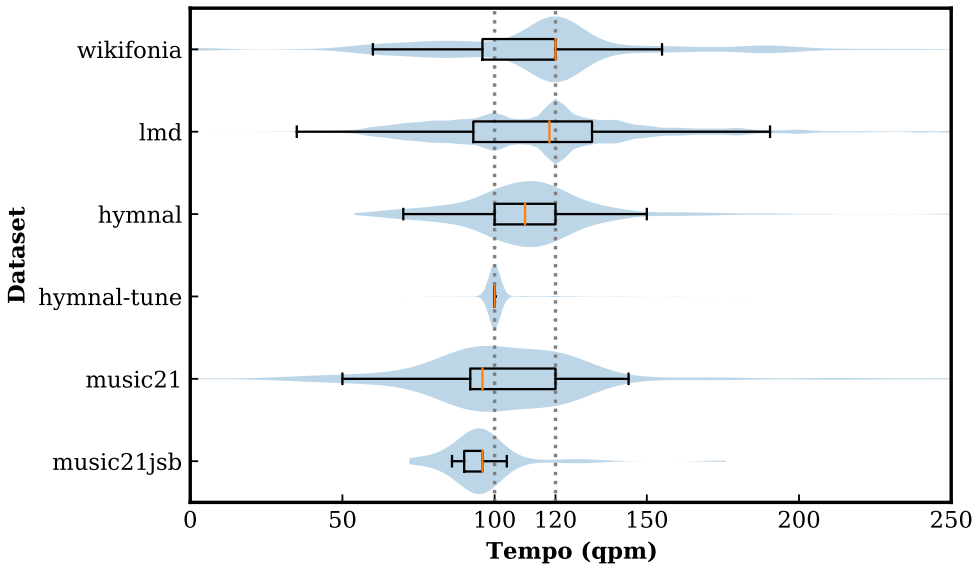


Figure 2.6. Initial-tempo distributions for different datasets (those without tempo information are not presented).

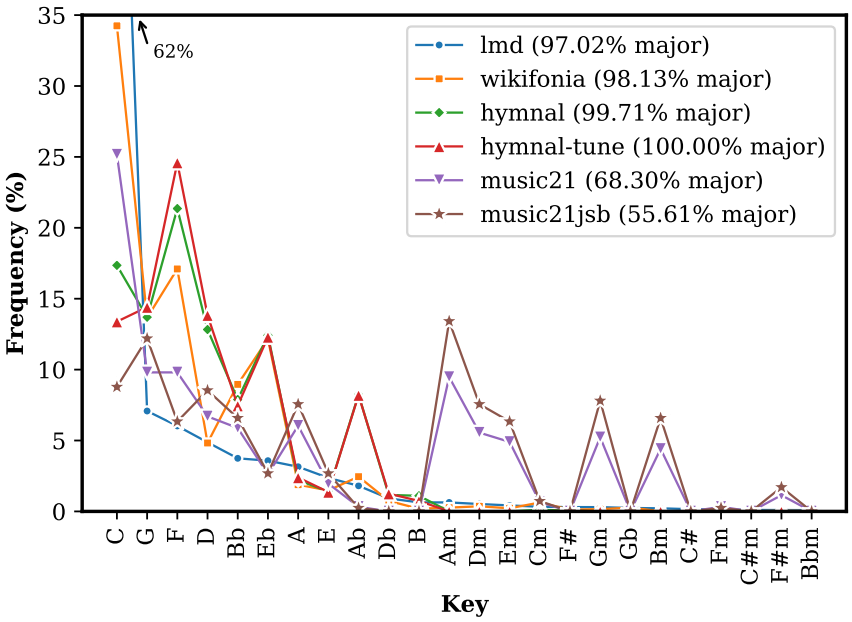


Figure 2.7. Key distributions for different datasets. The keys are sorted w.r.t. their frequencies in Lakh MIDI Dataset.

Finally, Figure 2.7 shows the histograms of keys for different datasets. We can see that the key distributions are rather imbalanced. Moreover, only less than 3% of songs are in minor keys for most datasets except the music21 Corpus. In particular, LMD has the most imbalanced key distributions, which might be due to the fact that C major is often set as the default key in music notation programs and MIDI editors/sequencers.[1] These statistics could provide a guide for choosing proper datasets in future research.

2.5 Experiments and Results

In this section, we conduct three experiments to analyze the relative complexities and the cross-dataset generalizabilities of the eleven datasets currently supported by MusPy (see Table 2.2). We implement four autoregressive models—a recurrent neural network (RNN), a long short-term memory (LSTM) network (Hochreiter and Schmidhuber, 1997), a gated recurrent unit (GRU) network (Cho et al., 2014) and a Transformer network (Vaswani et al., 2017).

[1]Note that key information is considered as a meta message in a MIDI file. It does not affect the playback and thus can be unreliable sometimes.

2.5.1 Experiment settings

For the data, we use the event representation as specified in Table 2.3 and discard velocity events as some datasets have no velocity information (e.g., datasets using ABC format). Moreover, we also include an end-of-sequence event, leading to in total 357 possible events. For simplicity, we downsample each song into four time steps per quarter note and fix the sequence length to 64, which is equivalent to four measures in 4/4 time. In addition, we discard repeat information in MusicXML data and use only melodies in Wikifonia dataset. We split each dataset into train–test–validation sets with a ratio of $8:1:1$. For the training, the models are trained to predict the next event given the previous events. We use the cross entropy loss and the Adam optimizer (Kingma and Ba, 2015). For evaluation, we randomly sample 1000 sequences of length 64 from the test split, and compute the perplexity of these sequences. We implement the models in Python using PyTorch. For reproducibility, source code and hyperparmeters are available at https://github.com/salu133445/muspy-exp.

2.5.2 Autoregressive models on different datasets

In this experiment, we train the model on some dataset $\mathcal{D}$ and test it on the same dataset $\mathcal{D}$. We present in Figure 2.8 the perplexities for different models on different datasets. We can see that all models have similar tendencies. In general, they achieve smaller perplexities for smaller, homogeneous datasets, but result in larger perplexities for larger, more diverse datasets. That is, the test perplexity could serve as an indicator for the diversity of a dataset. Moreover, Figure 2.9 shows perplexities versus dataset sizes (in hours). By categorizing datasets into multi-pitch (i.e., accepting any number of concurrent notes) and monophonic datasets, we can see that the perplexity is positively correlated to the dataset size within each group.

2.5.3 Cross-dataset generalizability

In this experiment, we train a model on some dataset $\mathcal{D}$, while in addition to testing it on the same dataset $\mathcal{D}$, we also test it on each other dataset $\mathcal{D}'$. We present in Figure 2.10 the perplexities for each train–test dataset pair. Here are some observations:

- Cross dataset generalizability is not symmetric in general. For example, a model

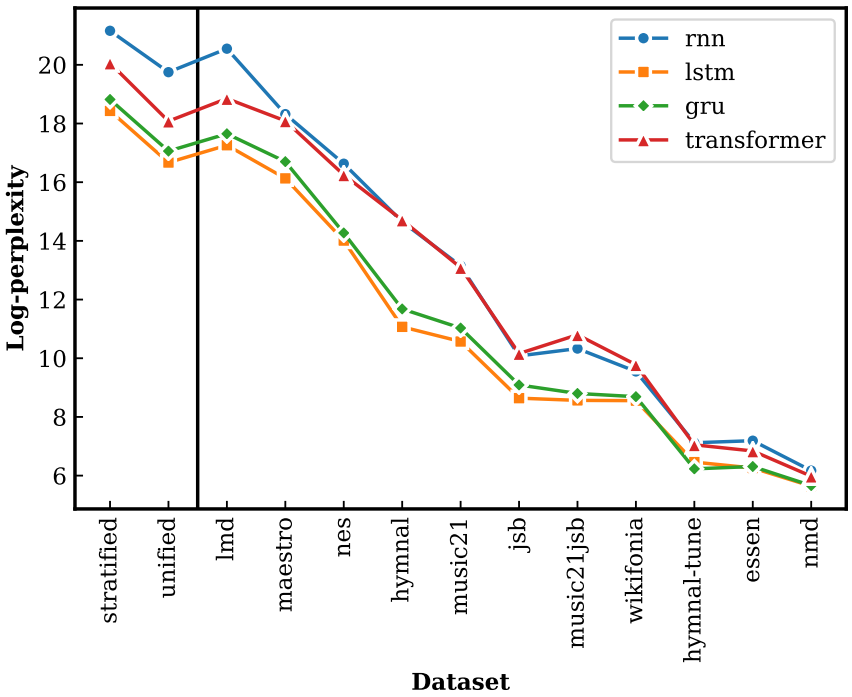


Figure 2.8. Log-perplexities for different models on different datasets, sorted by the values for the LSTM model.

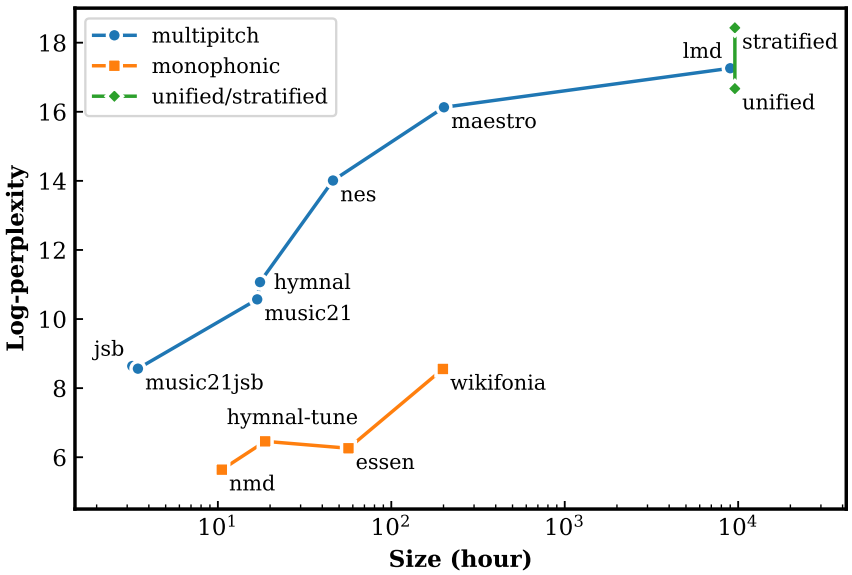


Figure 2.9. Log-perplexities for the LSTM model versus dataset size in hours. Each point corresponds to a dataset.

trained on LMD generalizes well to all other datasets, while not all models trained on other datasets generalize to LMD, which is possibly due to the fact that LMD is a large, cross-genre dataset.

- Models trained on multi-pitch datasets generalize well to monophonic datasets, while models trained on monophonic datasets do not generalize to multi-pitch datasets (see the red block in Figure 2.10).
- The model trained on JSBach Chorale Dataset does not generalize to any of the other datasets (see the orange block in Figure 2.10). This is possibly because its samples are downsampled to a resolution of quarter note, which leads to a distinct note duration distribution.
- Most datasets generalize worse to NES Music Database compared to other datasets (see the green block in Figure 2.10). This is possibly due to the fact that NES Music Database contains only game soundtracks.

2.5.4 Effects of combining heterogeneous datasets

From Figure 2.10 we can see that LMD has the best generalizability, possibly because it is large, diverse and cross-genre. However, a model trained on LMD does not generalize well to NES Music Database (see the brown block in the close-up of Figure 2.10). We are thus interested in whether combing multiple heterogeneous datasets could help improve generalizability.

We combine all eleven datasets listed in Table 2.2 into one large *unified* dataset. Since these datasets differ greatly in their sizes, simply concatenating the datasets might lead to severe imbalance problem and bias toward the largest dataset. Hence, we also consider a version that adopts stratified sampling during training. Specifically, to acquire a data sample in the *stratified* dataset, we uniformly choose one dataset out of the eleven datasets, and then randomly pick one sample from that dataset. Note that stratified sampling is disabled at test time.

We also include in Figures 2.8 to 2.10 the results for these two datasets. We can see from Figure 2.10 that combining datasets from different sources improves the generalizabil-

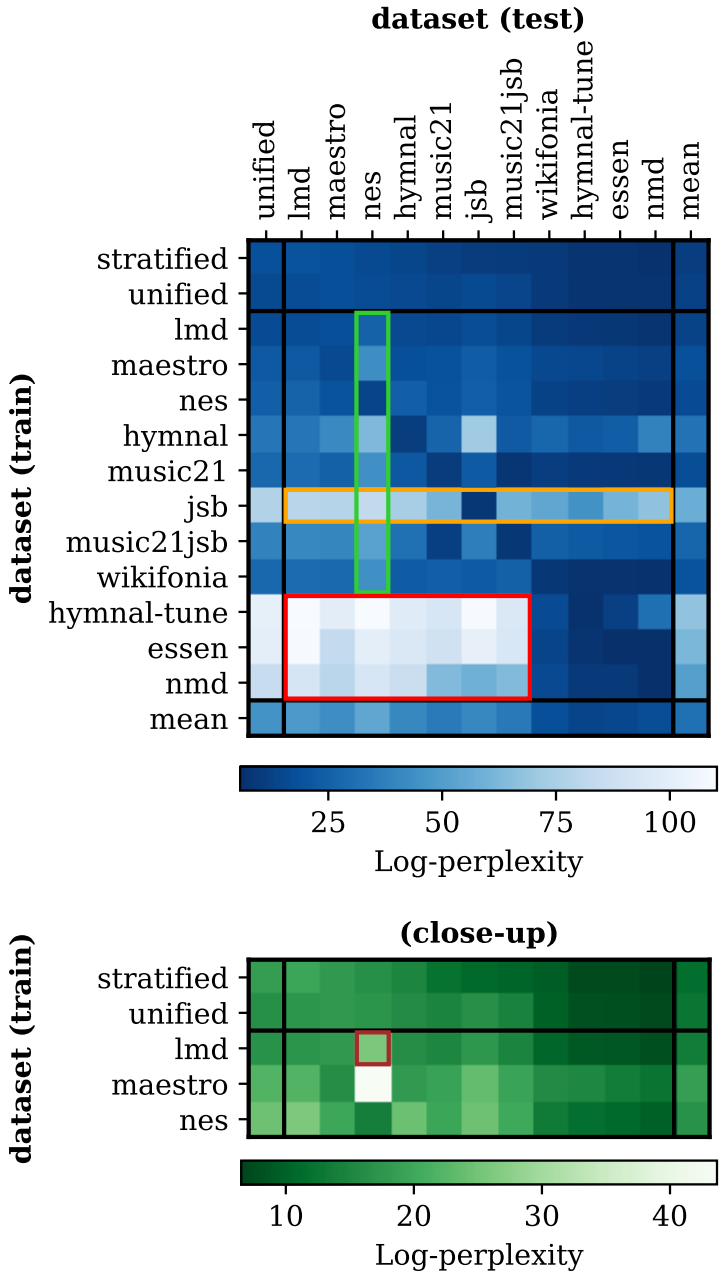


Figure 2.10. Cross-dataset generalizability results. The values and colors represent the log-perplexities of a LSTM model trained on a specific dataset (row) and tested on another dataset (column). The datasets are sorted by the diagonal values, i.e., trained and tested on the same dataset.

ity of the model. This is consistent with the finding in (Donahue et al., 2019) that models trained on certain cross-domain datasets generalize better to other unseen datasets. Moreover, stratified sampling alleviates the source imbalance problem by reducing perplexities in most datasets with a sacrifice of an increased perplexity on LMD.

2.6 Conclusion

We have presented MusPy, a new toolkit that provides essential tools for developing music generation systems. We discussed the designs and features of the library, along with data pipeline examples. With MusPy's dataset management system, we conducted a statistical analysis and experiments on the eleven currently supported datasets to analyze their relative diversities and cross-dataset generalizabilities. These results could help researchers choose appropriate datasets in future research. Finally, we showed that combining heterogeneous datasets could help improve generalizability of a machine learning model.

* * *

This chapter, in full, is a reprint of the material as it appears in "MusPy: A Toolkit for Symbolic Music Generation" by Hao-Wen Dong, Ke Chen, Julian McAuley and Taylor Berg-Kirkpatrick, which was published in the Proceedings of the International Society for Music Information Retrieval Conference (ISMIR) in 2020. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Multitrack Music Transformer

Abstract

Existing approaches for generating multitrack music with transformer models have been limited in terms of the number of instruments, the length of the music segments and slow inference. This is partly due to the memory requirements of the lengthy input sequences necessitated by existing representations. In this work, we propose a new multitrack music representation that allows a diverse set of instruments while keeping a short sequence length. Our proposed Multitrack Music Transformer (MMT) achieves comparable performance with state-of-the-art systems, landing in between two recently proposed models in a subjective listening test, while achieving substantial speedups and memory reductions over both, making the method attractive for real time improvisation or near real time creative applications. Further, we propose a new measure for analyzing musical self-attention and show that the trained model attends more to notes that form a consonant interval with the current note and to notes that are 4N beats away from the current step.

3.1 Introduction

Prior work has investigated various approaches for symbolic music generation (Briot et al., 2017; Ji et al., 2020), among which, the transformer model (Vaswani et al., 2017) has become popular given its recent successes in piano music generation (Huang et al., 2019; Huang and Yang, 2020; Hsiao et al., 2021; Muhamed et al., 2021). At the core of a

transformer model is the self-attention mechanism that allows the model to dynamically attend to different parts of the input sequence and aggregate information from the whole sequence. Such capabilities make it suitable for modeling the complex structures and textures in music. However, while prior work has also explored applying transformer models to generate multitrack music (Payne, 2019; Donahue et al., 2019; Ens and Pasquier, 2020; Rütte et al., 2022), successful implementations have only been reported either on a limited set of instruments (Payne, 2019; Donahue et al., 2019) or short music segments (Ens and Pasquier, 2020; Rütte et al., 2022). This is partly due to the long sequence length in existing multitrack music representations, which results in a large memory requirement in training. For example, a GPU with 11GB of memory can only generate 29 seconds of music on average using the REMI+ representation (Rütte et al., 2022) on an orchestral music dataset. Moreover, it can only generate less than four notes per second. These limitations together pose a challenge in scaling transformer models to longer music with many instruments, e.g., orchestral music, and for real-time use cases, e.g., automatic improvisation and human-AI music co-creation.

In this paper, we propose a new multitrack music representation to address the long sequence issue in existing multitrack music representations. Using the proposed representation, we present the Multitrack Music Transformer (MMT) for multitrack music generation. Unlike a standard transformer model, the proposed model uses a decoder-only transformer with multi-dimensional inputs and outputs to reduce its memory complexity. On an orchestral dataset, we show that our proposed model can generate longer music in a faster inference speed than two existing approaches. Through a subjective listening test, we show that the proposed model achieves reasonably good performance in terms of coherence, richness and arrangement as well as the overall quality. Moreover, our proposed representation allows a trained autoregressive model to generate music for a specific set of instruments, a task that has not been well studied in prior work.

Further, while the transformer model has been widely used on symbolic music, it remains unclear how self-attention work for symbolic music. Understanding musical self-attention could reveal future research directions in improving transformer models for music. To the best of our knowledge, existing analysis (Huang et al., 2019; Huang et al., 2018; Chen

Table 3.1. Comparisons of related transformer-based music models.

Model	Multitrack	Instrument control	Compound tokens	Generative modeling
REMI (Huang and Yang, 2020)				✓
MMM (Ens and Pasquier, 2020)	✓			✓
CP (Hsiao et al., 2021)			✓	✓
MusicBERT (Zeng et al., 2021)	✓		✓	
FIGARO (Rütte et al., 2022)	✓			✓
MMT (ours)	✓	✓	✓	✓

and Su, 2021; Wang and Xia, 2021) provides only case studies on few selected samples, lacking a systematic analysis on self-attention for music. Hence, we propose a new quantity to measure the average attention weights that a transformer model assigns to a certain key of a certain difference from the query. Our analysis shows that the proposed model learns a relative self-attention for certain aspects of music, specifically, beat, position and pitch.

Our proposed model provides a novel foundation for future work exploring longer-form and real-time capable multitrack music generation. The systematic analysis also provide insights into improving the self-attention mechanism for music. Audio samples can be found on our demo website: https://salu133445.github.io/mmt/. For reproducibility, all source code and hyperparameters are made publicly available at https://github.com/salu133445/mmt.

3.2 Related Work

Multitrack music generation. Prior work has explored various approaches for symbolic music generation (Briot et al., 2017; Ji et al., 2020), among which generating multitrack music is considered more challenging for its complex interdependency between voices and instruments. In (Dong et al., 2018a; Dong and Yang, 2018), the authors used a convolutional generative adversarial network to generate short, five-track pop music segments. In (Simon et al., 2018), the authors used a variational autoencoder with recurrent neural networks to learn a latent space for multitrack measures. In (Donahue et al., 2019; Payne, 2019), the authors used decoder-only transformer models to generate four-track game music and multi-instrument classical music, respectively. In (Rütte et al., 2022), the authors used

a transformer model to generate multitrack music given a fine-grained description of the characteristics of the desired music. Unlike these systems, our proposed model is built upon a more compact representation that allows it to accommodate longer sequences under the same GPU memory constraint.

Transformers for symbolic music. Another relevant line of research is on modeling symbolic music with transformer models (Vaswani et al., 2017). Some prior work focused on unconditioned generation, including generating piano music (Huang and Yang, 2020; Hsiao et al., 2021), lead sheets (Wu and Yang, 2020) and guitar tabs (Chen et al., 2020b) from scratch. Others studied controllable music generation (Shih et al., 2022; Rütte et al., 2022), polyphonic music score infilling (Chang et al., 2021) and general-purpose pre-training for symbolic music understanding (Chou et al., 2021; Zeng et al., 2021; Wang and Xia, 2021). In this work, we focus on unconditioned generation for evaluation purposes. However, our proposed model can also generate music for a specific set of instruments.

3.3 Proposed Method

3.3.1 Data Representation

We represent a music piece as a sequence of events $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, where each event $\boldsymbol{x}_i$ is encoded as a tuple of six variables:

$$(x_i^{type}, x_i^{beat}, x_i^{position}, x_i^{pitch}, x_i^{duration}, x_i^{instrument}) .$$

The first variable x^{type} determines the type of the event, among the following five event types:

- *Start-of-song*: Indicates the beginning of the song.
- *Instrument*: Specifies an instrument used in the song.
- *Start-of-notes*: Indicates the end of the instrument list and the beginning of the note list. (This event splits the sequence into two parts: a list of instrument events followed by a list of note events, making a trained autoregressive model readily applicable to instrument-informed generation task; see Section 3.3.2.)

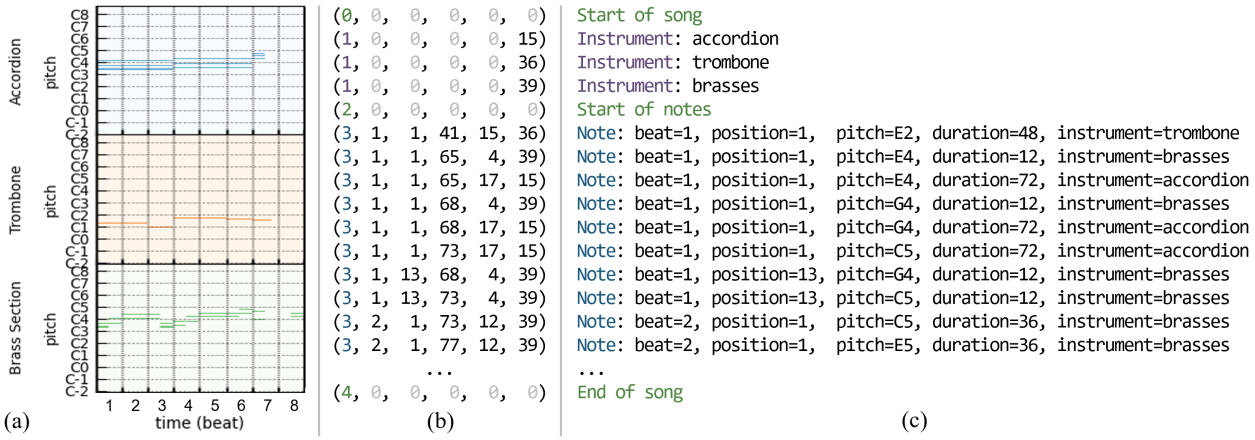


Figure 3.1. An example of the proposed representation—(a) an example of the first eight beats of a song in the orchestra dataset, shown as a multitrack piano roll, (b) the same song encoded by our proposed representation, where the grayed out zeros denote undefined values and (c) a human-readable translation of the codes shown in (b).

- *Note*: Specifies a note, whose onset, pitch, duration and instrument are defined by the other five variables: x^{beat}, $x^{position}$, x^{pitch}, $x^{duration}$ and $x^{instrument}$.
- *End-of-song*: Indicates the end of the song.

For any non-note-type event, the variables x^{beat}, $x^{position}$, x^{pitch}, $x^{duration}$, $x^{instrument}$ are set to zero, which is reserved for undefined values. Figure 3.1 shows an example of our proposed representation.

Following (Huang and Yang, 2020), we decompose the note onset information into beat and position information, where x^{beat} denotes the index of the beat that the note lies in, and $x^{position}$ the position of the note within that beat. To be specific, the actual onset of the note is equivalent to $r \cdot x^{beat} + x^{position}$, where r is the temporal resolution of a beat. For simplicity, we assume that the beats are always a quarter note apart in this work. This decomposition reduces the size of the vocabulary and helps the model learn the music meter system, as evidenced by (Huang and Yang, 2020). For the duration field, following (Rütte et al., 2022), we only allow a carefully-chosen set of common note duration values and replace any duration outside of this set with the closest known duration. For the instrument field, we map similar MIDI programs to the same instrument to reduce the total number of instruments, resulting in 64 unique instruments from the 128 MIDI programs. For example, both 'acoustic grand piano' and 'bright acoustic piano' are mapped to the same 'piano' instrument.

We note that the proposed representation leads to a significantly shorter sequence length as compared to two existing representations (Ens and Pasquier, 2020; Rütte et al., 2022) for multitrack music generation. On an orchestral dataset (Crestel et al., 2017), an encoded sequence of length 1,024 using our propsoed representation can represent 2.6 and 3.5 times longer music samples compared to (Ens and Pasquier, 2020) and (Rütte et al., 2022), respectively. Further, because the timing information is embedded into each note event, the proposed representation is invariant to permutation, i.e., reordering the note events do not affect the decoded music. For the sake of autoregressive training for the transformer model, we sort the notes with respect to the beat field, and subsequently the position, pitch, duration, instrument fields. This allows a trained autoregressive model to be readily applicable to the song continuation task.

3.3.2 Model

We present the Multitrack Music Transformer (MMT) for generating multitrack music using the representation proposed in Section 3.3.1. We base the proposed model on a decoder-only transformer model (Liu et al., 2018b; Brown et al., 2020). Unlike a standard transformer model, whose inputs and outputs are one-dimensional, the proposed model has multi-dimensional input and output spaces similar to (Hsiao et al., 2021), as illustrated in Figure 3.2. The model is trained to minimize the sum of the cross entropy losses of different fields under an autoregressive setting. We adopt a learnable absolute positional embedding (Vaswani et al., 2017). Once the training is done, the trained transformer model can be used in three different modes, depending on the inputs given to the model to start the generation:

- **Unconditioned generation**: Only a 'start-of-song' event is provided to the model. The model generates the instrument list and subsequently the note sequence.
- **Instrument-informed generation**: The model is given a 'start-of-song' event followed by a sequence of instrument codes and a 'start-of-notes' event to start with. The model then generates the note sequence. Note that we need the 'start-of-notes' event as it marks the end of the instrument list, otherwise the model may continue to generate instrument events.

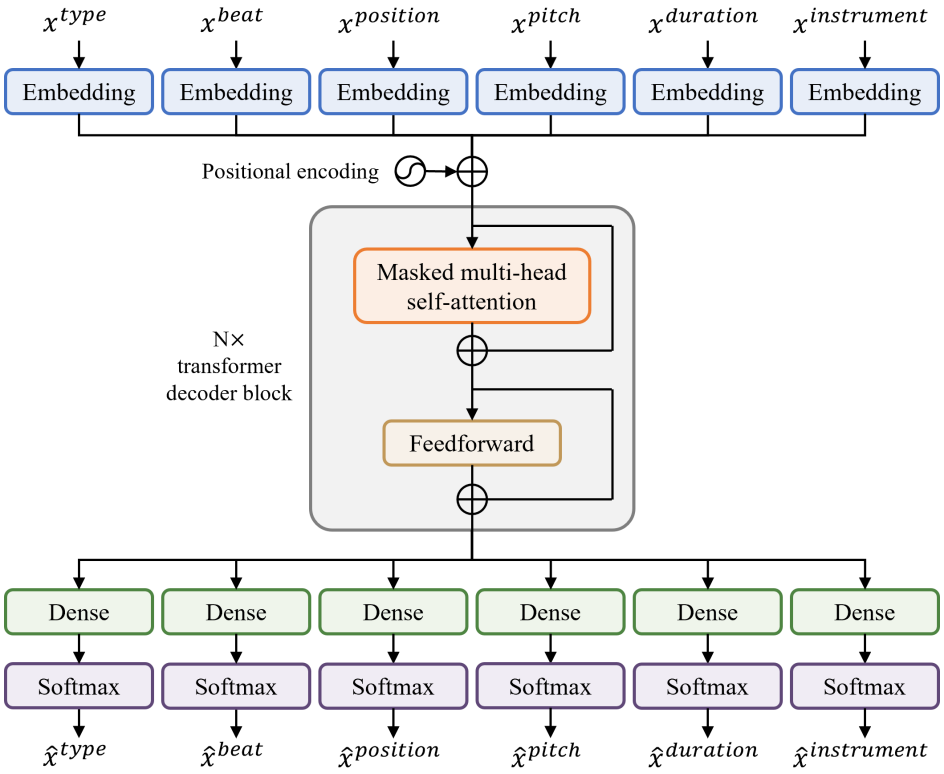


Figure 3.2. Illustration of the proposed MMT model.

- ***N*-beat continuation**: All instrument and note events in the first N beats are provided to the model. The model then generates subsequent note events that continue the input music.

During inference, the sampling process is stopped when an 'end-of-song' event is generated or the maximum sequence length is reached. We adopt the top-k sampling strategy on each field and set k to 10% of the number of possible outcomes per field. Moreover, since the type and beat fields in our representation are always sorted, we further enforce a monotonic constraint during decoding. For example, when sampling for x_{i+1}^{type}, we set the probability of getting a value smaller than x_i^{type} to zero. This prohibits the model from generating events in certain invalid order, e.g., an 'note' event before an 'instrument' event.

Finally, while existing multitrack music generation systems (Ens and Pasquier, 2020; Rütte et al., 2022) need to combine several generated tokens to form a note, the proposed MMT model generates a note at each inference step, i.e., a line in Figure 3.1(b) and (c). This offers MMT a significantly faster inference speed and smaller memory footprint thanks to the

reduced size of the self-attention matrix. However, since MMT predicts the six output fields nonautoregressively (i.e., independently), it cannot model the interdependencies between these fields of the same note. We will discuss this trade-off between time/memory complexity and modeling capacity in Section 3.4.2.

3.4 Results

3.4.1 Experiment Setup

In this work, we consider the Symbolic Orchestral Database (SOD) (Crestel et al., 2017). We set the temporal resolution to 12 time steps per quarter note. We discard tempo and velocity information as not all data contains such information. Further, we discard all drum tracks. We end up with 5,743 songs (357 hours). We reserve 10% of the data for validation and 10% for testing. We use MusPy (Dong et al., 2020) to process the data. For the proposed MMT model, we use 6 transformer decoder blocks, with a model dimension of 512 and 8 self-attention heads. All input embeddings have 512 dimensions. We trim the code sequences to a maximum length of 1,024 and a maximum beat of 256. During training, we augment the data by randomly shifting all the pitches by $s \sim U(-5, 6)$ $(s \in \mathbb{Z})$ semitones and randomly selecting a starting beat. We validate the model every 1K steps and stop the training at 200K steps or when there was no improvements for 20 validation rounds. We render all audio samples using FluidSynth with the MuseScore General SoundFont. We encourage the readers to listen to the sample generated music for the unconditional generation, instrument-informed generation and N-beat continuation tasks on our demo website: https://salu133445.github.io/mmt/.

3.4.2 Subjective Listening Test

To assess the quality of music samples generated by our proposed model, we conducted a listening test with 9 music amateurs recruited from our social networks, where all survey participants can play at least one musical instrument. In the questionnaire, each participant was asked to listen to 10 audio samples generated by each model and rate each audio sample according to three criteria—*coherence*, *richness* and *arrangement*.[1] We com-

[1]To be specific, we ask the following questions: *coherence*—"Is it temporally coherent? Is the rhythm steady? Are there many out-of-context notes?"; *richness*—"Is it rich and diverse in musical textures? Are there any

Table 3.2. Performance comparison of our proposed model against the baseline models. Mean values and 95% confidence intervals are reported.

	Number of parameters	Average sample length (sec)	Inference speed (notes per second)	Subjective listening test results			
				Coherence	Richness	Arrangement	Overall
MMM (Ens and Pasquier, 2020)	19.81 M	38.69	5.66	3.48 ± 0.35	3.05 ± 0.38	3.28 ± 0.37	3.17 ± 0.43
REMI+ (Rütte et al., 2022)	20.72 M	28.69	3.58	**3.90** ± **0.52**	**3.74** ± **0.21**	**3.74** ± **0.44**	**3.77** ± **0.41**
MMT (ours)	19.94 M	**100.42**	**11.79**	3.55 ± 0.46	3.53 ± 0.35	3.40 ± 0.44	3.33 ± 0.47

pared the MMT model against two baseline models based on the standard decoder-only transformer model. The first baseline model used the MultiTrack representation proposed in the MMM model (Ens and Pasquier, 2020), where we replaced the bar tokens with beat tokens. The other used a simplified version of the REMI+ representation used in the FIGARO model (Rütte et al., 2022), where we removed the time signature, tempo and chord tokens as such information is not generally available in our datasets. We will refer to the two baseline models as the MMM and REMI+ models. For a fair comparison, we trimmed all generated samples to a maximum of 64 beats. Moreover, as discussed in Section 3.1, the long sequence length of existing multitrack music representations restricts the model from learning long-term dependencies. Hence, we also computed the mean length of the generated samples and the inference speed in this experiment.

We summarize in Table 3.2 the evaluation results. Compared to the MMM model, our proposed MMT model achieves a higher score across all criteria. Further, MMT generates 2.6 times longer samples and is twice faster in inference speed. As compared to the REMI+ model, our proposed model achieves a mean opinion score (MOS) of 3.33, while the REMI+ model achieves an MOS of 3.77. However, MMT can generate 3.5 times longer samples and is 3.3 times faster in inference speed. This is because the baseline models need multiple inference passes to combine several generated tokens and form a note, whereas the MMT model generate a note in a single inference pass. Finally, we note that while offering a faster inference speed and longer generated sample length, our proposed model cannot model the interdependencies between the six output heads as it predicts each field independently.

repetitions and variations? Is it too boring?"; *arrangement*—"Are the instruments used reasonably? Are the instruments arranged properly?"

Table 3.3. Objective evaluation results. Mean values and 95% confidence intervals are reported. A closer value to that of the ground truth is considered better.

	Pitch class entropy	Scale consistency (%)	Groove consistency (%)
Ground truth	2.974 ± 0.018	92.26 ± 1.25	93.05 ± 1.00
MMM (Ens and Pasquier, 2020)	2.884 ± 0.023	93.13 ± 0.49	91.90 ± 0.64
REMI+ (Rütte et al., 2022)	**2.897 ± 0.019**	**93.12 ± 0.51**	**92.90 ± 0.49**
MMT (ours)	2.802 ± 0.025	94.74 ± 0.42	92.09 ± 0.49

For example, the REMI+ model first generates an instrument token and then generates the pitch token given the instrument token, which allows the model to rule out unsuitable pitches for that particular instrument. In contrast, the MMT model samples from each output head independently. We can clearly observe this trade-off between quality and between time/memory complexity can be clearly observed from Table 3.2.

3.4.3 Objective Evaluation

In addition the subjective listening test, we follow (Mogren, 2016; Wu and Yang, 2020) and measure the pitch class entropy, scale consistency and groove consistency for evaluating the performance of the proposed model on the unconditioned generation task. For these metrics, we consider a closer value to that of the ground truth better. Table 3.3 shows the evaluation results. We can see that the REMI+ model achieves closest values to those of the ground truth. We also notice that while the MMM model result in closer values of pitch class entropy and scale consistency to those of the ground truth, it achieves a lower score in the subjective listening test presented in Section 3.4.2 than our proposed MMT model.

3.4.4 Musical Self-attention

Despite the growing interests in applying transformer models to music, little effort has been made to understand how self-attention works for symbolic music—existing analyses (Huang et al., 2019; Huang et al., 2018; Chen and Su, 2021; Wang and Xia, 2021) provide only case studies on few selected samples. In this section, we aim to investigate musical self-attention in a systematic way. To this end, we propose two new quantities to measure the average relative attention. Mathematically, given a test set $\mathcal{D}$, we define the *mean relative*

attention for a field d (e.g., pitch or beat) as:

$$\gamma_k^{(d)} = \frac{\sum_{\boldsymbol{x}\in\mathcal{D}} \sum_{s>t} a_{s,t}(\boldsymbol{x}) \ \mathbb{1}_{\left(\boldsymbol{x}_t^{(d)} - \boldsymbol{x}_s^{(d)}\right)=k}}{\sum_{k'} \sum_{\boldsymbol{x}\in\mathcal{D}} \sum_{s>t} a_{s,t}(\boldsymbol{x}) \ \mathbb{1}_{\left(\boldsymbol{x}_t^{(d)} - \boldsymbol{x}_s^{(d)}\right)=k'}}, \tag{3.1}$$

where $\mathbb{1}[\cdot]$ is the indicator function and $a_{s,t}(\boldsymbol{x}) \in [0, 1]$ denotes the attention weight assigned by $\boldsymbol{x}_s$ to $\boldsymbol{x}_t$. Intuitively, $\gamma_k^{(d)}$ measures the average attention weight that model assigns to a certain key of a certain difference from the query. Note that each attention head has its own attention weight $a_{s,t}$ and thus its own γ_k. Moreover, we notice that $\gamma_k^{(d)}$ is biased towards differences that occur more frequently. Thus we further propose the *mean relative attention gain*:

$$\tilde{\gamma}_k^{(d)} = \gamma_k^{(d)} - \frac{\sum_{\boldsymbol{x}\in\mathcal{D}} \sum_{s>t} \ \mathbb{1}_{\left(\boldsymbol{x}_t^{(d)} - \boldsymbol{x}_s^{(d)}\right)=k}}{\sum_{k'} \sum_{\boldsymbol{x}\in\mathcal{D}} \sum_{s>t} \ \mathbb{1}_{\left(\boldsymbol{x}_t^{(d)} - \boldsymbol{x}_s^{(d)}\right)=k'}}, \tag{3.2}$$

which measures the difference between $\gamma_k^{(d)}$ and the same quantity obtained by assuming a uniform attention matrix.

In this experiment, we compute $\tilde{\gamma}_k^{beat}$, $\tilde{\gamma}_k^{position}$ and $\tilde{\gamma}_k^{pitch}$ on 100 test samples for the last attention layer of a trained MMT model. As shown in Figure 3.3(a), we can see that the 2nd and 6th attention heads attend more to nearby beats, while the other attention heads attend to beats in further past. In addition, several attention heads assign relatively larger weights to the beats that are $4N$ (i.e., 4, 8, 12, 16, etc.) beats away from the current one, as highlighted by the '$\star$' symbols. From Figure 3.3(b) we observe that the model pays most attention to notes that have the same position as the current note. That is, a note on beat attends more to the last note on beat, and a note off beat attends more to the last note off beat. Figure 3.3(c) shows that the model attends more to pitches within one octave above, and it pays more attention to pitches that form a consonant interval with the current note, e.g., a 4th, a 5th and an octave. We note that the learned self-attention generally comply with music theory principles.

While recent advances in symbolic music generation has borrowed various techniques from natural language modeling, music is fundamentally different from text in that music has a underlying temporal axis embedded and contains strong recurrence patterns

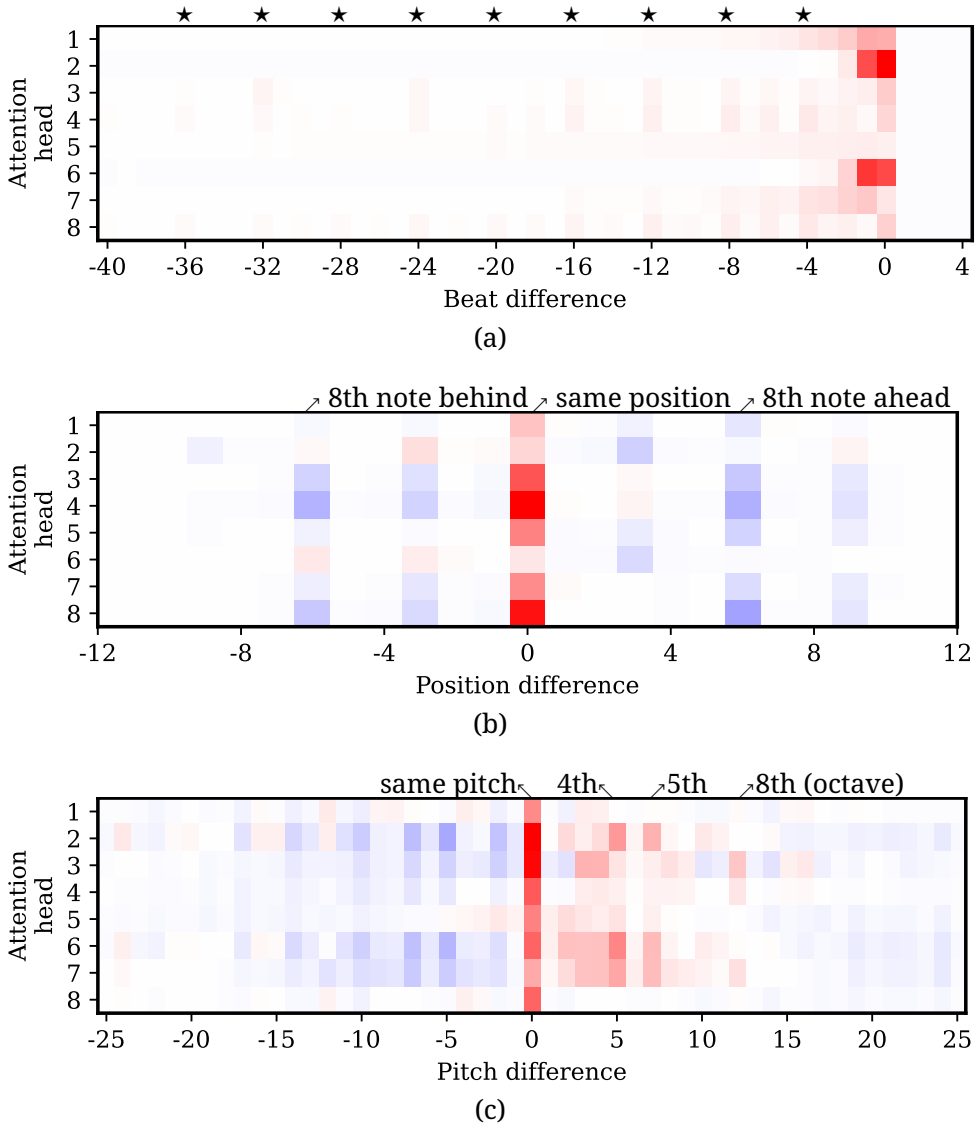


Figure 3.3. Mean relative attention gains (a) $\tilde{\gamma}_k^{beat}$, (b) $\tilde{\gamma}_k^{position}$ and (c) $\tilde{\gamma}_k^{pitch}$ (see Section 3.4.4 for definitions) of a trained MMT model. Red and blue colors indicate positive and negative values, respectively.

in many aspects. Our analysis here shows that our proposed model learns a relative self-attention for certain aspects of music, specifically, beat, position and pitch. We hope our analysis can shed light on further improvements in optimizing the self-attention mechanism for symbolic music modeling.

3.5 Conclusion

We have presented the Multitrack Music Transformer for multitrack music generation. Built upon a new multitrack representation, our proposed model can generate longer multitrack music in a faster inference speed than two existing approaches. We showed in a subjective listening test that the proposed model perform reasonably well against the two baseline models in terms of the quality of the generated music. Through a systematic analysis, we showed that our proposed model learns relative self-attention in certain aspects of music such as beats, positions and pitches. Our findings provide a novel foundation for future work exploring longer-form, real-time capable multitrack music generation and improving the self-attention mechanism for music.

* * *

This chapter, in full, is a reprint of the material as it appears in "Multitrack Music Transformer" by Hao-Wen Dong, Ke Chen, Shlomo Dubnov, Julian McAuley and Taylor Berg-Kirkpatrick, which was published in the Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) in 2023. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Towards Automatic Instrumentation by Learning to Separate Parts in Symbolic Multitrack Music

Abstract

Modern keyboards allow a musician to play multiple instruments at the same time by assigning zones—fixed pitch ranges of the keyboard—to different instruments. In this paper, we aim to further extend this idea and examine the feasibility of automatic instrumentation—dynamically assigning instruments to notes in solo music during performance. In addition to the online, real-time-capable setting for performative use cases, automatic instrumentation can also find applications in assistive composing tools in an offline setting. Due to the lack of paired data of original solo music and their full arrangements, we approach automatic instrumentation by learning to separate parts (e.g., voices, instruments and tracks) from their mixture in symbolic multitrack music, assuming that the mixture is to be played on a keyboard. We frame the task of part separation as a sequential multi-class classification problem and adopt machine learning to map sequences of notes into sequences of part labels. To examine the effectiveness of our proposed models, we conduct a comprehensive empirical evaluation over four diverse datasets of different genres and ensembles—Bach chorales, string quartets, game music and pop music. Our experiments show that the proposed models outperform various baselines. We also demonstrate the potential for our proposed models to

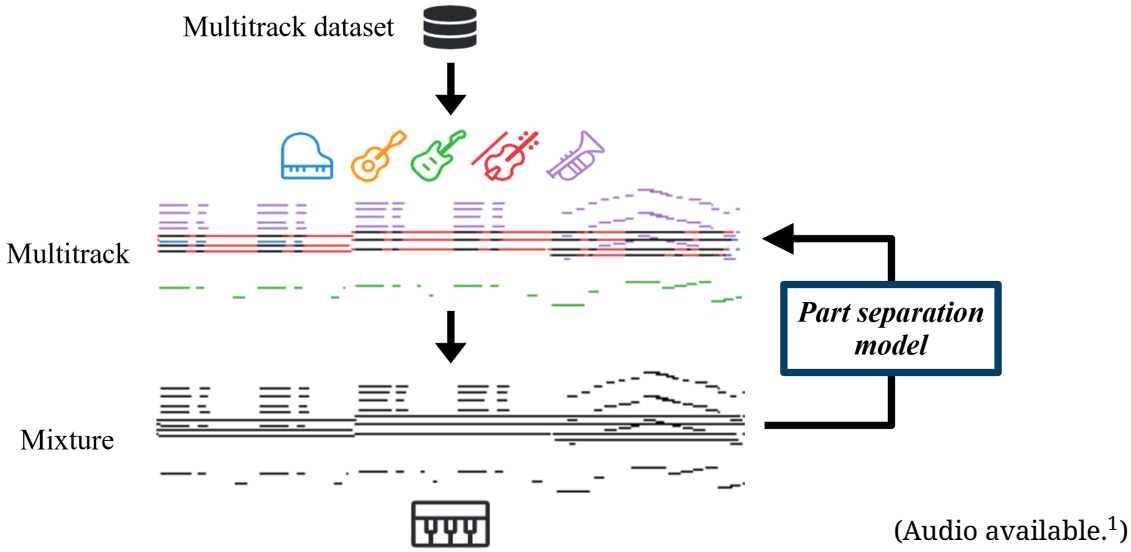
produce alternative convincing instrumentations for an existing arrangement by separating its mixture into parts. All source code and audio samples can be found at https://salu133445.github.io/arranger/.

4.1 Introduction

Music is an art of time and sound. It often contains complex textures and possibly *parts* for multiple voices, instruments and tracks. While jointly following the global style and flow of the song, each part possesses its own characteristics and can develop different musical ideas independently. For example, in pop music, guitar and piano tend to play chords and might span across a large pitch range, while bass is usually monophonic and stays in a lower range. While playing multiple instruments usually requires multiple performers, keyboardists potentially have the ability to control many instruments at once. Modern keyboards often offer the functionality of *zoning*, which allows a player to divide the pitch range into zones and assign each zone to a certain instrument. However, zoning is not ideal given its low flexibility that requires careful configuration and sometimes rearrangement of the music, and incapability for handling certain genres of music that have close and possibly overlapping harmony.

In this paper, we aim for the more ambitious goal of automatic instrumentation—a process that we define as dynamically assigning instruments to notes in solo music. A real-time, online automatic instrumentation model could allow a musician to have their keyboard performance instantaneously and seamlessly performed by a different ensemble. In addition to performative use cases, an offline automatic instrumentation model can also be useful to assist composers in suggesting proper instrumentation or providing a starting point for arranging a solo piece, especially for composers who have less experience arranging for a particular ensemble.

Automatic instrumentation is challenging as it requires domain knowledge of each target instrument, i.e., which pitches, rhythms, chords, and sequences thereof are playable, and it is hard to specify such knowledge by some fixed set of rules. In view of recent advances in machine learning, we propose to adopt a data-driven approach to this task.



(Audio available.[1])

Figure 4.1. Proposed pipeline. By downmixing a symbolic multitrack into a single-track mixture, we acquire paired data of solo music and its instrumentation. We then use these paired data to train a *part separation* model that aims to infer the part label (e.g., one out of the five instruments in this example) for each single note in a mixture. Automatic instrumentation can subsequently be accomplished by treating input from a keyboard player as a downmixed mixture (bottom) and separating out the relevant parts (top). The music is visualized in the piano roll representation, where the x- and y-axes represent time and pitch, respectively. Colors indicate the instruments.

However, it can be laborious to acquire paired data of original solo music and their full arrangements. Given the abundance of multitrack music data, we approach automatic instrumentation by learning to separate parts from their mixture in multitrack music, a task we call *part separation* (see Figure 4.1). Assuming that the mixture is to be played on a keyboard and the multitrack is the target arrangement that we want to generate by an automatic instrumentation model, we thereby have paired data for solo music and full arrangements.

We frame the new task of part separation as a sequential multi-class classification problem that aims to map sequences of notes into sequences of part labels. We adopt long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Transformer (Hsiao et al., 2021) models for the task. We conduct an extensive empirical evaluation showing the superiority of our proposed models to baselines for the related task of voice separation as well as strategies found in commodity keyboards. To showcase the potential of our proposed models, we also demonstrate their ability to produce alternative convincing instrumentations for existing arrangements. Audio for all examples and more samples are

available on the demo website.[1] All source code can be found in the project repository.[2]

4.2 Prior Work

Voice separation is a related task to part separation which involves separating blended scores into individual monophonic voices. While useful, voice separation is agnostic to constraints imposed by specific instruments—a composer using a voice separation algorithm would have to manually align voices to appropriate instruments. Some prior work investigates voice separation in small, carefully-annotated pop music datasets (Gray and Bunescu, 2016; Guiomard-Kagan et al., 2015). Some prior work on voice separation allows synchronous or overlapping notes in a voice (Kilian and Hoos, 2002; Cambouropoulos, 2006; Karydis et al., 2007; Cambouropoulos, 2008). However, their results are only reported on small test sets in certain genres. Others have adopted multilayer perceptrons (Gray and Bunescu, 2016; Valk and Weyde, 2018) and convolutional neural networks (Gray and Bunescu, 2020) with hand-crafted input features for voice separation. Another relevant work on hand detection in piano music used LSTMs to separate notes played by right and left hands in piano MIDI data (Hadjakos et al., 2019). To the best of our knowledge, no past work has examined the task of part separation in a general setting for multiple music genres.

In addition to voice separation, prior work has explored automatic music arrangement. The primary focus of prior work for automatic music arrangement has been on reduction—mapping musical scores for large ensembles to parts playable by a single specific instrument such as the piano (Chiu et al., 2009; Onuma and Hamanaka, 2010; Huang et al., 2012; Nakamura and Sagayama, 2015; Takamori et al., 2017; Nakamura and Yoshii, 2018), guitar (Tuohy and Potter, 2005; Hori et al., 2012; Hori et al., 2013) or bass (Abe et al., 2012). This past work focuses on identifying the least important notes to delete so that the resultant score is playable on a single instrument, whereas our work seeks to preserve the original score in its entirety and satisfy playability for multiple instruments simultaneously. As an exception, Crestel and Esling (Crestel and Esling, 2016) explore strategies for arranging orchestral music from piano, though their approach does not guarantee that all notes in the

[1] https://salu133445.github.io/arranger/
[2] https://github.com/salu133445/arranger

input piano map to parts in the output.

Music generation is another body of work that has used neural network sequential models for processing symbolic music (Briot et al., 2017). Simon and Oore (Simon and Oore, 2017) proposed a convenient approach for music generation which involved training recurrent neural network language models on a language-like "event-based" representation of music. Subsequently, recent work has explored event-based representations using Transformers (Huang et al., 2019; Payne, 2019; Donahue et al., 2019; Huang and Yang, 2020; Ens and Pasquier, 2020; Hsiao et al., 2021; Muhamed et al., 2021). In this work, we explore a more compact input representation of music that passes all of the information about a note into the model at once, rather than spreading it out across several events. We also note that Payne (Payne, 2019) generate music which contains parts for several instruments, but their model cannot be directly used to perform part separation of existing musical material.

4.3 Problem Formulation

Mathematically, we consider a piece of music x as a sequence of notes $(x_1, \ldots, x_N)$, where N is the number of notes. Each note is represented by a tuple of time t_i and pitch p_i, i.e., $x_i = (t_i, p_i)$. Alternatively, we could also include duration d_i as an input and have $x_i = (t_i, p_i, d_i)$. Each note is associated with a label $y_i \in \{1, \ldots, K\}$ that represents the part it is in, where K is the number of parts. The goal of part separation is to learn the mapping between notes and part labels. This formulation is rather flexible and has no assumptions on whether a part is monophonic or not—it can be a voice, an instrument, a track, etc.

In terms of the context given for predicting the label of each note, we can categorize part separation models into three classes: An *independent model* predicts the label for each note independently, without any context. An *online model* predicts the label of the current note x_i given only past information, i.e., notes $(x_1, \ldots, x_{i-1})$, as context. An *offline model* predicts the label of the current note x_i given past and future information, i.e., the full sequence of $(x_1, \ldots, x_N)$, as context. While independent and online models are preferable for use cases that require real-time outputs, e.g., live performance. Moreover, the inability to look into the future makes the real-time setting more challenging than the offline setting. On the other hand, offline models can find applications in assisstive composing tools.

4.4 Models

We consider the following input features for our models—(1) *time*: onset time, in time step,[3] (2) *pitch*: pitch as a MIDI note number, (3) *duration*: note length, in time step, and (4) *frequency*: fundamental frequency of the pitch, in Hz, computed by the formula $f = 440 \cdot 2^{(p-69)/12}$. In addition, we also consider features that encode the metric time grid of music similar to the BAR and POSITION events proposed in (Huang and Yang, 2020)—(5) *beat*: onset time, in beat, and (6) *position*: position within a beat, in time step.

Moreover, to help the models better disambiguate parts, we also include two simple hints—(7) *entry hints*: onset position for each instrument, encoded as a unit step function centered at its onset time and all zero if the instrument is not used, and (8) *pitch hints*: average pitch of each track. These hints allow the musician to use interactively to make the instrumentation process more controllable. For example, entry hints can be used to control the instruments available as they serve as switches for the instruments.

For the machine learning models, we consider the LSTM (Hochreiter and Schmidhuber, 1997) and its bidirectional version (BiLSTM) (Schuster and Paliwal, 1997). We use a three-layer stacked LSTM with 128 hidden units in each layer (64 hidden units per layer for BiLSTM). We also consider two variants of Transformer (Vaswani et al., 2017)—one based on the encoder (Transformer-Enc) and one based on the decoder (Transformer-Dec). They share the same architecture that is composed of three Transformer blocks, each of which has 128 hidden units and 8 heads in self-attention computation and 256 hidden units in the internal feedforward network. However, they have different attention masks: Transformer-Enc uses only the padding mask, while Transformer-Dec uses both the padding mask and the lookahead mask, which blocks its access to future information and makes it a online model. In this paper, the LSTM and Transformer-Dec models are made online models, and the BiLSTM and Transformer-Enc models are made offline models that take durations as inputs.

[3] Assuming that the music is in metrical timing, a time step is a factor of some musically-meaningful unit (e.g., a quarter note) and can be adjusted to match the desired temporal resolution.

4.5 Baseline Models

In order to gain an insight into how the proposed models perform, we include two heuristic algorithms and a voice separation model from the literature in our empirical study.

4.5.1 Zone-based algorithm

This algorithm simulates a common feature in modern keyboards where a player can preassign a pitch range (i.e., the 'zone') for each instrument and notes will automatically be assigned to the corresponding instrument as the player performs. This algorithm finds the optimal zones for the whole training data and uses these optimal zones at test time. For the oracle case, the optimal zones for each sample are computed and used at test time. We note that the oracle case might not be easily achievable as it can be hard for a musician to set the zones optimally beforehand, especially for improvisation.

4.5.2 Closest-pitch algorithm

The closest-pitch algorithm keeps track of the last active pitches p'_i for each track i. For each incoming pitch p, it finds the pitch among the last active pitches that has the closest pitch to p and assigns the upcoming note with the same label as the chosen pitch. This is a casual model and it also relies on the onset hints. We can formulate this algorithm as follows. For $i = 1, \ldots, N$, we have

$$\hat{y}_i = \begin{cases} y_i, & \text{if } x_i \text{ is an onset} \\ \underset{j \in \{1,\ldots,K\}}{\arg\min} (p_i - p'_j)^2 + M a_i, & \text{otherwise} \end{cases},$$

where p'_i is the last active pitch of track i before time t and a_i indicates whether track i is active, i.e., a concurrent note has not yet been released. We set M to a large positive number when we assume each part is monophonic, which we will refer to as the 'mono' version of this algorithm, otherwise set $M = 0$.

Table 4.1. Statistics of the four datasets considered in this paper.

Dataset	Hours	Files	Notes	Parts	Ensemble	Most common label
Bach chorales (Cuthbert and Ariza, 2010)	3.23	409	96.6K	4	soprano, alto, tenor, bass	bass (27.05%)
String quartets (Thickstun et al., 2017)	6.31	57	226K	4	first violin, second violin, viola, cello	first violin (38.72%)
Game music (Donahue et al., 2018)	45.05	4.61K	2.46M	3	pulse wave I, pulse wave II, triangle wave	pulse wave II (39.35%)
Pop music (Raffel, 2016)	1.02K	16.2K	63.6M	5	piano, guitar, bass, strings, brass	guitar (42.50%)

4.5.3 Multilayer perceptron (MLP)

We adapt the voice separation model proposed in (Valk and Weyde, 2018) to the task of part separation. This model uses multilayer perceptron (MLP) to predict the label for the current note based on hand-crafted features that encodes its nearby context. We use entry hints rather than predicting them by the proposed voice entry estimation heuristics. We remove the 'interval' feature as there is no upper bound for the number of concurrent notes and change the proximity function to L1 distance. The oracle case of this model replaces error-prone prior predictions with ground truth history labels. In our implementation, we use three fully-connected layers with 128 hidden units each.

4.6 Data

In order to examine the effectiveness of the proposed models, we consider four datasets—(1) *Bach chorales* in Music21 (Cuthbert and Ariza, 2010), (2) *string quartets* in MusicNet (Thickstun et al., 2017), (3) *game music* in Nintendo Entertainment System (NES) Music Database (Donahue et al., 2018) and (4) *pop music* in Lakh MIDI Dataset (Raffel, 2016), which are diverse in their genres, sizes and ensembles (see Table 4.1 for a comparison).

As these datasets are noisy in different ways, we need to further clean the data. For the game music dataset, we discard the percussive noise track in the original dataset as they do not follow the standard 128-pitch system used in other tracks. For the pop music dataset, we use a cleaned subset derived in (Dong et al., 2018a), which contains only pop songs. We mapped the instruments to the five most common instrument families—piano, guitar, bass, strings and brass. We follow the General MIDI 1 specification on the mapping from an instrument to its instrument family. Instruments that fall outside of these five families are discarded. We note that the lead melody track might occasionally be discarded during the mapping process due to the high variance on instruments used for the melody track.

Moreover, we discard songs with only one active track as the task becomes trivial in this case. We note that all Bach chorales, string quartets and most pop songs are in metrical timing, where a time step corresponds to some fraction of a quarter note. Thus, we downsample them into 24 time steps per quarter note, which can cover 32nd notes and triplets. As songs in the game music dataset are in absolute time, we downsample them to a temporal resolution equivalent to 24 time steps per quarter note in a tempo of 125 quarter notes per minute (qpm).

Finally, we split each dataset into train–test–validation sets with a ratio of $8:1:1$ except the game music dataset, where we use the original splits provided with the NES Music Database. We use MusPy (Dong et al., 2020) and music21 (Cuthbert and Ariza, 2010) for processing MIDI and MusicXML files.

4.7 Experiments

4.7.1 Implementation details

We use a batch size of 16, a sequence length of 500 for training and a maximum sequence length of 2000 for validation and testing. We clip the time by 4096 time steps (i.e., roughly 170 quarter notes), the beat by 4096 beats, and durations by 192 time steps (i.e., 8 quarter notes). We randomly transpose the music by -5 to +6 semitones during training for data augmentation. We use the cross entropy loss with the Adam optimizer with $\alpha = 0.001$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ (Kingma and Ba, 2015). We apply dropout (Srivastava et al., 2014) to

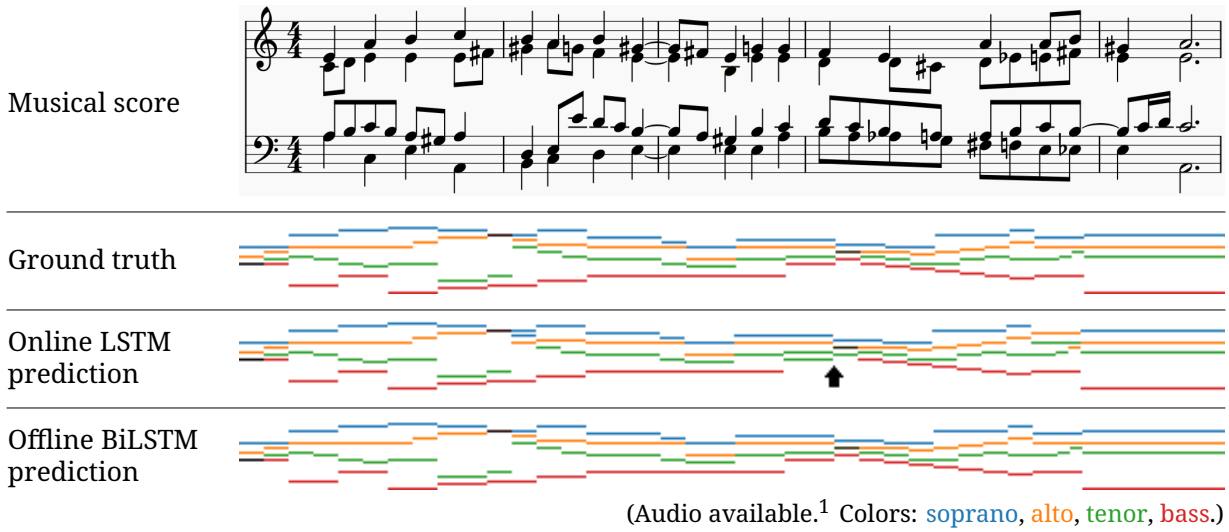


Figure 4.2. Example of the Bach chorales dataset—*Wer nur den lieben Gott läßt walten*, BWV 434, measures 1–5. The LSTM model makes two errors for the bass, as indicated by the arrow. The BiLSTM model gives a perfect prediction.

prevent overfitting and layer normalization (Ba et al., 2016) to speed up the training. All models are implemented in TensorFlow (Abadi et al., 2016) and experiments are run on NVIDIA GeForce RTX 2070s.

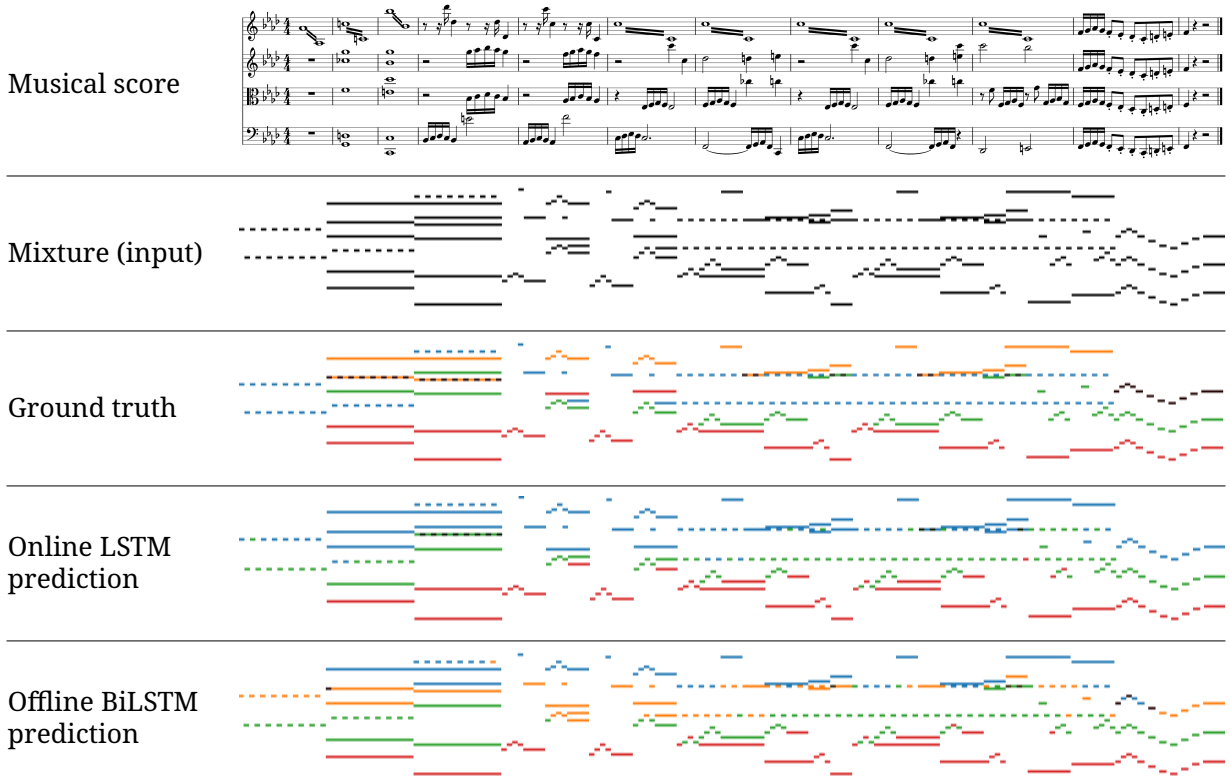
4.7.2 Qualitative results and error analysis

We present in Figures 4.2 to 4.5 several examples in the four datasets. Some representative cases include overlapping pitch ranges or chords for two polyphonic instruments (see Figures 4.3 and 4.5), overlapping melodies and chords (see Figure 4.5) and a sequence of short notes crossing a single long note (see Figure 4.4).

4.7.3 Quantitative results

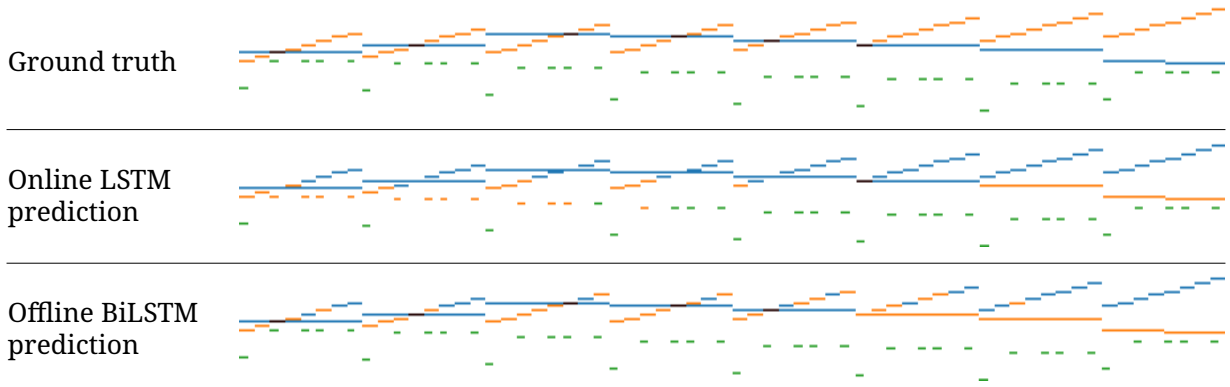
We conduct an extensive empirical evaluation over different dataset and models in different settings. We present the results in Table 4.2.[4] First, we notice the improved performance for the oracle cases on the MLP baseline. The large gap of performance is possibly because it predicts each note independently and the errors can propagate over time. This emphasizes the need to incorporate sequential models for this task. Moreover, the BiLSTM model outperforms its LSTM counterpart for most cases. This is reasonable

[4]Due to high computation cost, we report the oracle cases for the zone-based algorithm and MLP model on a subset of 100 test samples, and omit the oracle case of the zone-based algorithm for the pop music dataset.



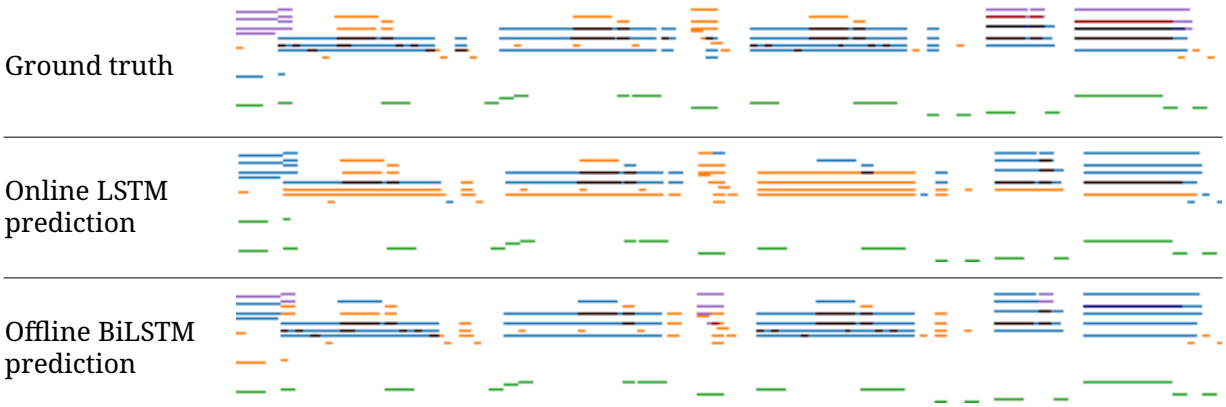
(Audio available.[1] Colors: first violin, second violin, viola, cello.)

Figure 4.3. Hard excerpt in the string quartets dataset—Beethoven's *String Quartet No. 11 in F minor, Op. 95*, movement 1, measures 72–83. The tremolos of the first violin (measures 1–3 and 6–10), the double stops for the second violin, viola and cello (measures 2–3) and the overlapping pitch ranges (measures 2–5) together compose a complex texture. Both models fail to handle the violins and viola properly, especially the second violin.



(Audio available.[1] Colors: pulse wave I, pulse wave II, triangle wave.)

Figure 4.4. Hard excerpt in the game music dataset—*Theme of Universe* from *Miracle Ropit's Adventure in 2100*. Both models perform poorly when there is a sequence of short notes crossing a single long note.



(Audio available.[1] Colors: piano, guitar, bass, strings, brass.)

Figure 4.5. Hard excerpt in the pop music dataset—*Blame It On the Boogie* by The Jacksons. The BiLSTM model correctly identify and separate the overlapping guitar melody and piano chords, while the LSTM model fails in this case.

as the BiLSTM model has access to the future information, which could, for example, help identify the direction of an arpeggio. Further, the LSTM and BiLSTM models outperform their Transformer counterparts—Transformer-Dec and Transformer-Enc, respectively—across all settings. However, the Transformer models benefits from faster inference speed at test time as compared to the LSTM models. Finally, we notice that the proposed models perform relatively poorly on the string quartets and game music datasets, possibly because the two violins in the string quartets dataset and the two pulse waves in the game music dataset are sometimes used interchangeably. We examine the use of pitch hints to help the models in the following section.

4.7.4 Effect of input features

In order to compare the effectiveness of different input features, we also report in Table 4.3 the performance for the LSTM model with different input features. First of all, pitch, note and beat embedding leads to improvements on all datasets, especially significant on the string quartet (30% gain) and pop music (15% gain) datasets. Second, entry hints improve the performance by 10% for the game music dataset, which is possibly because it helps disambiguate the two interchangeable pulse wave tracks. Interestingly, they have negative impacts on the Bach chorales and pop music datasets. Third, duration inputs are always helpful and help achieve the highest accuracy on the Bach chorales and pop music

Table 4.2. Comparison of our proposed models and baseline algorithms. Performance is measured in accuracy (%).

Model	Bach	String	Game	Pop
Online models				
Zone-based	73.14	58.85	43.67	57.07
MLP (Valk and Weyde, 2018)	81.63	29.85	43.08*	33.50*
LSTM	**93.02**	**67.43**	**50.22**	**74.14**
Transformer-Dec	91.51	57.03	45.82	62.14
Zone-based (oracle)	78.33	66.89	79.54*	†
MLP (Valk and Weyde, 2018) (oracle)	97.59	58.16	65.30	44.62
Offline models				
BiLSTM	**97.13**	**74.38**	**52.93**	**77.23**
Transformer-Enc	96.81	58.86	49.14	66.57
Online models (+entry hints)				
Closest-pitch	68.87	50.69	57.14	47.45
Closest-pitch (mono)	89.76	42.82	49.91	32.28
LSTM	**92.70**	**62.64**	**62.11**	**74.19**
Transformer-Dec	91.17	62.12	56.73	67.19
Offline models (+entry hints)				
BiLSTM	**97.39**	**71.51**	**64.79**	**75.59**
Transformer-Enc	93.81	56.72	54.67	67.23

*Reported on a subset of 100 test samples due to high computation cost.
†Omitted due to high computation cost.

Table 4.3. Effects of input features to the online LSTM model. Performance is measured in accuracy (%). Abbreviations: 'Emb'—pitch, beat and position embedding, 'Dur'—duration, 'EH'—entry hints, 'PH'—pitch hints.

Emb	Dur	EH	PH	Bach	String	Game	Pop
				92.10	37.29	43.89	58.78
✓				93.02	67.43	50.22	74.14
✓	✓			**96.17**	66.96	51.38	**78.17**
✓		✓		92.70	62.64	62.11	74.19
✓	✓	✓		95.95	68.17	63.35	74.74
✓			✓	92.87	**70.20**	**67.45**	75.89

Table 4.4. Comparisons of time encoding and data augmentation strategies for the online LSTM model. Performance is measured in accuracy (%).

Strategy	Bach	String	Game	Pop
Time encoding				
Raw time	91.97	37.26	44.10	37.92
Raw beat and position	**93.13**	66.72	48.60	68.42
Time embedding	92.21	**68.31**	49.32	70.64
Beat and position emb.	93.02	67.43	**50.22**	**74.14**
Data augmentation				
No augmentation	**93.03**	**69.36**	49.03	70.73
Light augmentation	92.85	68.66	46.38	71.10
Strong augmentation	93.02	67.43	**50.22**	**74.14**

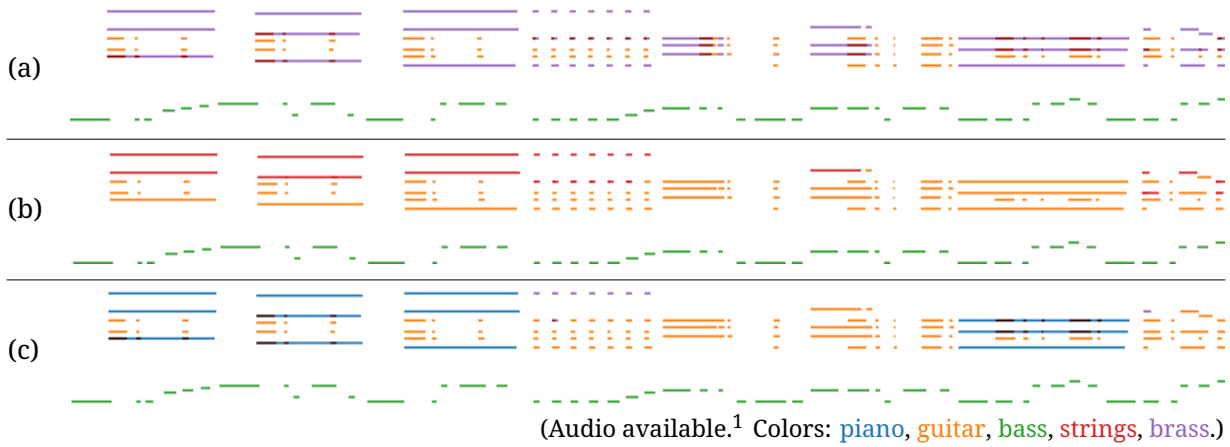
datasets. For example, durations would be critical in distinguishing the overlapping guitar melody and piano chords in the example shown in Figure 4.5. Last, pitch hints improve the performance for all datasets but Bach chorales, possibly because the vocal ranges for SATB are strict in chorales. Pitch hints help achieve the highest accuracies for the string quartets and game music datasets as they help disambiguate interchangeable tracks.

4.7.5 Effects of time encoding

In this experiment, we examine the effects of time encoding. In particular, we consider four variants—(1) raw time as a number, (2) raw beat and position as two numbers, (3) time embedding and (4) beat and position embedding (see Section 4.4 for the definition of beat and position). We report in Table 4.4 the results and we can see that using raw time gives the worst performance. Interestingly, the other three encoding strategies achieve comparable performance.

4.7.6 Effects of data augmentation

In this experiment, we compare the following three strategies of data augmentation—(1) no augmentation, (2) *light augmentation*, where each song is randomly transposed by -1 to +1 semitone during training and (3) *strong augmentation*, where each song is randomly transposed by -5 to +6 semitones during training. We report in Table 4.4 the results. We can see that data augmentation is generally harmful for the Bach chorales and string quartets datasets, possibly because classical music has strict rules on the pitch ranges of voices and



(Audio available.[1] Colors: piano, guitar, bass, strings, brass.)

Figure 4.6. *Quando Quando Quando* by Tony Renis—(a) original instrumentation and the versions produced by (b) the online LSTM model without entry hints and (b) the offline BiLSTM model with entry hints. The LSTM model assigns the chords to the guitar, the most common instrument in the pop music dataset except the high pitches, which are assigned to the strings. The BiLSTM model is able to separate the long chords from the short ones and assigns the former to the piano.

instruments. However, for game and pop music datasets, where rules on keys and pitch ranges in classical music are loosened, the models yield better performance with proper data augmentation.

4.8 Discussion

In Figure 4.6, we depict the original instrumentation of the song *Quando Quando Quando* alongside the instrumentations generated by our best performing models for both the online and offline settings. While neither model produces an instrumentation identical to that of the original, both produce instrumentations that "cluster" notes similarly to the original and are reasonable rearrangements of the song. This indicates a fundamental ambiguity of the task, though we note that such ambiguity is less present in some genres than others—our models are able to achieve high accuracy on the Bach chorales dataset despite its small size. However, for larger and more diverse datasets (e.g., the pop music dataset), accuracy might not be the best metric for measuring the performance of the models, and we plan to include human evaluations in future work.

One limitation of this work lies in the generalizability to real keyboard music since the downmixed music might not be playable on a keyboard, e.g., having more than ten concurrent notes or impossible fingering. Moreover, we did not use the MIDI velocity

information in our models, and it could provide an additional signal for separation.

Finally, in addition to its immediate musical applications, we believe that our proposed part separation task may be useful for large-scale pre-training of symbolic music models. Pre-training music generation models on large, heterogeneous music corpora has already been observed to improve performance (Donahue et al., 2019; Hung et al., 2019). Given that our proposed task represents an additional source of musical knowledge supervision, we speculate that additionally pre-training on this task could improve performance for many downstream tasks, e.g., genre classification and melody extraction.

4.9 Conclusion

In this paper, we have proposed a new task of part separation in multitrack music and examined its feasibility under both the online and offline settings. Through a comprehensive empirical evaluation over four diverse datasets, we showed the effectiveness of our proposed models against various baselines. We also presented promising results for applying part separation models to automatic instrumentation. Moreover, we discussed the fundamental ambiguity and limitations of the task and future research directions.

* * *

This chapter, in full, is a reprint of the material as it appears in "Towards Automatic Instrumentation by Learning to Separate Parts in Symbolic Multitrack Music" by Hao-Wen Dong, Chris Donahue, Taylor Berg-Kirkpatrick and Julian McAuley, which was published in the Proceedings of the International Society for Music Information Retrieval Conference (ISMIR) in 2021. The dissertation author was the primary investigator and author of this paper.

Chapter 5

Deep Performer: Score-to-Audio Music Performance Synthesis

Abstract

Music performance synthesis aims to synthesize a musical score into a natural performance. In this paper, we borrow recent advances in text-to-speech synthesis and present the Deep Performer—a novel system for score-to-audio music performance synthesis. Unlike speech, music often contains polyphony and long notes. Hence, we propose two new techniques for handling polyphonic inputs and providing a fine-grained conditioning in a transformer encoder-decoder model. To train our proposed system, we present a new violin dataset consisting of paired recordings and scores along with estimated alignments between them. We show that our proposed model can synthesize music with clear polyphony and harmonic structures. In a listening test, we achieve competitive quality against the baseline model, a conditional generative audio model, in terms of pitch accuracy, timbre and noise level. Moreover, our proposed model significantly outperforms the baseline on an existing piano dataset in overall quality.

5.1 Introduction

Music synthesis is a complex process that involves both the physics behind a musical instrument and the art of music performance. It remains challenging for a machine to synthesize a natural performance for several reasons. First, it requires a computational

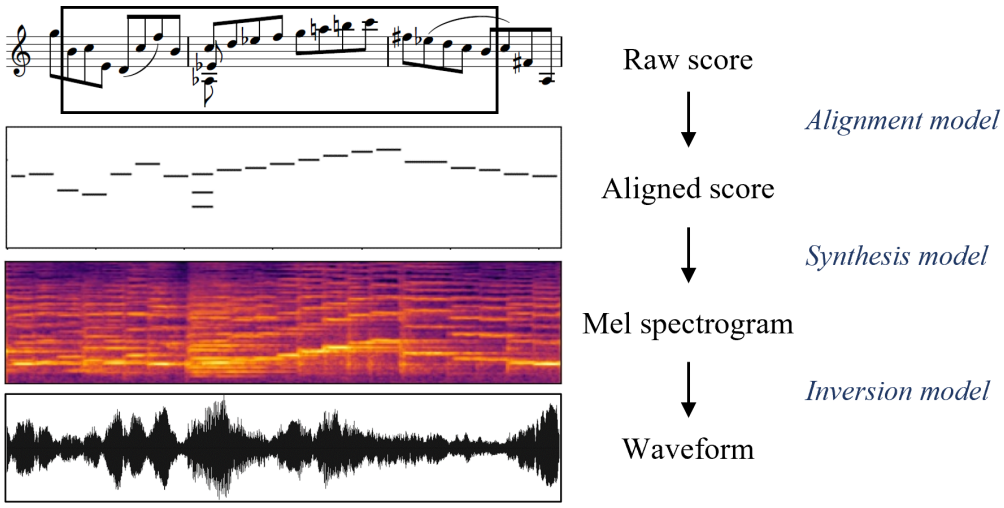


Figure 5.1. An overview of the proposed three-stage pipeline for score-to-audio music performance synthesis.

model for interpreting and phrasing a musical score. Second, it requires either an explicit or implicit model of the physics and acoustics by which a musical instrument sounds. Third, it requires an understanding of different playing techniques and styles for a musical instrument. While most existing systems only address one of these three challenges at a time, we aim to tackle all these challenges with a data-driven approach using machine learning in this work. We present the Deep Performer—a novel three-stage system for score-to-audio music synthesis, as illustrated in Figure 5.1.

Prior work has studied music synthesis via various approaches. One line of research focuses on generating realistic samples of musical notes (Engel et al., 2017; Défossez et al., 2018; Engel et al., 2019), while in this work we aim to generate the full performance. Some approach music synthesis by conditioning generative audio models with aligned piano rolls (Manzelli et al., 2018; Hawthorne et al., 2019), which we will include as the baseline model in our experiments. Others study synthesizing audio from the fundamental frequency (F0) contour and loudness curve extracted from a recording (Engel et al., 2020; Hayes et al., 2021), or from lyrics and demo singing audio (Ren et al., 2020). On the other hand, some use neural networks to generate expressive timing and dynamics from raw scores (Oore et al., 2020). Many have also studied inverting mel spectrograms back to waveforms (Shen et al., 2018; Prenger et al., 2019; Kumar et al., 2019; Kong et al., 2020a), including Hifi-GAN (Kong et al., 2020a), which we will use as the inversion model in our proposed system. To the best of our

knowledge, prior work on deep neural network based music synthesis either requires an input with expressive timing (Manzelli et al., 2018; Wang and Yang, 2019; Hawthorne et al., 2019; Schimbinschi et al., 2019; Kim et al., 2019; Ren et al., 2020; Engel et al., 2020; Hayes et al., 2021) or allows only monophonic (i.e., one pitch at a time) inputs (Ren et al., 2020; Engel et al., 2020; Wu et al., 2022b). Our proposed system represents the first that allows unaligned, polyphonic scores as inputs.

In light of the similarity between text-to-speech (TTS) and score-to-audio synthesis, we borrow recent advances from TTS synthesis (Tan et al., 2021) to music synthesis and propose a three-stage system for score-to-audio music synthesis. Despite the similarity, music synthesis differs from speech synthesis in that music often contains polyphony, and that long notes are common in music. In order to handle polyphonic music, we propose a new *polyphonic mixer* for aligning the encoder and decoder in a transformer encoder-decoder network (Vaswani et al., 2017; Wang et al., 2017). To provide a fine-grained conditioning to the model, we propose a new *note-wise positional encoding* so that the model can learn to behave differently at the beginning, middle and end of a note. Due to the lack of a proper dataset for training a score-to-audio music synthesis model, we collect and release a new dataset of 6.5 hours of high-quality violin recordings along with their scores and estimated alignments. Through our experiments, we show the effectiveness of our proposed system both qualitatively and quantitatively. Finally, we conduct a subjective listening test to compare our proposed model against a baseline model that uses Hifi-GAN (Kong et al., 2020a) to synthesize the waveform directly from an aligned piano roll. Audio samples can be found on our project website.[1]

5.2 Methods

We illustrate in Figure 5.1 the proposed three-stage system for score-to-audio music synthesis, which consists of the following three components: (1) an *alignment model* that predicts the expressive timing for each note from a musical score, (2) a *synthesis model* that synthesizes the mel spectrogram from the aligned score, and (3) an *inversion model* that generates the audio waveform given the synthesized mel spectrogram.

[1] https://salu133445.github.io/deepperformer/

5.2.1 Alignment model

The alignment model consists of a transformer encoder that takes as inputs a sequence of notes and the tempo, followed by a fully-connected layer that outputs the onset and duration of each note. The input score uses metric time with a musically-meaningful unit, e.g., quarter notes, while the output alignment is in the unit of frames. Each note is specified by its pitch, onset, duration and (optional) velocity. In addition, we provide the performer IDs so that the model can learn the different playing styles of performers. The alignment model is trained to minimize the mean squared error (MSE) between the ground truth and predicted onsets and durations, in frames.

5.2.2 Synthesis model

Given the similarity between TTS and score-to-audio synthesis, we propose a transformer encoder-decoder model for our synthesis model based on (Ren et al., 2019). In (Ren et al., 2019), each text embedding produced by the encoder is expanded multiple times according to its duration, and then the expanded text embeddings are concatenated to obtain the frame embeddings to be fed to the decoder. This is called the state expansion mechanism (Ren et al., 2019; Yu et al., 2020). However, unlike speech, music often contains polyphony. In order to handle polyphonic inputs, we propose the *polyphonic mixer*. As illustrated in Figure 5.2, the encoder first encodes the input notes into a sequence of note embeddings. Then, the polyphonic mixer mixes the note embeddings into a sequence of frame embeddings by summing up the note embeddings for the same frame according to their onsets and durations. Finally, the decoder decodes the frame embeddings into a sequence of mel spectrogram frames.

In the state expansion mechanism (Ren et al., 2019; Yu et al., 2020), the output vectors remain constant for the duration of a note, and the positional information within each note is missing. However, we argue that such note-wise positional information is critical for the model to behave differently at the beginning, middle and end of a note. Hence, we propose the *note-wise positional encoding* to provide a fine-grained conditioning to the decoder. Mathematically, let $p \in [0, 1]$ be the relative position within a note. For a note embedding $\mathbf{v}_{\text{note}}$, we have the corresponding frame embedding at position p as $\mathbf{v}_{\text{frame}} = (1 + p\mathbf{w}) \odot \mathbf{v}_{\text{note}}$,

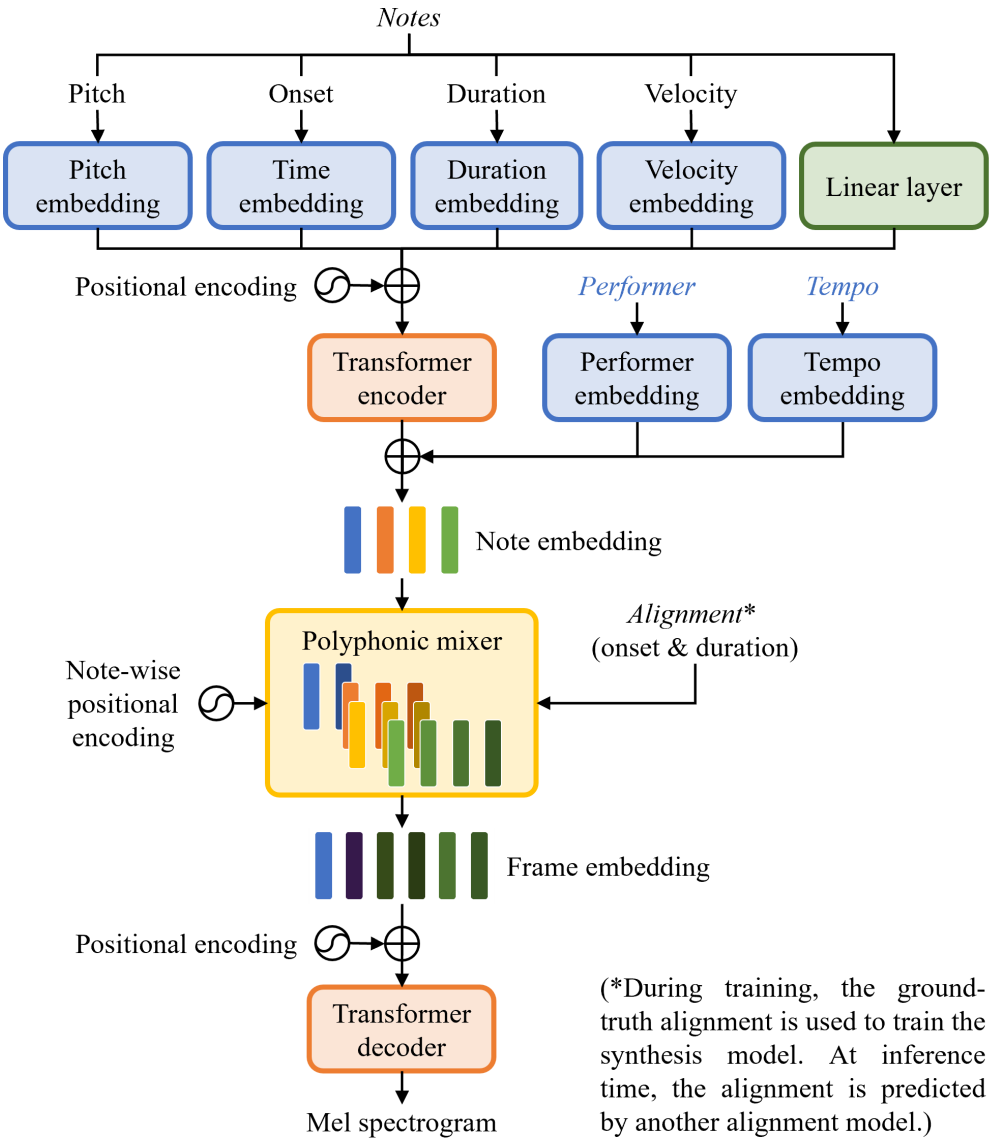


Figure 5.2. An illustration of the proposed synthesis model.

where $\mathbf{w}$ is a learnable vector initialized to small random numbers so that $\mathbf{v}_{\text{frame}} \approx \mathbf{v}_{\text{note}}$ initially. The synthesis model is trained to minimize the MSE between the synthesized mel spectrograms and the ground truth, in log scale.

5.2.3 Inversion model

Prior work has studied various approaches for synthesizing waveforms from mel spectrograms (Shen et al., 2018; Prenger et al., 2019; Kumar et al., 2019; Kong et al., 2020a). In this work, we adopt the state-of-the-art Hifi-GAN model (Kong et al., 2020a) as our inversion model. We note that the proposed three-stage pipeline allows us to use different datasets for training the models. For example, training the inversion model does not require aligned data and thus it can be trained on a larger dataset as unaligned data are relatively easier to acquire.

5.3 Data

Due to the lack of a dataset that provides paired audios and scores with fine alignments for training our proposed system, we compile a new dataset of 6.5 hours of professional violin recordings along with their scores and estimated alignments. For copyright concern, we choose Bach's sonatas and partitas for solo violin (BWV 1001–1006) for the ease to acquire high-quality public recordings from the web. The dataset consists of performances by 17 violinists recorded in various recording setups. To acquire the alignment between a recording and its score, we synthesize the scores using FluidSynth (*FluidSynth* n.d.), an open-source software synthesizer, with MuseScore General SoundFont (*MuseScore General SoundFont* n.d.) and perform dynamic time warping on the constant-Q spectrogram of the synthesized audio and that of the recording. We present in Figure 5.3 an example of the dataset and its estimated alignment. To facilitate future research on score-to-audio music synthesis, we release the dataset and the source code for the alignment process to the public.[2] As discussed in Section 5.2.3, the inversion model does not require aligned data for training, and thus we also collect an internal dataset of 156 hours of commercial recordings to train the inversion model. Apart from violin, we also consider the MAESTRO dataset (Hawthorne et al.,

[2] https://salu133445.github.io/bach-violin-dataset/

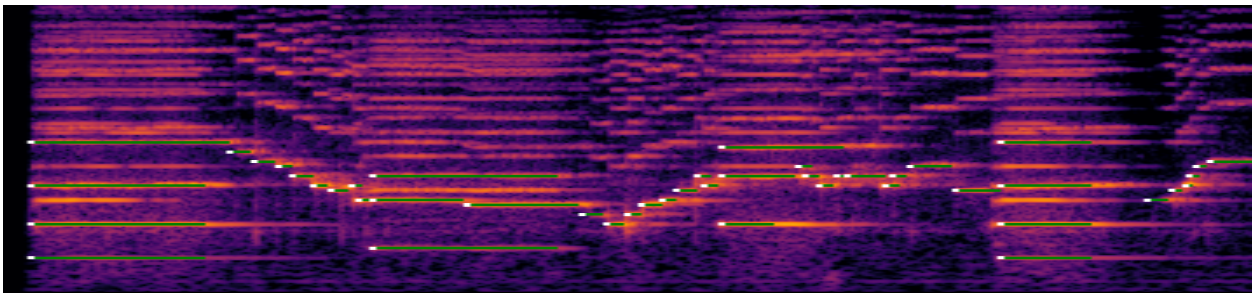


Figure 5.3. An example of the constant-Q spectrogram of the first 20 seconds of a violin recording and the estimated onsets (white dots) and durations (green lines).

2019), which contains 200 hours of piano recordings with finely-aligned MIDI recordings for 10 competition years of the International Piano-e-Competition (*International Piano-e-Competition* n.d.). However, since it does not provide the raw scores, we can only train the synthesis and inversion models on this dataset.

5.4 Experiments & Results

5.4.1 Implementation details

We use 3 transformer layers in the encoder for the alignment model. The synthesis model shares the same encoder architecture as the alignment model and has 6 transformer layers in the decoder. We use 128 dimensions for all embeddings. For the inversion model, we use the same network architecture as the Hifi-GAN v2 model in (Kong et al., 2020a). We use velocity information only for the piano dataset as it is only available in this dataset. Since performer information is unavailable for the piano dataset, we use the competition years as the performer IDs. We use a temporal resolution of 24 time steps per quarter note for the scores. We downsample the audios to 16 kHz mono and use a hop size of 256 in spectrogram computation, i.e., a temporal resolution of 16 ms. The audios are sliced into 5-second clips for training, where 10% of them are reserved for validation purpose. We use the Adam optimizer (Kingma and Ba, 2015) with a batch size of 16. Unlike (Ren et al., 2019), we train the alignment and synthesis models separately as we find that joint training hinders convergence. We train the alignment model for 10K steps and all the synthesis models for 100K (violin) and 250K (piano) steps. For each dataset, the inversion model is trained for 1M steps and shared by different synthesis models. We base our implementation on the code

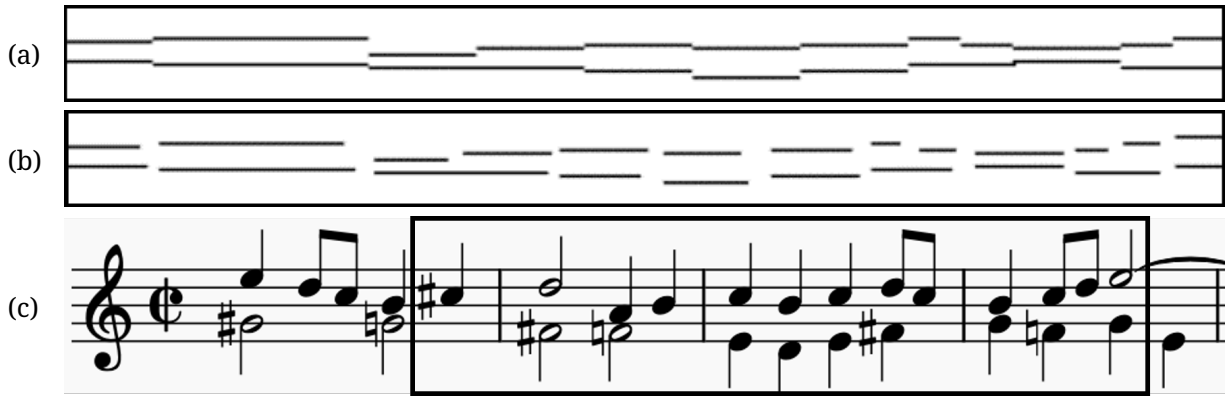


Figure 5.4. Examples of the alignments predicted by (a) the constant-tempo baseline model and (b) Deep Performer, our proposed model. (c) shows the input score.

kindly released in (Chien et al., 2021; Kong et al., 2020a). We use pretty_midi (Raffel and Ellis, 2014) and MusPy (Dong et al., 2020) to process the scores.

5.4.2 Qualitative and quantitative results

We show in Figure 5.4 an example of the alignment predicted by our proposed alignment model alongside that generated by assuming a constant tempo. We can see that our proposed model is able to predict realistic timing and insert rests between notes. To showcase the effectiveness of the proposed polyphonic mixer, we present in Figure 5.5 examples of the synthesized mel spectrograms for two polyphonic scores, where we can observe clear harmonic structures and polyphony.

Next, we compare our proposed synthesis model against a baseline model that uses a Hifi-GAN (Kong et al., 2020a) to synthesize the waveform directly from an aligned piano roll. For a fair comparison, we condition this model with the performer IDs and provide a *position roll* that encodes the note-wise position information. (A position roll is similar to a piano roll, but the values decrease linearly from 1 to 0, from the beginning of a note to its end.) As can be seen from Figure 5.6(a) and (b), our proposed model produces smoother contours and clearer harmonic structures, especially on the high frequency end, while the baseline model generates sharper yet noisier results. Table 5.1 shows the final MSE between the synthesized mel spectrograms and the ground truths. We can see that our proposed model achieves a lower MSE than the baseline model on both datasets. Finally, due to the reduced temporal resolution of a mel spectrogram compared to that of a waveform, our

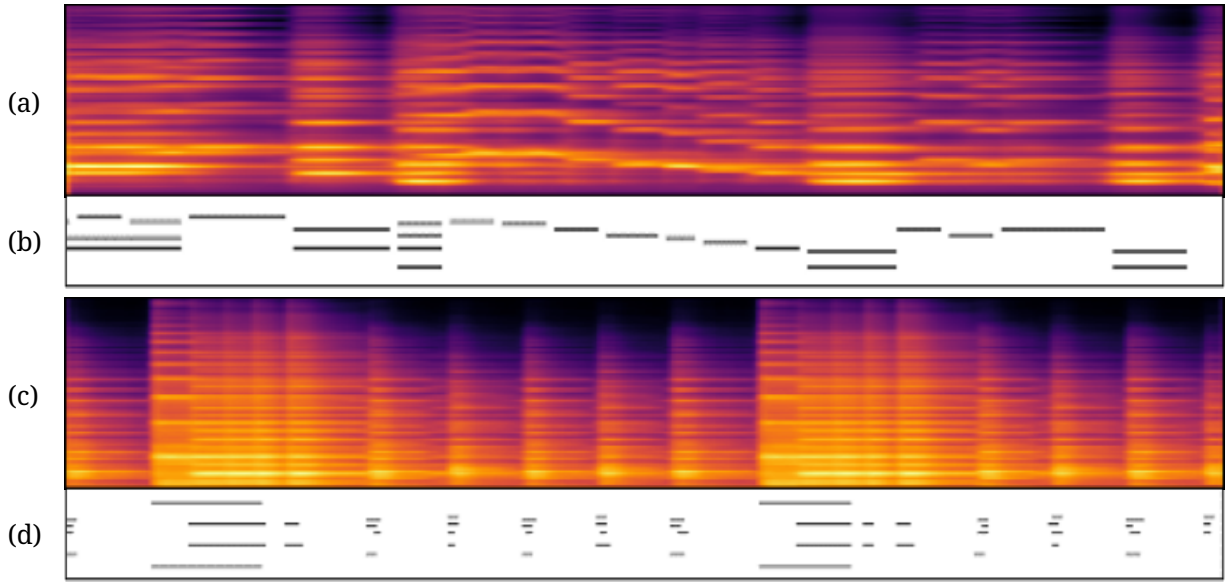


Figure 5.5. Examples of the mel spectrograms, in log scale, synthesized by our proposed model for (a) violin and (c) piano. (b) and (d) show the input scores for (a) and (c), respectively.

Table 5.1. Comparisons of the final MSE between the synthesized mel spectrograms and the ground truths, in log scales.

	Violin	Piano
Hifi-GAN baseline	0.892	0.722
Deep Performer (ours)	0.700	0.436
- without note-wise positional encoding	0.700	0.433
- without performer embedding	1.030	0.523
- without encoder (using piano roll input)	0.844	0.621

proposed model is faster in training than the baseline model. Audio samples can be found on our project website.[1]

5.4.3 Subjective listening test

To further evaluate our proposed system, we conduct a subjective listening test with 15 participants recruited from our social networks, where 14 of them plays a musical instrument. We randomly choose 5 musical scores from each dataset and synthesize them with different models. The participants are instructed to rate the synthesized audios in a 5-point Likert scale in terms of pitch accuracy, timbre and noise level as well as the overall quality. We report the results in Table 5.2. We can see that our proposed model significantly outperforms the baseline model on the piano dataset and achieves comparable performance

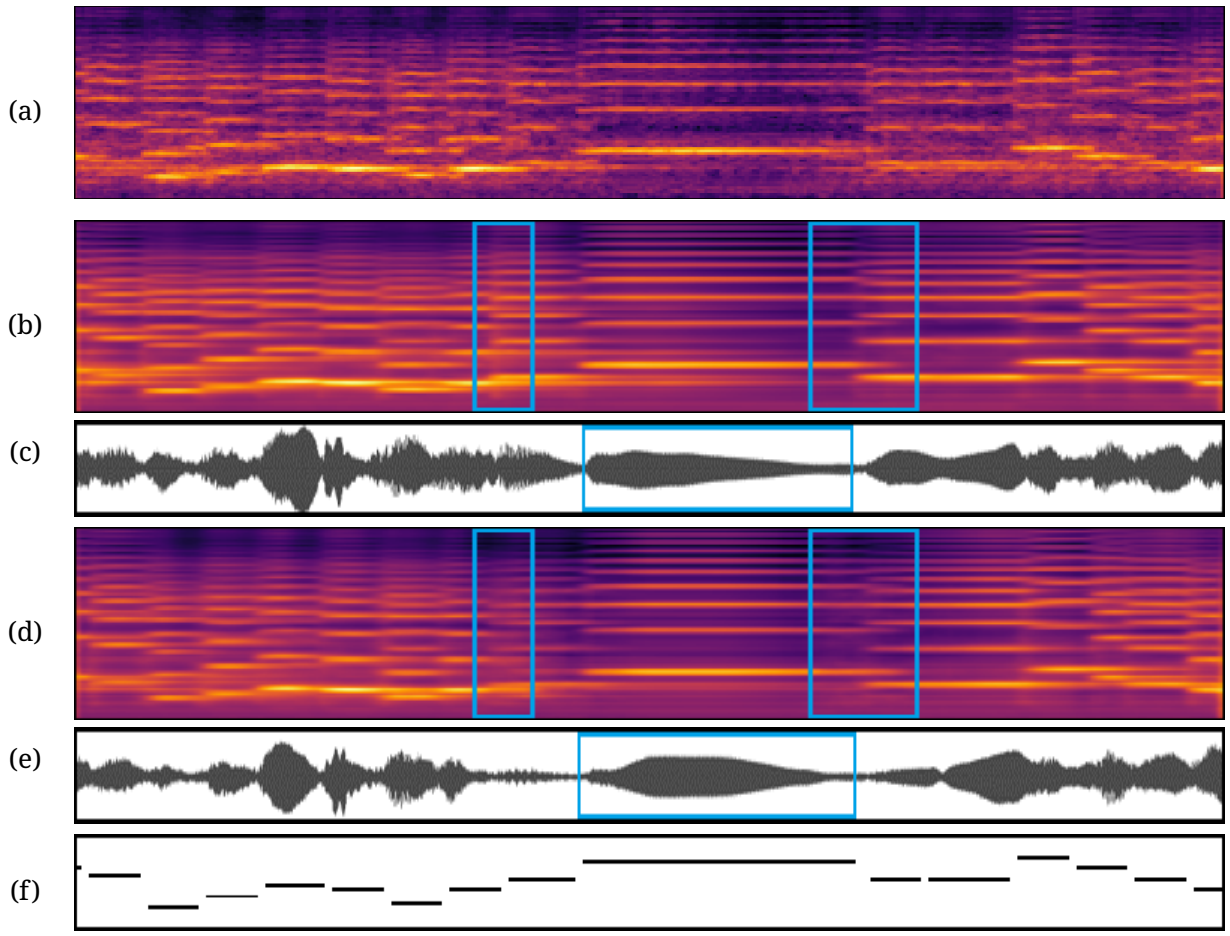


Figure 5.6. Examples of the mel spectrograms, in log scale, synthesized by (a) the baseline model, (b) our proposed synthesis model, and (d) our proposed synthesis model without the note-wise positional encoding. (c) and (e) show the waveforms for (b) and (d), respectively. (f) shows the input score.

Table 5.2. Results of the subjective listening test. The mean opinion scores (MOS) and 95% confidence intervals are reported.

	Violin				Piano
	Pitch accuracy	Timbre	Noise level	Overall	Overall
Hifi-GAN baseline	4.02 ± 0.31	3.13 ± 0.26	2.51 ± 0.29	2.57 ± 0.22	1.49 ± 0.17
Deep Performer (ours)	4.22 ± 0.30	3.26 ± 0.30	2.67 ± 0.31	2.58 ± 0.21	2.17 ± 0.24
- without note-wise positional encoding	4.13 ± 0.29	3.24 ± 0.27	2.52 ± 0.29	2.61 ± 0.23	2.37 ± 0.23
- without performer embedding	3.05 ± 0.52	2.54 ± 0.42	2.04 ± 0.31	2.01 ± 0.25	2.26 ± 0.25
- without encoder (using piano roll input)	4.30 ± 0.36	2.91 ± 0.28	2.39 ± 0.28	2.22 ± 0.18	1.43 ± 0.16

to the baseline on the violin dataset.

5.4.4 Ablation study

To measure the contributions of different components of the proposed model, we consider three ablated versions of our model. The first removes the note-wise positional encoding. The second removes the performer embedding. The third removes the encoder and uses piano rolls and position rolls (see Section 5.4.2) as the inputs to the decoder, while keeping the performer embedding. As we can see from Figure 5.6(b)–(e), note-wise positional encoding help the model produce clearer note transitions and a more realistic waveform envelope (see the highlighted regions). We also report in Tables 5.1 and 5.2 the results for these ablated models. We can see that the performer embedding significantly improves the quality across all criteria. While we show above the effectiveness of the note-wise positional encoding, its impact does not reach statistical significance in our subjective listening test, possibly overshadowed by the artifacts produced by the models. Finally, including an encoder network improves the quality significantly, suggesting that the encoder can learn a more effective representation of the score as compared to the piano roll representation.

5.5 Conclusion

We presented a novel three-stage system for synthesizing natural music performance from unaligned musical scores. We proposed the polyphonic mixer for aligning the encoder and decoder with polyphonic inputs. In addition, we also proposed the note-wise positional encoding for providing a fined-grained conditioning to the synthesis model. Through the subjective listening test, we show that our proposed model significantly outperforms the baseline model on the piano dataset and achieves competitive quality against the baseline on the violin dataset. For future work, we plan to utilize the articulation marks and ornaments on scores to better model playing techniques (Yang et al., 2016; Shih et al., 2017), disentangle the timbre from room acoustics to enhance controllability (Engel et al., 2020), and incorporate adversarial losses (Isola et al., 2017; Yang et al., 2021) to improve the sharpness of the results.

* * *

This chapter, in full, is a reprint of the material as it appears in "Deep Performer: Score-to-Audio Music Performance Synthesis" by Hao-Wen Dong, Cong Zhou, Taylor Berg-Kirkpatrick and Julian McAuley, which was published in the Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) in 2022. The dissertation author was the primary investigator and author of this paper.

Appendices

5.A Preprocessing details

We downmix the recordings to mono and downsample them to 16 kHz using FFmpeg. We then convert them into mel spectrograms using librosa. For the mel spectrogram computation, we use a filter length of 1024, a hop length of 256 and a window size of 1024 in the short-time Fourier transform (STFT), and we use 80 mel bands in mel scale conversion. We summarize these parameters in Table 5.3.

Table 5.3. Preprocessing parameters

Parameter	Value
Audio channels	mono
Sampling rate	16 kHz
STFT filter length	1024
STFT hop length	256
STFT window size	1024
Mel bands	80

5.B Network architectures

5.B.1 Alignment model

We illustrate the proposed alignment model in Figure 5.7. We use 128 dimensions for all embeddings. For the transformer encoder, we use 3 transformer layers, each consisting of a multi-head attention (MHA) and a position-wise feed-forward network (FFN) sub-layer. We use 64 hidden neurons and 2 attention heads for each MHA layer. For each FFN layer, we use 256 hidden neurons with kernel sizes of 9 and 1 for the two convolutional layers. Further, we use a maximum sequence length of 1000 and clip the time and duration to 96. We summarize these hyperparameters in Table 5.4.

5.B.2 Synthesis model

For the synthesis model, we use 128 dimensions for all embeddings. For the transformer model, we use 3 and 6 transformer layers for the encoder and decoder, respectively. We use 128 hidden neurons and 2 attention heads for each MHA layer. For each FFN layer,

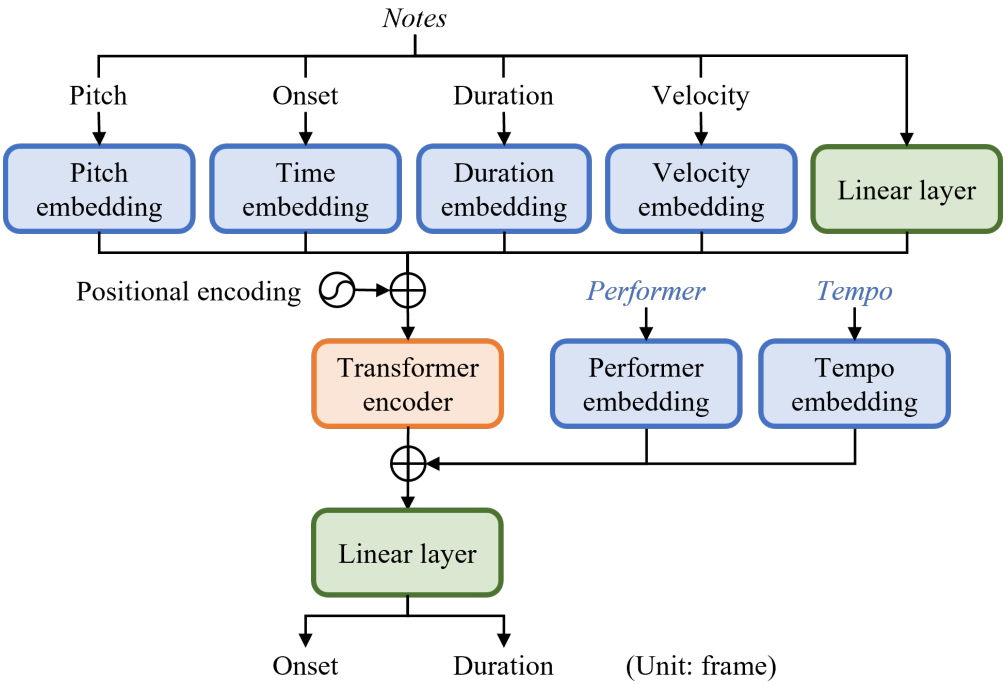


Figure 5.7. An illustration of the proposed alignment model.

Table 5.4. Alignment model architecture

Parameter	Value
Encoder layers	3
MHA heads	2
MHA hidden neurons	64
FFN hidden neurons	256
FFN kernel sizes	9, 1
Max sequence length	1000
Max time	96
Max duration	96

we use 256 hidden neurons with kernel sizes of 9 and 1 for the two convolutional layers. In addition, we use a maximum sequence length of 1000. We also clip the time and duration to 96 and 100 for the violin and piano datasets, respectively. We summarize these hyperparameters in Table 5.5. We base our implementation on the source code kindly provided in (Chien et al., 2021).[3]

Table 5.5. Synthesis model architecture

Parameter	Value
Encoder layers	3
Decoder layers	6
MHA heads	2
MHA hidden neurons	128
FFN hidden neurons	512
FFN kernel sizes	9, 1
Max sequence length	1000
Max time	96*
Max duration	96*

*100 for the piano dataset

5.B.3 Inversion model

For the inversion model, we use the network architecture of the Hifi-GAN v2 model proposed in (Kong et al., 2020a). We base our implementation on the source code kindly provided in (Kong et al., 2020a).[4]

5.B.4 Baseline model

We base the baseline model on the same Hifi-GAN v2 model (Kong et al., 2020a). In addition, we include an additional linear layer that maps the input piano roll to a hidden vector whose dimension matches the input dimension of the Hifi-GAN v2 model. Further, we include an additional embedding layer to condition the baseline model on the input performer IDs. The outputs of these two layers are summed up and fed as the input to the Hifi-GAN v2 model.

[3]https://github.com/ming024/FastSpeech2
[4]https://github.com/jik876/hifi-gan

5.C Training details

We use a batch size of 16 and apply a dropout rate of 0.2 after each sub-layer. We use the same optimizer settings as the original implementation of transformer (Vaswani et al., 2017). For the alignment model, we apply the learning rate annealing schedule used in (Chien et al., 2021). We summarize these hyperparameters in Table 5.6. Unlike (Ren et al., 2019), we train the alignment and synthesis models separately as we find that joint training hinders convergence. For the violin dataset, we train the alignment, synthesis and inversion models for 10K, 100K and 1M steps, respectively. For the piano dataset, we train the synthesis and inversion models for 250K and 1M steps, respectively. For each dataset, the inversion model is trained once and used with different synthesis models.

Table 5.6. Training hyperparameters

Parameter	Value
Batch size	16
Dropout	0.2
Adam optimizer β_1	0.9
Adam optimizer β_2	0.98
Adam optimizer ϵ	10^{-9}
Gradient clipping threshold	1.0
Warm up steps (alignment model)	1000
Warm up steps (synthesis model)	4000
Learning rate annealing steps*	10K, 20K, 50K
Learning rate annealing rate*	0.5

*Applied to the alignment model only

Chapter 6

CLIPSep: Learning Text-queried Sound Separation with Noisy Unlabeled Videos

Abstract

Recent years have seen progress beyond domain-specific sound separation for speech or music towards universal sound separation for arbitrary sounds. Prior work on universal sound separation has investigated separating a target sound out of an audio mixture given a text query. Such text-queried sound separation systems provide a natural and scalable interface for specifying arbitrary target sounds. However, supervised text-queried sound separation systems require costly labeled audio-text pairs for training. Moreover, the audio provided in existing datasets is often recorded in a controlled environment, causing a considerable generalization gap to noisy audio in the wild. In this work, we aim to approach text-queried universal sound separation by using only unlabeled data. We propose to leverage the visual modality as a bridge to learn the desired audio-textual correspondence. The proposed CLIPSep model first encodes the input query into a query vector using the contrastive language-image pretraining (CLIP) model, and the query vector is then used to condition an audio separation model to separate out the target sound. While the model is trained on image-audio pairs extracted from unlabeled videos, at test time we can instead query the model with text inputs in a zero-shot setting, thanks to the joint language-image embedding learned by the CLIP model. Further, videos in the wild often contain off-screen sounds

and background noise that may hinder the model from learning the desired audio-textual correspondence. To address this problem, we further propose an approach called *noise invariant training* for training a query-based sound separation model on noisy data. Experimental results show that the proposed models successfully learn text-queried universal sound separation using only noisy unlabeled videos, even achieving competitive performance against a supervised model in some settings.

6.1 Introduction

Humans can focus on to a specific sound in the environment and describe it using language. Such abilities are learned using multiple modalities—auditory for selective listening, vision for learning the concepts of sounding objects, and language for describing the objects or scenes for communication. In machine listening, selective listening is often cast as the problem of sound separation, which aims to separate sound sources from an audio mixture (Cherry, 1953; Bach and Jordan, 2005). While text queries offer a natural interface for humans to specify the target sound to separate from a mixture (Liu et al., 2022; Kilgour et al., 2022), training a text-queried sound separation model in a supervised manner requires labeled audio-text paired data of single-source recordings of a vast number of sound types, which can be costly to acquire. Moreover, such isolated sounds are often recorded in controlled environments and have a considerable domain gap to recordings in the wild, which usually contain arbitrary noise and reverberations. In contrast, humans often leverage the visual modality to assist learning the sounds of various objects (Baillargeon, 2002). For instance, by observing a dog barking, a human can associate the sound with the dog, and can separately learn that the animal is called a "*dog*." Further, such learning is possible even if the sound is observed in a noisy environment, e.g., when a car is passing by or someone is talking nearby, where humans can still associate the barking sound solely with the dog. Prior work in psychophysics also suggests the intertwined cognition of vision and hearing (Sekuler et al., 1997; Shimojo and Shams, 2001; Rahne et al., 2007).

Motivated by this observation, we aim to tackle text-queried sound separation using

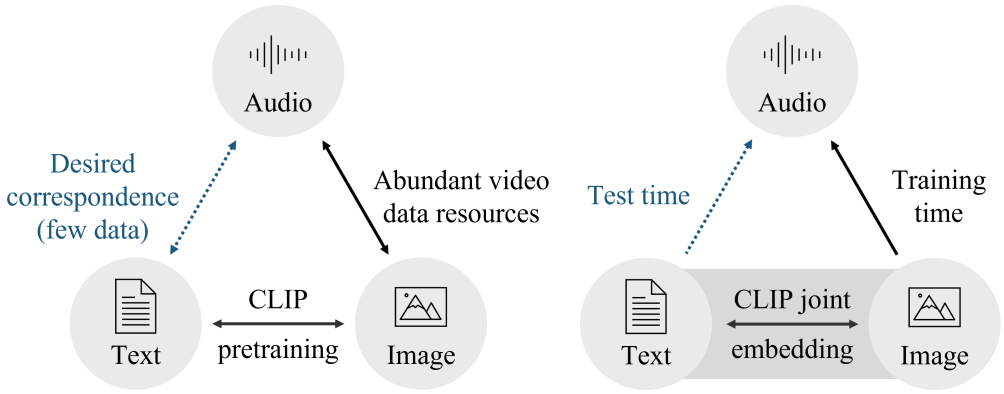


Figure 6.1. An illustration of modality transfer.

only unlabeled videos in the wild. We propose a text-queried sound separation model called CLIPSep that leverages abundant unlabeled video data resources by utilizing the contrastive image-language pretraining (CLIP) (Radford et al., 2021) model to bridge the audio and text modalities. As illustrated in Figure 6.1, during training, the image feature extracted from a video frame by the CLIP-image encoder is used to condition a sound separation model, and the model is trained to separate the sound that corresponds to the image query in a self-supervised setting. Thanks to the properties of the CLIP model, which projects corresponding text and images to close embeddings, at test time we instead use the text feature obtained by the CLIP-text encoder from a text query in a zero-shot setting.

However, such zero-shot modality transfer can be challenging when we use videos in the wild for training as they often contain off-screen sounds and voice overs that can lead to undesired audio-visual associations. To address this problem, we propose the *noise invariant training* (NIT), where query-based separation heads and permutation invariant separation heads jointly estimate the noisy target sounds. We validate in our experiments that the proposed noise invariant training reduces the zero-shot modality transfer gap when the model is trained on a noisy dataset, sometimes achieving competitive results against a fully supervised text-queried sound separation system.

Our contributions can be summarized as follows: 1) We propose the first text-queried universal sound separation model that can be trained on unlabeled videos. 2) We propose a new approach called *noise invariant training* for training a query-based sound separation

model on noisy data in the wild. Audio samples can be found on an our demo website.[1] For reproducibility, all source code, hyperparameters and pretrained models are available at: https://github.com/sony/CLIPSep.

6.2 Related Work

Universal sound separation. Much prior work on sound separation focuses on separating sounds for a specific domain such as speech (Wang and Chen, 2018) or music (Takahashi and Mitsufuji, 2021; Mitsufuji et al., 2021). Recent advances in domain specific sound separation lead several attempts to generalize to arbitrary sound classes. Kavalerov et al. (2019) reported successful results on separating arbitrary sounds with a fixed number of sources by adopting the *permutation invariant training* (PIT) (Yu et al., 2017), which was originally proposed for speech separation. While this approach does not require labeled data for training, a post-selection process is required as we cannot not tell what sounds are included in each separated result. Follow-up work (Ochiai et al., 2020; Kong et al., 2020b) addressed this issue by conditioning the separation model with a class label to specify the target sound in a supervised setting. However, these approaches still require labeled data for training, and the interface for selecting the target class becomes cumbersome when we need a large number of classes to handle open-domain data. Wisdom et al. (2020) later proposed an unsupervised method called mixture invariant training (MixIT) for learning sound separation on noisy data. MixIT is designed to separate all sources at a time and also requires a post-selection process such as using a pre-trained sound classifier (Scott et al., 2021), which requires labeled data for training, to identify the target sounds. We summarize and compare related work in Table 6.1.

Query-based sound separation. Visual information has been used for selecting the target sound in speech (Ephrat et al., 2019; Afouras et al., 2020), music (Zhao et al., 2018; Zhao et al., 2019; Tian et al., 2021) and universal sounds (Owens and Efros, 2018; Gao et al., 2018; Rouditchenko et al., 2019). While many image-queried sound separation approaches require clean video data that contains isolated sources, Tzinis et al. (2021) introduced an unsupervised method called AudioScope for separating on-screen sounds using noisy videos

[1] https://sony.github.io/CLIPSep/

Table 6.1. Comparisons of related work in sound separation. '(✓)' indicates that the problem of noisy training data is partially addressed. CLIPSep-NIT denotes the proposed CLIPSep model trained with the noise invariant training. To the best of our knowledge, no previous work has attempted the problem of label-free text-queried sound separation.

Method	Query type	Unlabeled data	Noisy data
USS (Kavalerov et al., 2019)	×	✓	
MixIT (Wisdom et al., 2020)	×	✓	✓
Universal Sound Selector (Ochiai et al., 2020)	Label		
USS-Label (Kong et al., 2020b)	Label		(✓)
PixelPlayer (Zhao et al., 2018)	Image	✓	(✓)
AudioScope (Tzinis et al., 2021)	Image	✓	✓
SoundFilter (Gfeller et al., 2021)	Audio	✓	
Zero-shot audio separation (Chen et al., 2022)	Audio		(✓)
Text-Queried (Liu et al., 2022)	Text		
Text/Audio-Queried (Kilgour et al., 2022)	Text / Audio		
CLIPSep (ours)	Text / Image	✓	
CLIPSep-NIT (ours)	Text / Image	✓	✓

based on the MixIT model. While image queries can serve as a natural interface for specifying the target sound in certain use cases, images of target sounds become unavailable in low-light conditions and for sounds from out-of-screen objects.

Another line of research uses the audio modality to query acoustically similar sounds. Chen et al. (2022) showed that such approach can generalize to unseen sounds. Later, Gfeller et al. (2021) cropped two disjoint segments from single recording and used them as a query-target pair to train a sound separation model, assuming both segments contain the same sound source. However, in many cases, it is impractical to prepare a reference audio sample for the desired sound as the query.

Most recently, text-queried sound separation has been studied as it provides a natural and scalable interface for specifying arbitrary target sounds as compared to systems that use a fixed set of class labels. Liu et al. (2022) employed a pretrained language model to encode the text query, and condition the model to separate the corresponding sounds. Kilgour et al. (2022) proposed a model that accepts audio or text queries in a hybrid manner. These approaches, however, require labeled text-audio paired data for training. Different from prior work, our goal is to learn text-queried sound separation for arbitrary sound *without* labeled data, specifically using unlabeled noisy videos in the wild.

Contrastive language-image-audio pretraining. The CLIP model (Radford et al., 2021) has been used as a pretraining of joint embedding spaces among text, image and audio modalities for downstream tasks such as audio classification (Wu et al., 2022a; Guzhov et al., 2022) and sound guided image manipulation (Lee et al., 2022). Pretraining is done either in a supervised manner using labels (Guzhov et al., 2022; Lee et al., 2022) or in a self-supervised manner by training an additional audio encoder to map input audio to the pretrained CLIP embedding space (Wu et al., 2022a). In contrast, we explore the zero-shot modality transfer capability of the CLIP model by freezing the pre-trained CLIP model and directly optimizing the rest of the model for the target sound separation task.

6.3 Method

6.3.1 CLIPSep—Learning text-queried sound separation without labeled data

In this section, we propose the CLIPSep model for text-queried sound separation without using labeled data. We base the CLIPSep model on Sound-of-Pixels (SOP) (Zhao et al., 2018) and replace the video analysis network of the SOP model. As illustrated in Figure 6.2, during training, the model takes as inputs an audio mixture $\boldsymbol{x} = \sum_{i=1}^{n} \boldsymbol{s}_i$, where $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ are the n audio tracks, along with their corresponding images $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$ extracted from the videos. We first transform the audio mixture $\boldsymbol{x}$ into a magnitude spectrogram X and pass the spectrogram through an audio U-Net (Ronneberger et al., 2015; Jansson et al., 2017) to produce k ($\geq n$) intermediate masks $\tilde{M}_1, \ldots, \tilde{M}_k$. On the other stream, each image is encoded by the pretrained CLIP model (Radford et al., 2021) into an embedding $\boldsymbol{e}_i \in \mathbb{R}^{512}$. The CLIP embedding $\boldsymbol{e}_i$ will further be projected to a *query vector* $\boldsymbol{q}_i \in \mathbb{R}^k$ by a *projection layer*, which is expected to extract only audio-relevant information from $\boldsymbol{e}_i$.[2] Finally, the query vector $\boldsymbol{q}_i$ will be used to mix the intermediate masks into the final predicted masks $\hat{M}_i = \sum_{j=1}^{k} \sigma\big(w_{ij} q_{ij} \tilde{M}_j + b_i\big)$, where $\boldsymbol{w}_i \in \mathbb{R}^k$ is a learnable scale vector, $b_i \in \mathbb{R}$ a learnable bias, and $\sigma(\cdot)$ the sigmoid function. Now, suppose M_i is the ground truth mask for source $\boldsymbol{s}_i$. The training objective of the model is the sum of the weighted binary cross entropy losses for

[2] We extract three frames with 1-sec intervals and compute their mean CLIP embedding as the input to the projection layer to reduce the negative effects when the selected frame does not contain the objects of interest.

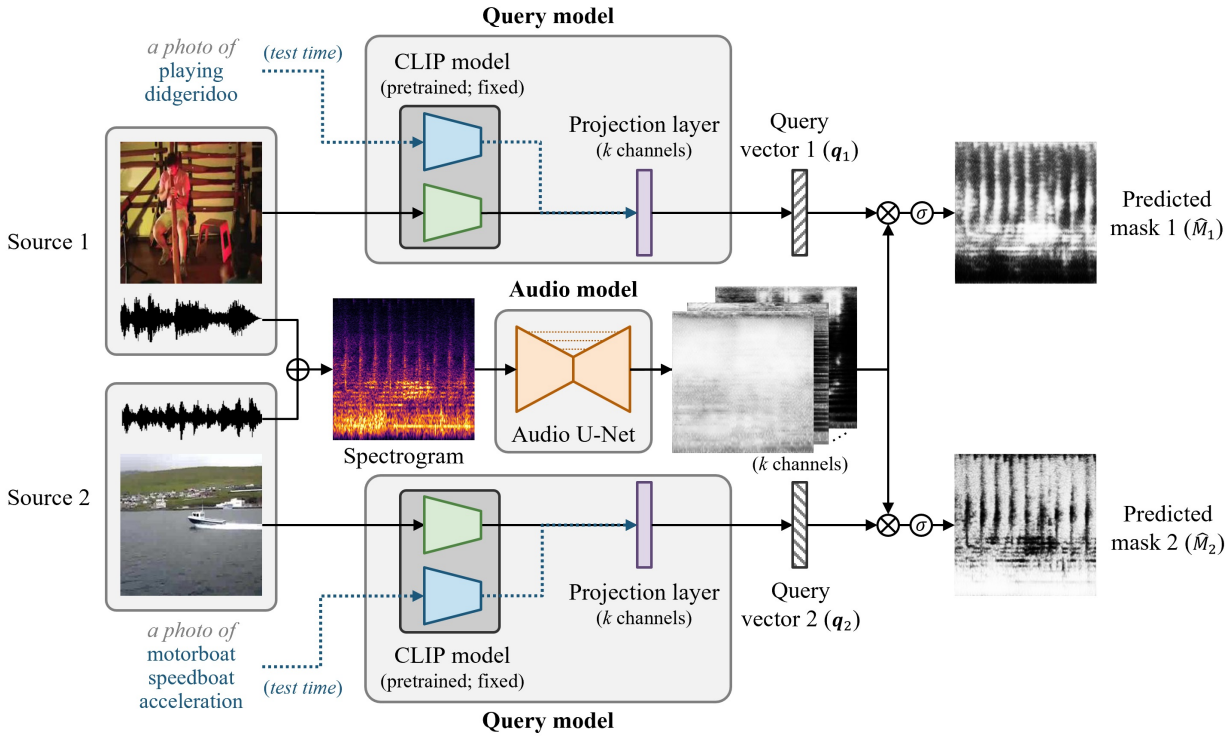


Figure 6.2. An illustration of the proposed CLIPSep model for $n = 2$. During training, we mix audio from two videos and train the model to separate each audio source given the corresponding video frame as the query. At test time, we instead use a text query in the form of "*a photo of [user input query]*" to query the sound separation model. Thanks to the properties of the pretrained CLIP model, the query vectors we obtain for the image and text queries are expected to be close.

each source:

$$\mathcal{L}_{CLIPSep} = \sum_{i=1}^{n} WBCE(M_i, \hat{M}_i) = \sum_{i=1}^{n} X \odot \Big(- M_i \log \hat{M}_i - (1 - M_i) \log \big(1 - \hat{M}_i\big) \Big). \tag{6.1}$$

At test time, thanks to the joint image-text embedding offered by the CLIP model, we feed a text query instead of an image to the query model to obtain the query vector and separate the target sounds accordingly (see Section 6.A for an illustration). As suggested by Radford et al. (2021), we prefix the text query into the form of "*a photo of [user input query]*" to reduce the generalization gap.[3]

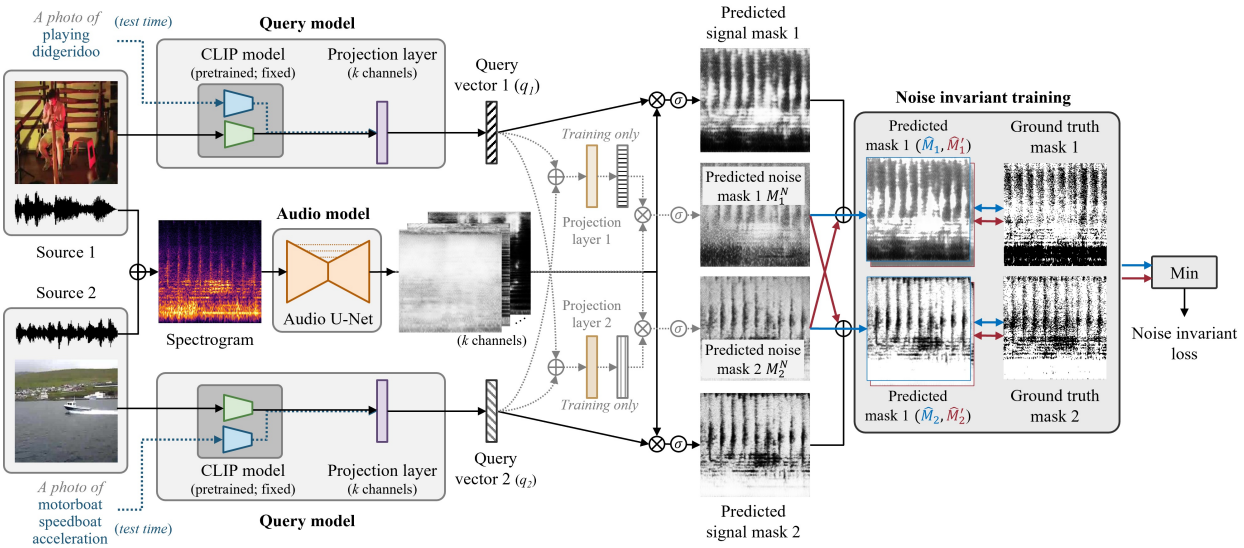


Figure 6.3. An illustration of the proposed CLIPSep-NIT model for $n = 2$. Similar to CLIPSep, we train the model to separate each audio source given the corresponding query image during training and switch to using a text query at test time. The two predicted noise masks are interchangeable for loss computation during training, and they are discarded at test time (grayed out paths).

6.3.2 Noise invariant training—Handling noisy data in the wild

While the CLIPSep model can separate sounds given image or text queries, it assumes that the sources are clean and contain few query-irrelevant sounds. However, this assumption does not hold for videos in the wild as many of them contain out-of-screen sounds and various background noises. Inspired by the mixture invariant training (MixIT) proposed by Wisdom et al. (2020), we further propose the *noise invariant training* (NIT) to tackle the challenge of training with noisy data. As illustrated in Figure 6.3, we introduce n additional permutation invariant heads called *noise heads* to the CLIPSep model, where the masks predicted by these heads are interchangeable during loss computation. Specifically, we introduce n additional projection layers, and each of them takes as input the sum of all query vectors produced by the *query heads* (i.e., $\sum_{i=1}^{n} \boldsymbol{q}_i$) and produce a vector that is later used to mix the intermediate masks into the predicted *noise mask*. In principle, the *query masks* produced by the query vectors are expected to extract query-relevant sounds due to their stronger correlations to their corresponding queries, while the interchangeable noise masks should 'soak up' other sounds. Mathematically, let $M_1^Q, \ldots, M_n^Q$ be the predicted query masks

[3]Similar to how we prepare the image queries, we create four queries from the input text query using four query templates (see Section 6.B) and take their mean CLIP embedding as the input to the projection layer.

and $M_1^N, \ldots, M_n^N$ be the predicted noise masks. Then, the noise invariant loss is defined as:

$$\mathcal{L}_{NIT} = \min_{(j_1,\ldots,j_n)\in\Sigma_n} \sum_{i=1}^{n} WBCE\left(M_i, \min\left(1, \hat{M}_i^Q + \hat{M}_{j_i}^N\right)\right), \tag{6.2}$$

where Σ_n denotes the set of all permutations of $\{1, \ldots, n\}$.[4] Take $n = 2$ for example.[5] We consider the two possible ways for combining the query heads and the noise heads:

$$\text{(Arrangement 1)} \qquad \hat{M}_1 = \min\left(1, \hat{M}_1^Q + \hat{M}_1^N\right), \quad \hat{M}_2 = \min\left(1, \hat{M}_2^Q + \hat{M}_2^N\right), \tag{6.3}$$

$$\text{(Arrangement 2)} \qquad \hat{M}_1' = \min\left(1, \hat{M}_1^Q + \hat{M}_2^N\right), \quad \hat{M}_2' = \min\left(1, \hat{M}_2^Q + \hat{M}_1^N\right). \tag{6.4}$$

Then, the noise invariant loss is defined as the smallest loss achievable:

$$\mathcal{L}_{NIT}^{(2)} = \min\left(WBCE\big(M_1, \hat{M}_1\big) + WBCE\big(M_2, \hat{M}_2\big), WBCE\big(M_1, \hat{M}_1'\big) + WBCE\big(M_2, \hat{M}_2'\big)\right). \tag{6.5}$$

Once the model is trained, we discard the noise heads and use only the query heads for inference (see Section 6.A for an illustration). Unlike the MixIT model (Wisdom et al., 2020), our proposed noise invariant training still allows us to specify the target sound by an input query, and it does not require any post-selection process as we only use the query heads during inference.

In practice, we find that the model tends to assign part of the target sounds to the noise heads as these heads can freely enjoy the optimal permutation to minimize the loss. Hence, we further introduce a regularization term to penalize producing high activations on the noise masks:

$$\mathcal{L}_{REG} = \max\left(0, \sum_{i=1}^{n} \text{mean}\left(\hat{M}_i^N\right) - \gamma\right), \tag{6.6}$$

where $\gamma \in [0, n]$ is a hyperparameter that we will refer to as the *noise regularization level*. The proposed regularization has no effect when the sum of the means of all the noise masks

[4] We note that CLIPSep-NIT considers $2n$ sources in total as the model has n queried heads and n noise heads. While PIT (Yu et al., 2017) and MixIT (Wisdom et al., 2020) respectively require $O((2n)!)$ and $O(2^{2n})$ search to consider $2n$ sources, the proposed NIT only requires $O(n!)$ permutation in the loss computation.

[5] Since our goal is not to further separate the noise into individual sources but to separate the sounds that correspond to the query, n may not need to be large. In practice, we find that the CLIPSep-NIT model with $n = 2$ already learns to handle the noise properly and can successfully transfer to the text-queried mode. Thus, we use $n = 2$ throughout this paper and leave the testing on larger n as future work.

Table 6.2. Results on the MUSIC dataset. Standard errors are reported in the mean SDR column. Bold values indicate the largest SDR achieved per group.

Model	Unlabeled data	Post-proc. free	Query type		SDR [dB]	
			Training	Test	Mean	Median
Mixture	-	-	-	-	0.00 ± 0.89	0.00
Text-queried models						
CLIPSep	✓	✓	Image	Text	**5.49** ± **0.72**	**4.97**
CLIPSep-Text		✓	Text	Text	7.91 ± 0.81	7.46
CLIPSep-Hybrid		✓	Text + Image	Text	**8.36** ± **0.83**	**8.72**
Image-queried models						
SOP (Zhao et al., 2018)	✓	✓	Image	Image	6.59 ± 0.85	**6.22**
CLIPSep	✓	✓	Image	Image	**7.03** ± **0.70**	5.85
CLIPSep-Text		✓	Text	Image	6.25 ± 0.72	6.19
CLIPSep-Hybrid		✓	Text + Image	Image	**8.06** ± **0.79**	**8.01**
Nonqueried models						
LabelSep		✓	Label	Label	8.18 ± 0.80	**7.82**
PIT (Yu et al., 2017)	✓		×	×	**8.68** ± **0.76**	7.67

is lower than a predefined threshold γ, while having a linearly growing penalty when the sum is higher than γ. Finally, the training objective of the CLIPSep-NIT model is a weighted sum of the noise invariant loss and regularization term: $\mathcal{L}_{CLIPSep\text{-}NIT} = \mathcal{L}_{NIT} + \lambda\mathcal{L}_{REG}$, where $\lambda \in \mathbb{R}$ is a weight hyperparameter. We set $\lambda = 0.1$ for all experiments, which we find work well across different settings.

6.4 Experiments

We base our implementations on the code provided by Zhao et al. (2018) (https://github.com/hangzhaomit/Sound-of-Pixels). Implementation details can be found in Section 6.C.

6.4.1 Experiments on clean data

We first evaluate the proposed CLIPSep model without the noise invariant training on musical instrument sound separation task using the MUSIC dataset, as done in (Zhao et al., 2018). This experiment is designed to focus on evaluating the quality of the learned query vectors and the zero-shot modality transferability of the CLIPSep model on a small, clean dataset rather than showing its ability to separate arbitrary sounds. The MUSIC dataset is a collection of 536 video recordings of people playing a musical instrument out of 11

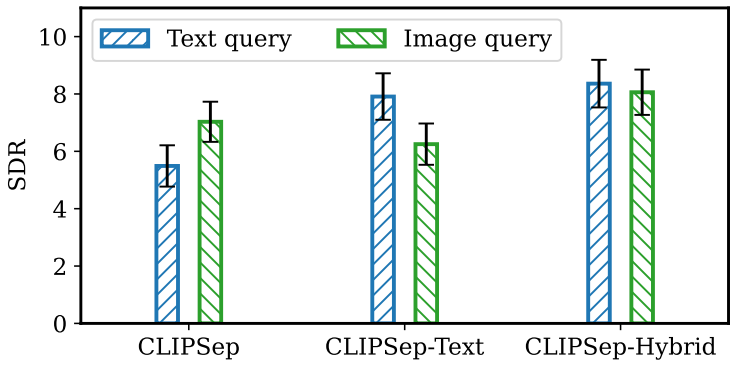


Figure 6.4. Mean SDR and standard errors of the models trained and tested on different modalities.

instrument classes. Since no existing work has trained a text-queried sound separation model using only unlabeled data to our knowledge, we compare the proposed CLIPSep model with two baselines that serve as upper bounds—the PIT model (Yu et al., 2017, see Section 6.D for an illustration) and a version of the CLIPSep model where the query model is replaced by learnable embeddings for the labels, which we will refer to as the LabelSep model. In addition, we also include the SOP model (Zhao et al., 2018) to investigate the quality of the query vectors as the CLIPSep and SOP models share the same network architecture except the query model.

We report the results in Table 6.2. Our proposed CLIPSep model achieves a mean signal-to-distortion ratio (SDR) (Vincent et al., 2006) of 5.49 dB and a median SDR of 4.97 dB using text queries in a zero-shot modality transfer setting. When using image queries, the performance of the CLIPSep model is comparable to that of the SOP model. This indicates that the CLIP embeddings are as informative as those produced by the SOP model. The performance difference between the CLIPSep model using text and image queries at test time indicates the *zero-shot modality transfer gap*. We observe 1.54 dB and 0.88 dB differences on the mean and median SDRs, respectively. Moreover, we also report in Table 6.2 and Figure 6.4 the performance of the CLIPSep models trained on different modalities to investigate their modality transferability in different settings. We notice that when we train the CLIPSep model using text queries, dubbed as CLIPSep-Text, the mean SDR using text queries increases to 7.91 dB. However, when we test this model using image queries, we observe a 1.66 dB difference on the mean SDR as compared to that using text queries, which is close to the

Table 6.3. Results of the MUSIC⁺ and VGGSound-Clean⁺ evaluations (see Section 6.4.2). Standard errors are reported in the mean SDR [dB] columns. Bold values indicate the largest SDR achieved per group. We use $\gamma = 0.25$ for CLIPSep-NIT. Note that the LabelSep model does not work on the MUSIC dataset due to the different label taxonomies of the MUSIC and VGGSound datasets.

			MUSIC⁺		VGGSound-Clean⁺	
Model	Unlabeled data	Post-proc. free	Mean SDR	Median SDR	Mean SDR	Median SDR
Mixture	-	-	4.49 ± 1.41	2.04	-0.77 ± 1.31	-0.84
Text-queried models						
CLIPSep	✓	✓	9.71 ± 1.21	8.73	2.76 ± 1.00	**3.95**
CLIPSep-NIT	✓	✓	**10.27 ± 1.04**	**10.02**	**3.05 ± 0.73**	3.26
BERTSep		✓	4.67 ± 0.44	4.41	5.09 ± 0.80	5.49
CLIPSep-Text		✓	10.73 ± 0.99	9.93	5.49 ± 0.82	5.06
Image-queried models						
SOP (Zhao et al., 2018)	✓	✓	11.44 ± 1.18	11.18	2.99 ± 0.84	3.89
CLIPSep	✓	✓	**12.20 ± 1.17**	**12.42**	**5.46 ± 0.79**	**5.35**
CLIPSep-NIT	✓	✓	11.28 ± 1.08	10.83	4.84 ± 0.66	3.57
CLIPSep-Text		✓	9.89 ± 1.04	8.09	2.45 ± 0.70	1.74
Nonqueried models						
PIT (Yu et al., 2017)	✓		**12.24 ± 1.20**	**12.53**	**5.73 ± 0.79**	**4.97**
LabelSep		✓	-	-	5.55 ± 0.81	5.29

mean SDR difference we observe for the model trained with image queries. Finally, we train a CLIPSep model using both text and image queries in alternation, dubbed as CLIPSep-Hybrid. We see that it leads to the best test performance for both text and image modalities, and there is only a mean SDR difference of 0.30 dB between using text and image queries. As a reference, the LabelSep model trained with labeled data performs worse than the CLIPSep-Hybrid model using text queries. Further, the PIT model achieves a mean SDR of 8.68 dB and a median SDR of 7.67 dB, but it requires post-processing to figure out the correct assignments.

6.4.2 Experiments on noisy data

Next, we evaluate the proposed method on a large-scale dataset aiming at universal sound separation. We use the VGGSound dataset (Chen et al., 2020a), a large-scale audio-visual dataset containing more than 190,000 10-second videos in the wild out of more than 300 classes. We find that the audio in the VGGSound dataset is often noisy and contains off-screen sounds and background noise. Although we train the models on such noisy data,

it is not suitable to use the noisy data as targets for evaluation because it fails to provide reliable results. For example, if the target sound labeled as "dog barking" also contains human speech, separating only the dog barking sound provides a lower SDR value than separating the mixture of dog barking sound and human speech even though the text query is "dog barking". (Note that we use the labels only for evaluation but not for training.) To avoid this issue, we consider the following two evaluation settings:

- **MUSIC**$^+$: Samples in the MUSIC dataset are used as clean targets and mixed with a sample in the VGGSound dataset as an interference. The separation quality is evaluated on the clean target from the MUSIC dataset. As we do not use the MUSIC dataset for training, this can be considered as zero-shot transfer to a new data domain containing unseen sounds (Radford et al., 2019; Brown et al., 2020). To avoid the unexpected overlap of the target sound types in the MUSIC and VGGSound datasets caused by the label mismatch, we exclude all the musical instrument playing videos from the VGGSound dataset in this setting.

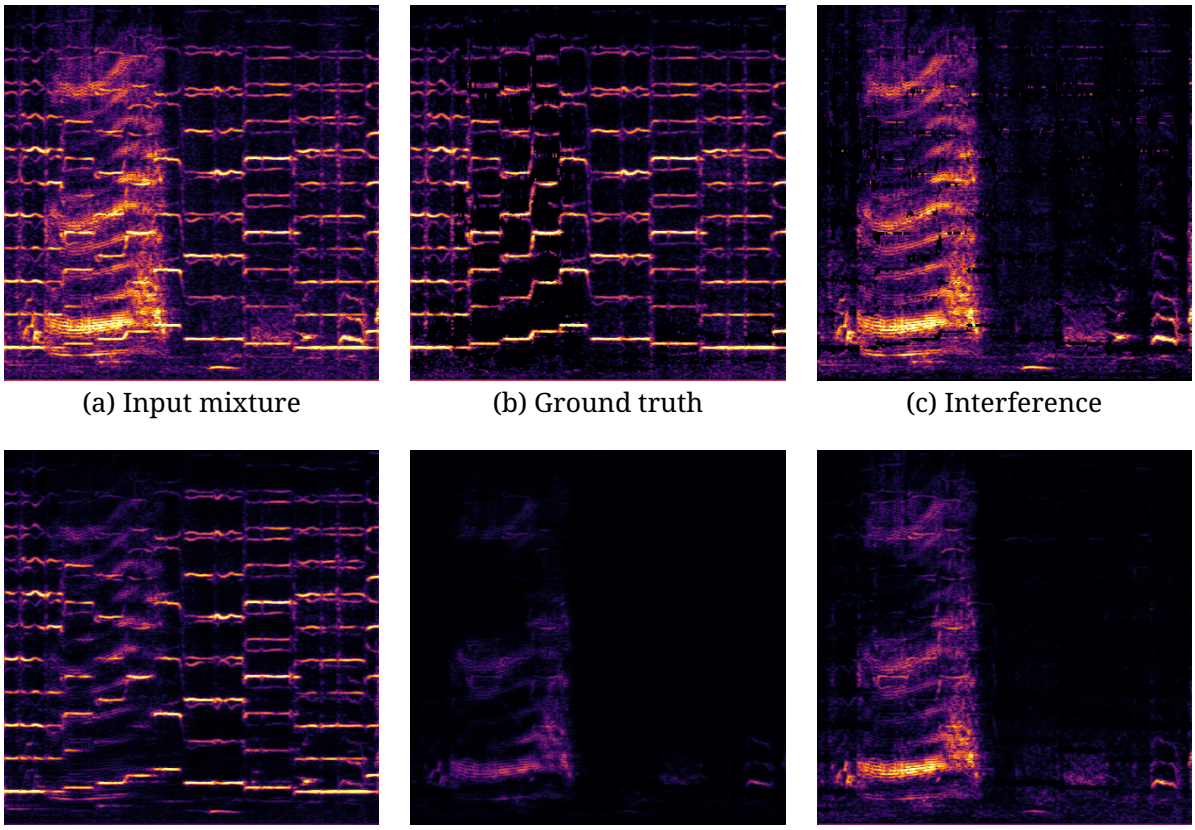
- **VGGSound-Clean**$^+$: We manually collect 100 clean samples that contain distinct target sounds from the VGGSound test set, which we will refer to as VGGSound-Clean. We mix an audio sample in VGGSound-Clean with another in the test set of VGGSound. Similarly, we consider the VGGSound audio as an interference sound added to the relatively cleaner VGGSound-Clean audio and evaluate the separation quality on the VGGSound-Clean stem.

Table 6.3 shows the evaluation results. First, CLIPSep successfully learns text-queried sound separation even with noisy unlabeled data, achieving 5.22 dB and 3.53 dB SDR improvements over the mixture on MUSIC$^+$ and VGGSound-Clean$^+$, respectively. By comparing CLIPSep and CLIPSep-NIT, we observe that NIT improves the mean SDRs in both settings. Moreover, on MUSIC$^+$, CLIPSep-NIT's performance matches that of CLIPSep-Text, which utilizes labels for training, achieving only a 0.46 dB lower mean SDR and even a 0.05 dB higher median SDR. This result suggests that the proposed self-supervised text-queried sound separation method can learn separation capability competitive with the fully supervised model in some target sounds. In contrast, there is still a gap between them on VGGSound-

Clean$^+$, possibly because the videos of non-music-instrument objects are more noisy in both audio and visual domains, thus resulting in a more challenging zero-shot modality transfer. This hypothesis is also supported by the higher zero-shot modality transfer gap (mean SDR difference of image- and text-queried mode) of 1.79 dB on VGGSound-Clean$^+$ than that of 1.01 dB on MUSIC$^+$ for CLIPSep-NIT. In addition, we consider another baseline model that replaces the CLIP model in CLIPSep with a BERT encoder (Devlin et al., 2019), which we call BERTSep. Interestingly, although BERTSep performs similarly to CLIPSep-Text on VGGSound-Clean$^+$, the performance of BERTSep is significantly lower than that of CLIPSep-Text on MUISC$^+$, indicating that BERTSep fails to generalize to unseen text queries. We hypothesize that the CLIP text embedding captures the timbral similarity of musical instruments better than the BERT embedding do, because the CLIP model is aware of the visual similarity between musical instruments during training. Moreover, it is interesting to see that CLIPSep outperforms CLIPSep-NIT when an image query is used at test time (domain-matched condition), possibly because images contain richer context information such as objects nearby and backgrounds than labels, and the models can use such information to better separate the target sound. While CLIPSep has to fully utilize such information, CLIPSep-NIT can use the noise heads to model sounds that are less relevant to the image query. Since we remove the noise heads from CLIPSep-NIT during the evaluation, it can rely less on such information from the image, thus improving the zero-shot modality transferability. Figure 6.5 shows an example of the separation results on MUSIC$^+$ (see Figures 6.12 to 6.15 for more examples). We observe that the two noise heads contain mostly background noise. Audio samples can be found on our demo website.[1]

6.4.3 Examining the effects of the noise regularization level γ

In this experiment, we examine the effects of the noise regularization level γ in Equation (6.6) by changing the value from 0 to 1. As we can see from Figure 6.6 (a) and (b), CLIPSep-NIT with $\gamma = 0.25$ achieves the highest SDR on both evaluation settings. This suggests that the optimal γ value is not sensitive to the evaluation dataset. Further, we also report in Figure 6.6 (c) the total mean noise head activation, $\sum_{i=1}^{n} \text{mean}(\hat{M}_i^N)$, on the validation set. As $\hat{M}_i^N$ is the mask estimate for the noise, the total mean noise head activation value



(a) Input mixture (b) Ground truth (c) Interference

(d) Prediction (e) Noise head 1 (f) Noise head 2

Figure 6.5. Example results of the proposed CLIPSep-NIT model with $\gamma = 0.25$ on the MUSIC$^+$ dataset. We mix the an audio sample ("violin" in this example) in the MUSIC dataset with an interference audio sample ("people sobbing" in this example) in the VGGSound dataset to create an artificial mixture. (b) and (c) show the reconstructed signals using the ground truth ideal binary masks. The spectrograms are shown in the log frequency scale. We observe that the proposed model successfully separates the desired sounds (i.e., (d)) from query-irrelevant noise (i.e., (e) and (f)).

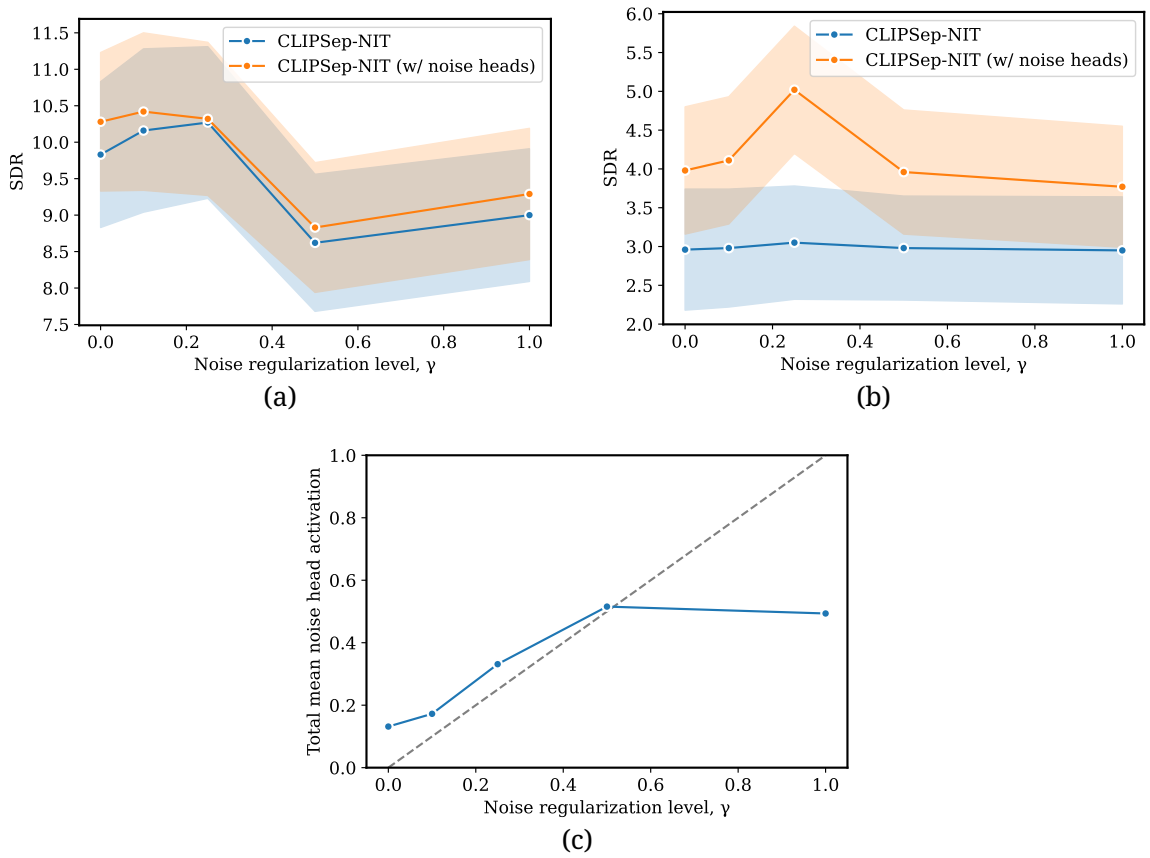


Figure 6.6. Effects of the noise regularization level γ for the proposed CLIPSep-NIT model—mean SDR for the (a) MUSIC$^+$ and (b) VGGSound-Clean$^+$ evaluations, and (c) the total mean noise head activation, $\sum_{i=1}^{n} \text{mean}(\hat{M}_i^N)$, on the validation set. The shaded areas show standard errors.

indicates to what extent signals are assigned to the noise head. We observe that the proposed regularizer successfully keeps the total mean noise head activation close to the desired level, γ, for $\gamma \leq 0.5$. Interestingly, the total mean noise head activation is still around 0.5 when $\gamma = 1.0$, suggesting that the model inherently tries to use both the query-heads and the noise heads to predict the noisy target sounds. Moreover, while we discard the noise heads during evaluation in our experiments, keeping the noise heads can lead to a higher SDR as shown in Figure 6.6 (a) and (b), which can be helpful in certain use cases where a post-processing procedure similar to the PIT model (Yu et al., 2017) is acceptable.

6.5 Discussions

For the experiments presented in this paper, we work on labeled datasets so that we can evaluate the performance of the proposed models. However, our proposed models

do not require any labeled data for training, and can thus be trained on larger unlabeled video collections in the wild. Moreover, we observe that the proposed model shows the capability of combing multiple queries, e.g., "*a photo of [query A] and [query B],*" to extract multiple target sounds, and we report the results on the demo website. This offers a more natural user interface against having to separate each target sound and mix them via an additional post-processing step. We also show in Section 6.G that our proposed model is robust to different text queries and can extract the desired sounds.

In our experiments, we often observe a modality transfer gap greater than 1 dB difference of SDR. A future research direction is to explore different approaches to reduce the modality transfer gap. For example, the CLIP model is pretrained on a different dataset, and thus finetuning the CLIP model on the target dataset can help improve the underlying modality transferability within the CLIP model. Further, while the proposed noise invariant training is shown to improve the training on noisy data and reduce the modality transfer gap, it still requires a sufficient audio-visual correspondence for training video. In other words, if the audio and images are irrelevant in most videos, the model will struggle to learn the correspondence between the query and target sound. In practice, we find that the data in the VGGSound dataset often contains off-screen sounds and the labels sometimes correspond to only part of the video content. Hence, filtering on the training data to enhance its audio-visual correspondence can also help reduce the modality transfer gap. This can be achieved by self-supervised audio-visual correspondence prediction (Arandjelović and Zisserman, 2017a; Arandjelović and Zisserman, 2017b) or temporal synchronization (Korbar et al., 2018; Owens and Efros, 2018).

Another future direction is to explore the semi-supervised setting where a small subset of labeled data can be used to improve the modality transferability. We can also consider the proposed method as a pretraining on unlabeled data for other separation tasks in the low-resource regime. We include in Section 6.H a preliminary experiment in this aspect using the ESC-50 dataset (Piczak, 2015).

6.6 Conclusion

In this work, we have presented a novel text-queried universal sound separation model that can be trained on noisy unlabeled videos. In this end, we have proposed to use the contrastive image-language pretraining to bridge the audio and text modalities, and proposed the noise invariant training for training a query-based sound separation model on noisy data. We have shown that the proposed models can learn to separate an arbitrary sound specified by a text query out of a mixture, even achieving competitive performance against a fully supervised model in some settings. We believe our proposed approach closes the gap between the ways humans and machines learn to focus on a sound in a mixture, namely, the multi-modal self-supervised learning paradigm of humans against the supervised learning paradigm adopted by existing label-based machine learning approaches.

6.7 Acknowledgements

We would like to thank Stefan Uhlich, Giorgio Fabbro and Woosung Choi for their helpful comments during the preparation of this manuscript. We also thank Mayank Kumar Singh for supporting the setup of the subjective test in Section 6.F. Hao-Wen thank J. Yang and Family Foundation and Taiwan Ministry of Education for supporting his PhD study.

* * *

This chapter, in full, is a reprint of the material as it appears in "CLIPSep: Learning Text-queried Sound Separation with Noisy Unlabeled Videos" by Hao-Wen Dong, Naoya Takahashi, Yuki Mitsufuji, Julian McAuley and Taylor Berg-Kirkpatrick, which was published in the Proceedings of the International Conference on Learning Representations (ICLR) in 2023. The dissertation author was the primary investigator and author of this paper.

Appendices

6.A Inference Pipeline of CLIPSep and CLIPSep-NIT

Figure 6.7 illustrates the inference pipeline for the proposed CLIPSep and CLIPSep-NIT models. For the CLIPSep-NIT model, we discard the noise heads at test time. The masked spectrogram is combined with the input phase and converted to the waveform by the inverse short-time Fourier transform (STFT).

6.B Query Ensembling

Radford et al., 2021 suggest that using a *prompt template* in the form of "*a photo of [user input query]*" helps bridge the distribution gap between text queries used for zero-shot image classification and text in the training dataset for the CLIP model. They further show that the ensemble of various prompt templates improve the generalizability. Motivated by this observation, we adopt a similar idea and use several query templates at test time (see Table 6.4). These query templates are heuristically chosen to handle the noisy images extracted from videos.

6.C Implementation Details

We implement the audio model as a 7-layer U-Net (Ronneberger et al., 2015). We use $k = 32$. We use binary masks as the ground truth masks during training while using the raw, real-valued masks for evaluation. We train all the models for 200,000 steps with a batch size of 32. We use the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. In addition, we clip the norm of the gradients to 1.0 (Zhang et al., 2020). We adopt the following learning rate schedule with a warm-up—the learning rate starts from 0 and grows to 0.001 after 5,000 steps, and then it linearly drops to 0.0001 at 100,000 steps and keeps this value thereafter. We validate the model every 10,000 steps using image queries as we do not assume labeled data is available for the validation set. We use a sampling rate of 16,000 Hz and work on audio clips of length 65,535 samples ($\approx$ 4 seconds). During training, we randomly sample a center frame from a video and extract three frames (images) with 1-sec

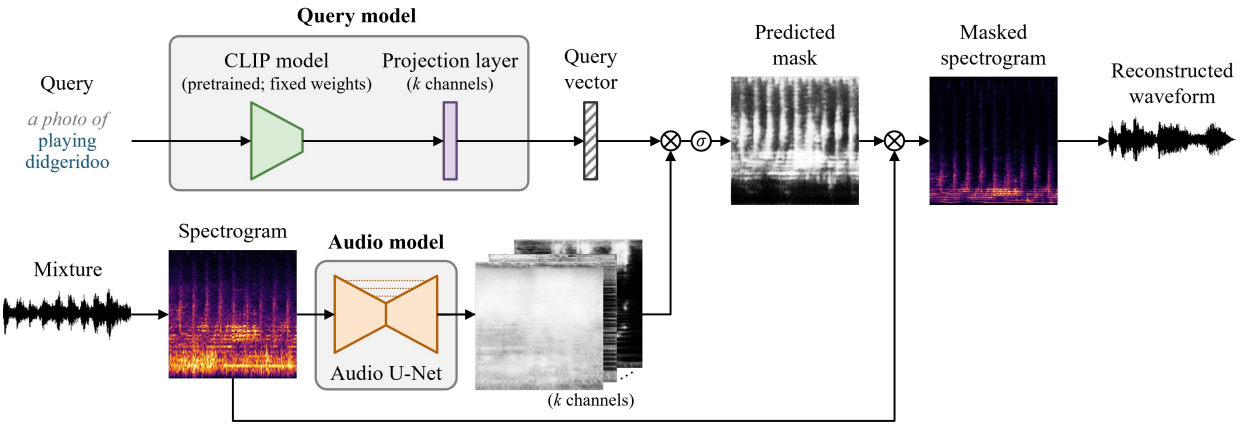


Figure 6.7. Inference pipeline of the proposed CLIPSep and CLIPSep-NIT models. Note that the mixture spectrogram and the masks are shown in log frequency scale, while the masked spectrogram is shown in linear frequency scale.

Table 6.4. Query templates used in our experiments.

Query template	Example query for the label "*dog barking*"
a photo of [*user input query*]	a photo of dog barking
a photo of the small [*user input query*]	a photo of the small dog barking
a low resolution photo of a [*user input query*]	a low resolution photo of a dog barking
a photo of many [*user input query*]	a photo of many dog barking

intervals and 4-sec audio around the center frame. During inference, for image-queried models, we extract three frames with 1-sec intervals around the center of the test clip. For the spectrogram computation, we use a filter length of 1024, a hop length of 256 and a window size of 1024 in the short-time Fourier transform (STFT). We resize images extracted from video to a size of 224-by-224 pixels. For the CLIPSep-Hybrid model, we alternatively train the model with text and image queries, i.e., one batch with all image queries and next with all text queries, and so on. We implement all the models using the PyTorch library (Paszke et al., 2019). We compute the signal-to-distortion ratio (SDR) using museval (Stöter et al., 2018).

In our preliminary experiments, we also tried directly predicting the final mask by conditioning the audio model on the query vector. We applied this modification for both SOP and CLIPSep models, however, we observe that this architecture is prone to overfitting. We hypothesize that this is because the audio model is powerful enough to remember the subtle clues in the query vector, which hinder the generalization to a new sound and query. In contrast, the proposed architecture first predicts over-determined masks and then combines

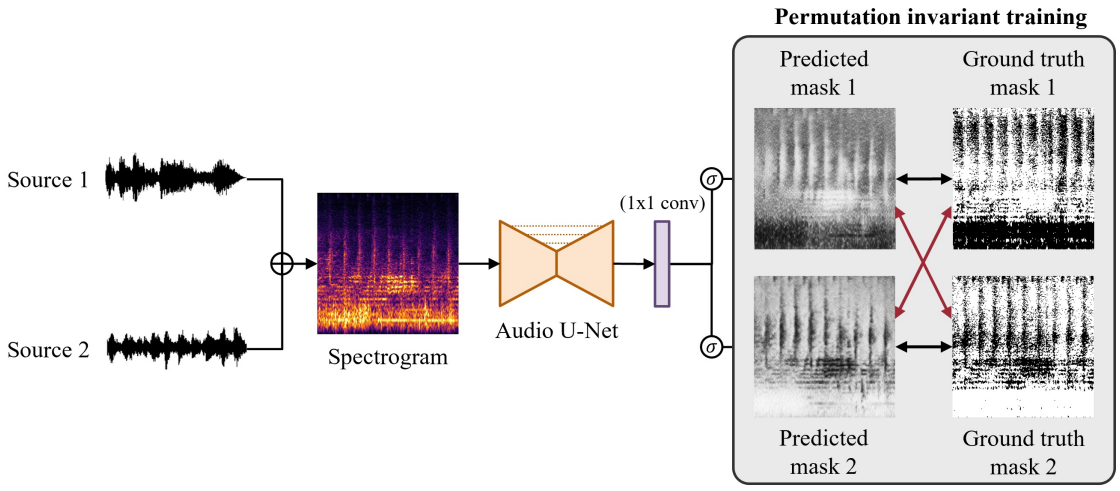


Figure 6.8. An illustration of the PIT model for $n = 2$. The two predicted masks are interchangeable during the loss computation. Since the two predicted masks are interchangeable, the PIT model requires an additional post-selection step to obtain the target sound.

them on the basis of the query vector, which avoids the overfitting problem due to the simple fusion step.

6.D Permutation Invariant Training

Figure 6.8 illustrates the permuatation invariant training (PIT) model (Yu et al., 2017). The permutation invariant loss is defined as follows for $n = 2$.

$$\mathcal{L}_{PIT} = \min\left(WBCE(M_1, \hat{M}_1) + WBCE(M_2, \hat{M}_2), WBCE(M_1, \hat{M}_2) + WBCE(M_2, \hat{M}_1)\right), \tag{6.7}$$

where $\hat{M}_1$ and $\hat{M}_2$ are the predicted masks. Note that the PIT model requires an additional post-selection step to obtain the target sound.

6.E Qualitative Example Results

We show in Figures 6.12 to 6.15 some example results. More results and audio samples can be found at https://sony.github.io/CLIPSep/.

Table 6.5. Subjective test results.

Model	Query type	Correct [%]	Wrong [%]	Both [%]	None [%]
Mixture	-	17.5	10.0	72.5	0.0
CLIPSep	Image	70.6	0.0	29.4	0.0
	Text	66.3	3.8	30.0	0.0
CLIPSep-NIT	Image	68.8	1.9	28.1	1.3
	Text	83.8	0.6	15.0	0.6

6.F Subjective Evaluation

We conduct a subjective test to evaluate whether the SDR results aligned with perceptual quality. As done in the Sound of Pixel (Zhao et al., 2018), separated audio samples are randomly presented to evaluators, and the following question is asked: "Which sound do you hear? 1. A, 2. B, 3. Both, or 4. None of them". Here A and B are replaced by labels of their mixture sources, e.g. A=accordion, B=engine accelerating. Ten samples (including naturally occurring mixture) are evaluated for each model and 16 evaluators have participated in the evaluation. Table 6.5 shows the percentages of samples which are correctly identified the target sound class (Correct), which are incorrectly identified the target sound sources (Wrong), which are selected as both sounds are audible (Both), and which are selected as neither of the sounds are audible (None). The results indicate that the evaluators more often choose the correct sound source for CLIPSep-NIT (83.8%) than CLIPSep (66.3%) with text queries. Notably, CLIPSep-NIT with text-query obtained a higher correct score than that with image-query, which matches the training mode. This is probably because image queries often contain information about backgrounds and environments, hence, some noise and off-screen sounds are also suggested by the image-queries and leak to the query head. In contrast, text-queries purely contain the information of target sounds, thus, the query head more aggressively extract the target sounds.

6.G Robustness to different queries

To examine the model's robustness to different queries, we take the same input mixture and query the model with different text queries. We use the CLIPSep-NIT model on the MUSIC$^+$ dataset and report in Figure 6.16 the results. We see that the model is robust to

Table 6.6. Results on the ESC-50 dataset. Standard errors are reported in the mean SDR column.

Model	Post-processing free	Dataset		SDR	
		Training	Finetuning	Mean	Median
Mixture	-	-	-	0.00 ± 0.44	0.00
PIT	×	VGGSound	-	4.90 ± 0.26	2.44
CLIPSep	✓	VGGSound	-	1.07 ± 0.28	2.34
	✓	ESC-50	-	5.18 ± 0.26	5.09
	✓	VGGSound	ESC-50	6.73 ± 0.26	5.89

different text queries and can extract the desired sounds. Audio samples can be found at https://sony.github.io/CLIPSep/.

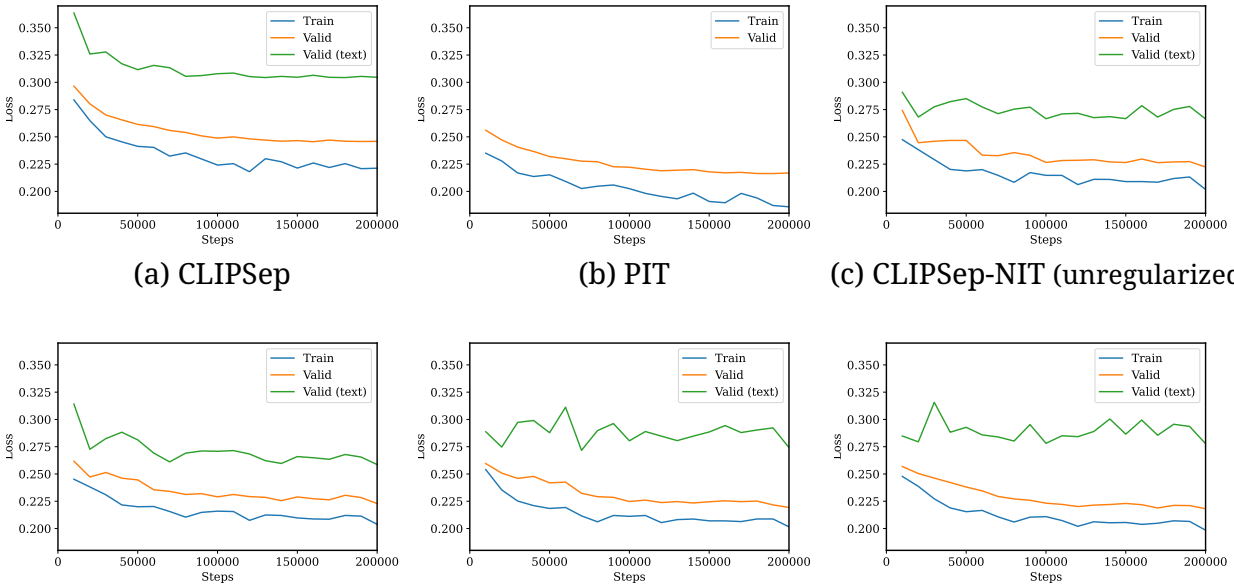
6.H Finetuning Experiments on the ESC-50 Dataset

In this experiment, we aim to examine the possibilities of having a clean dataset for further finetuning. We consider the ESC-50 dataset (Piczak, 2015), a collection of 2,000 high-quality environmental audio recordings, as the clean dataset here.[6] We report the experimental results in Table 6.6. We can see that the model pretrained on VGGSound does not generalize well to the ESC-50 dataset as the ESC-50 contains much cleaner sounds, i.e., without query-irrelevant sounds and background noise. Further, if we train the CLIPSep model from scratch on the ESC-50 dataset, it can only achieve a mean SDR of 5.18 dB and a median SDR of 5.09 dB. However, if we take the model pretrained on the VGGSound dataset and finetune it on the ESC-50 dataset, it can achieve a mean SDR of 6.73 dB and a median SDR of 4.89 dB, resulting in an improvement of 1.55 dB on the mean SDR.

6.I Training Behaviors

We present in Figure 6.9 the training and validation losses along the training progress. Please note that we only show the results obtained using text queries for reference but do not use them for choosing the best model. We also evaluate the intermediate checkpoints every 10,000 steps and present in Figure 6.10 the test SDR along the training progress. In addition, for the CLIPSep-NIT model, we visualize in Figure 6.11 the total mean noise head activation,

[6] https://github.com/karolpiczak/ESC-50

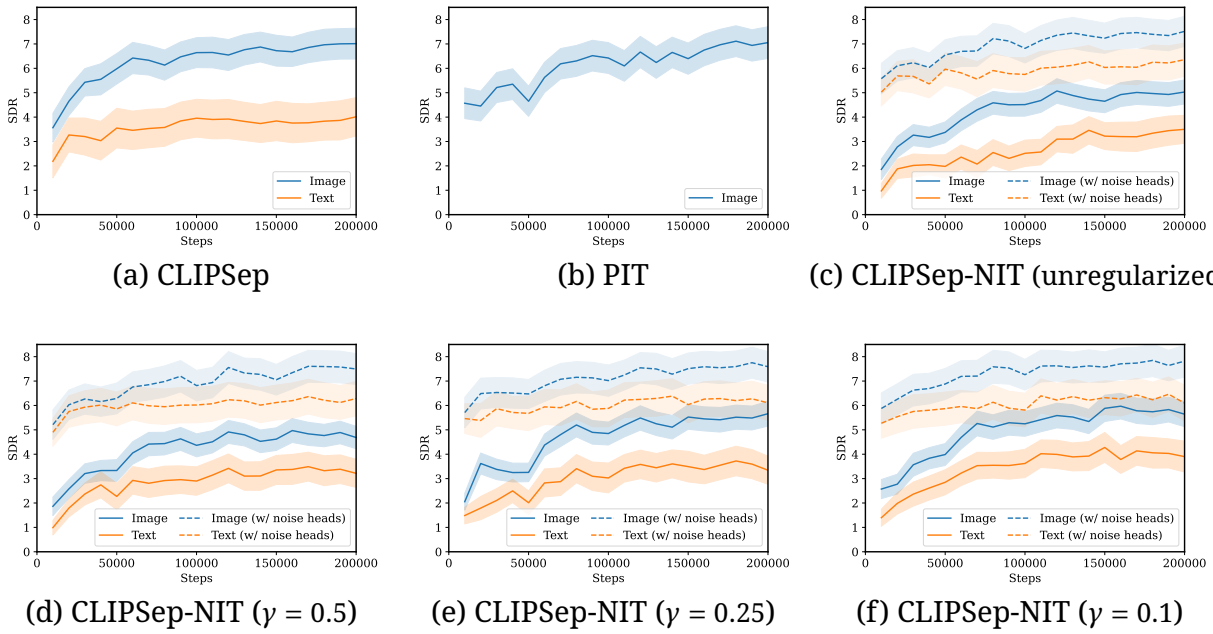


(a) CLIPSep (b) PIT (c) CLIPSep-NIT (unregularized)

(d) CLIPSep-NIT ($\gamma = 0.5$) (e) CLIPSep-NIT ($\gamma = 0.25$) (f) CLIPSep-NIT ($\gamma = 0.1$)

Figure 6.9. Training and validation losses along the training progress on the VGGSound dataset. We also include the losses computed using text queries instead of image queries. The y-axes are intentionally set to the same range for easy comparison. Note that we do not use the validation results obtained with text queries for choosing the best model.

$\sum_{i=1}^{n} \text{mean}(\hat{M}_i^N)$, along the training progress. We can see that the total mean noise head activation stays around the desired level for $\gamma = 0.1, 0.25$. For $\gamma = 0.5$ and the unregularized version, the total mean noise head activation converges to a similar value around 0.55.



(a) CLIPSep (b) PIT (c) CLIPSep-NIT (unregularized)

(d) CLIPSep-NIT ($\gamma = 0.5$) (e) CLIPSep-NIT ($\gamma = 0.25$) (f) CLIPSep-NIT ($\gamma = 0.1$)

Figure 6.10. Test SDR along the training progress on the VGGSound-Clean dataset. The y-axes are intentionally set to the same range for easy comparison.

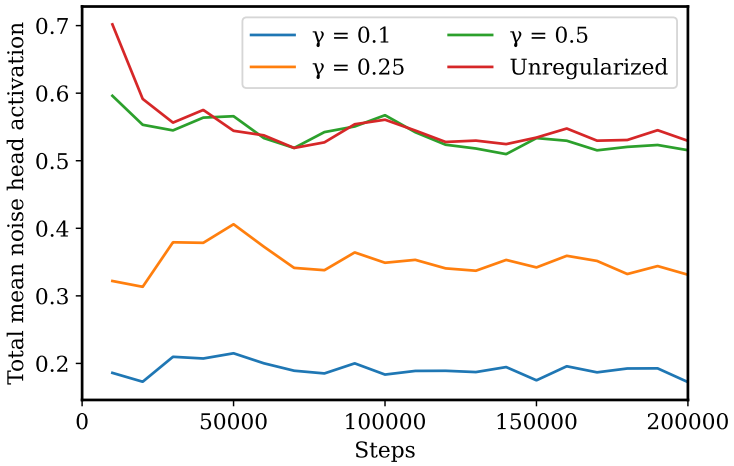


Figure 6.11. Total mean noise head activation, $\sum_{i=1}^{n} \text{mean}(\hat{M}_i^N)$, on the validation set for the CLIPSep-NIT models along the training progress.

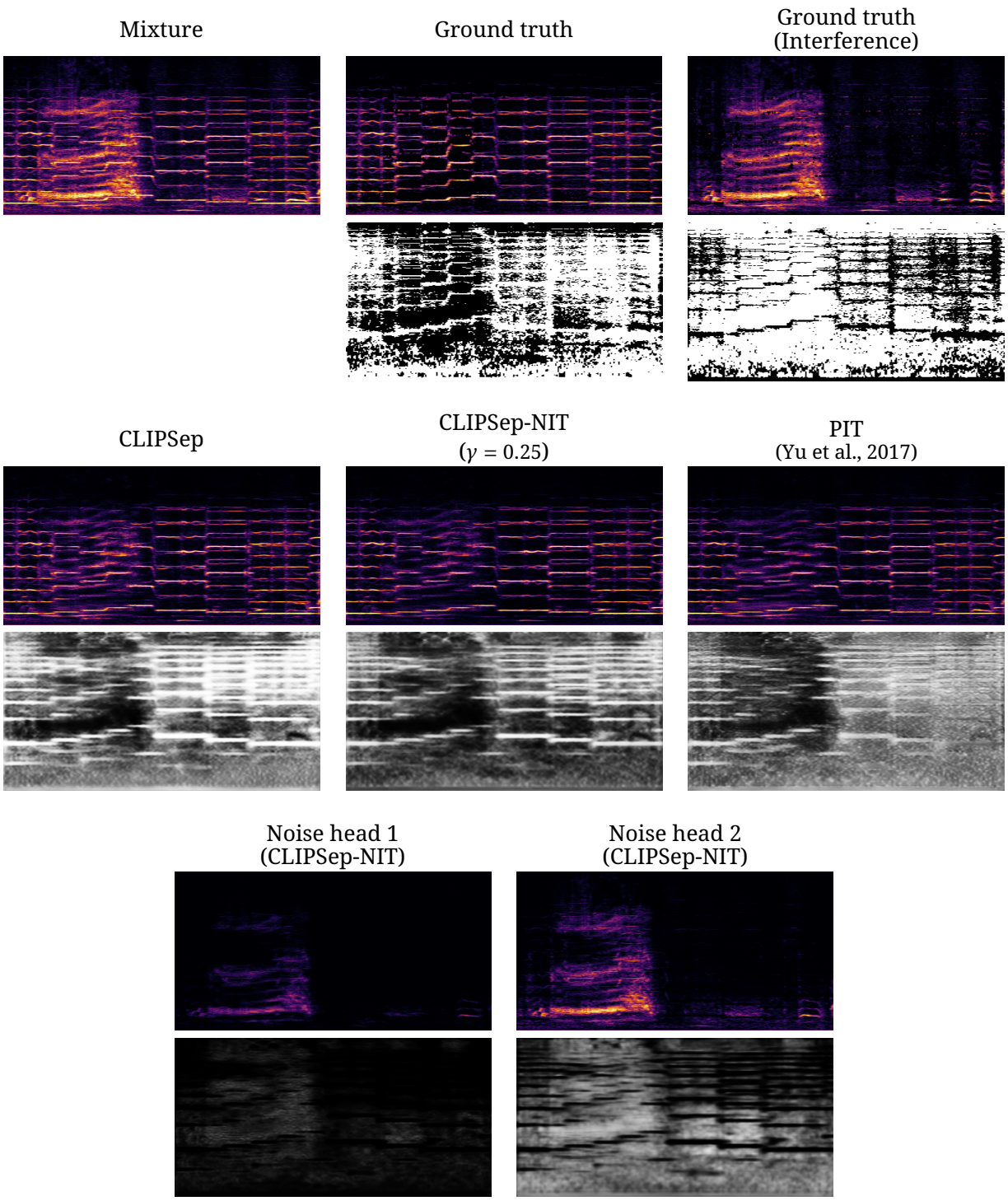


Figure 6.12. Example results on the MUSIC$^+$ dataset. Target source—"violin"; interference—"people sobbing"; query—"*violin*". The spectrograms and masks are shown in the log and linear frequency scales, respectively.

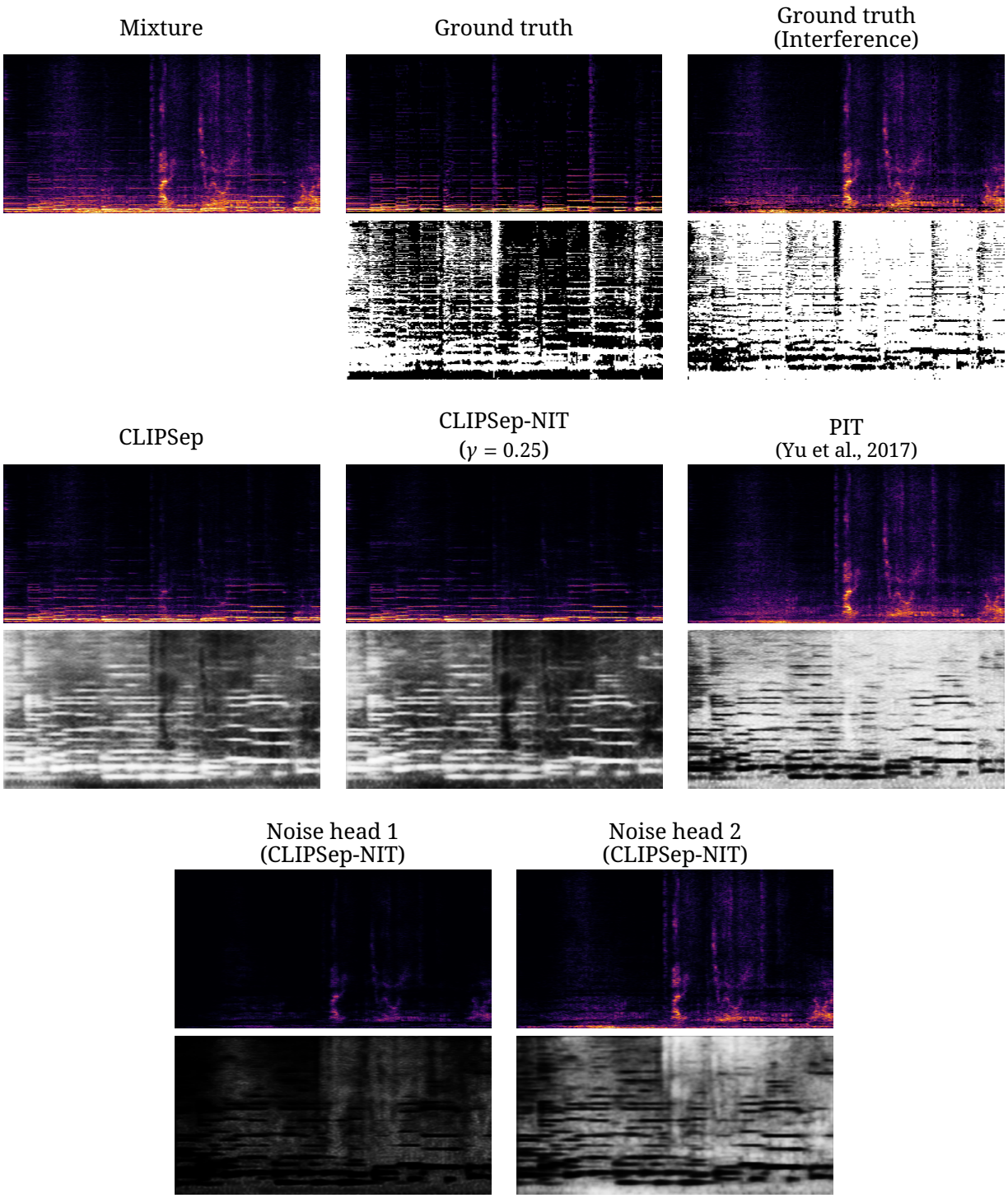


Figure 6.13. Example results on the MUSIC$^+$ dataset. Target source—"acoustic guitar"; interference—"cheetah chirrup", query—"*acoustic guitar*". The spectrograms and masks are shown in the log and linear frequency scales, respectively.

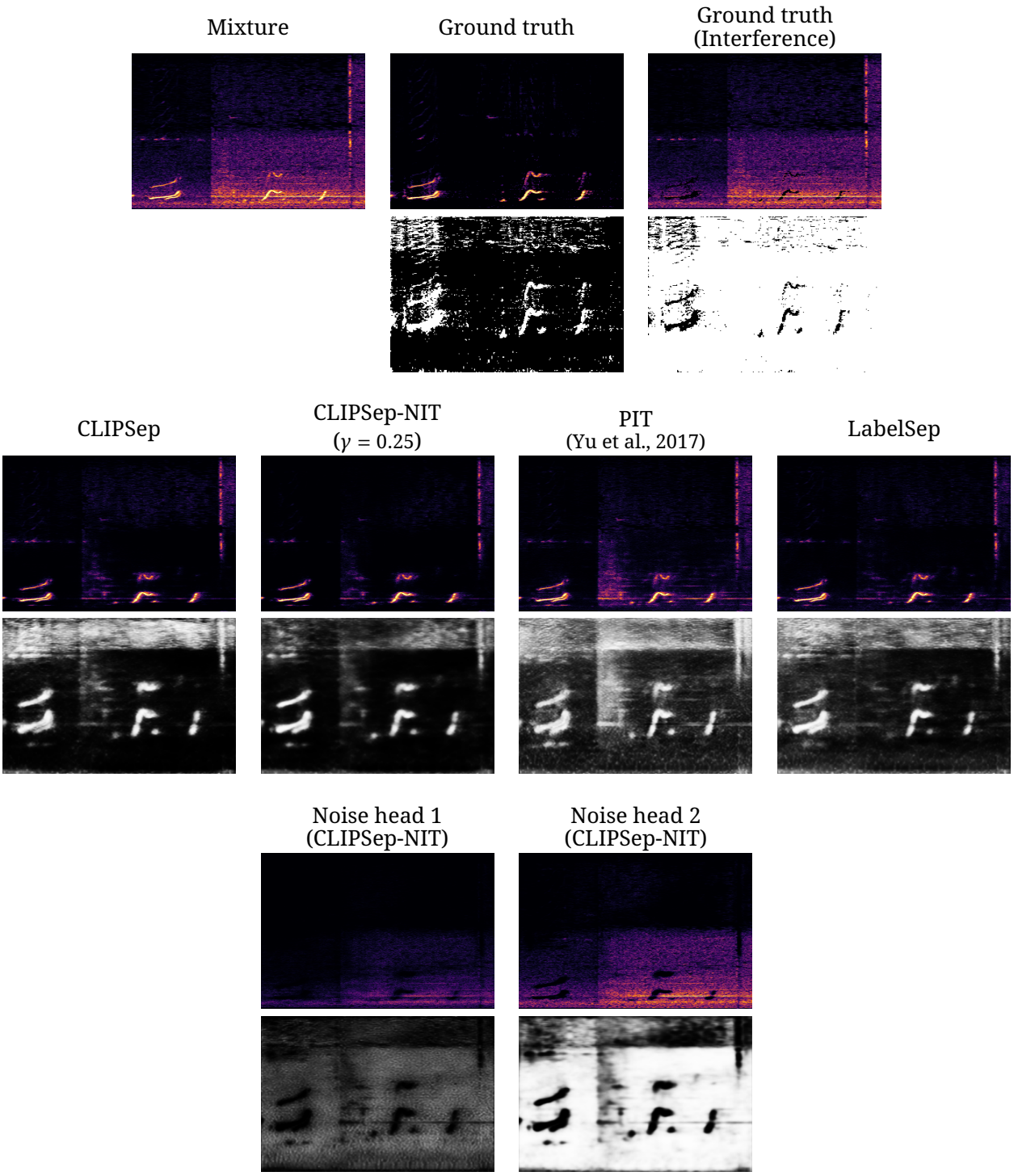


Figure 6.14. Example results on the VGGSound-Clean$^+$ dataset. Target source—"cat growling"; interference—"railroad car train wagon"; query—"*cat growling*". The spectrograms and masks are shown in the log and linear frequency scales, respectively.

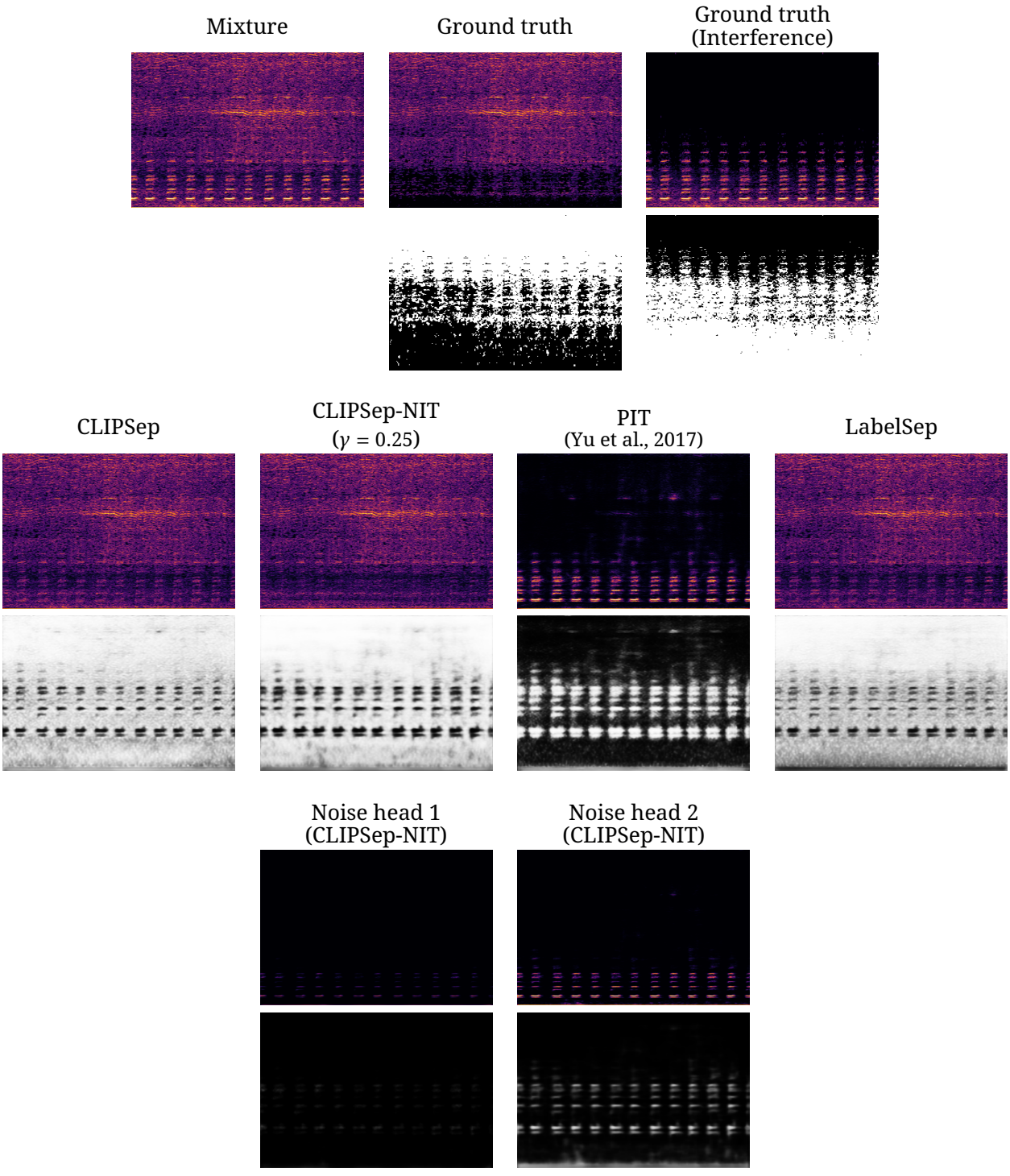


Figure 6.15. Example results on the VGGSound-Clean$^+$ dataset. Target source—"electric grinder grinding"; interference—"vehicle horn car horn honking"; query—"*electric grinder grinding*". The spectrograms and masks are shown in the log and linear frequency scales, respectively. Note that the PIT model requires a post-selection step to get the correct source. Without the post-selection step, the PIT model return the right source in only a 50% chance.

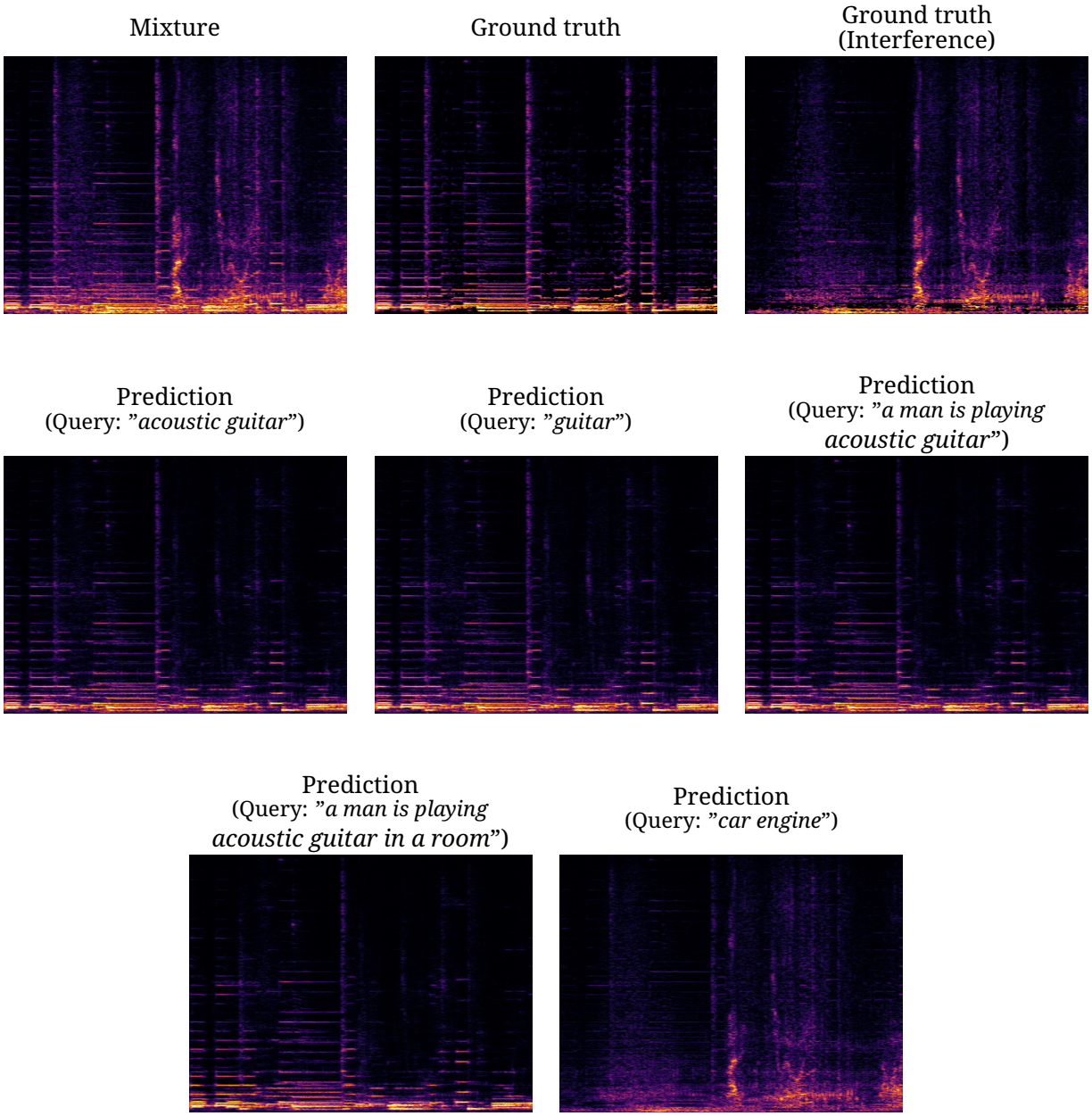


Figure 6.16. Query robustness experiment on the MUSIC$^+$ dataset. Target source—“acoustic guitar”; interference—“cheetah chirrup”. The spectrograms are shown in the log frequency scale.

Chapter 7

CLIPSonic: Text-to-Audio Synthesis with Unlabeled Videos and Pretrained Language-Vision Models

Abstract

Recent work has studied text-to-audio synthesis using large amounts of paired text-audio data. However, audio recordings with high-quality text annotations can be difficult to acquire. In this work, we approach text-to-audio synthesis using unlabeled videos and pretrained language-vision models. We propose to learn the desired text-audio correspondence by leveraging the visual modality as a bridge. We train a conditional diffusion model to generate the audio track of a video, given a video frame encoded by a pretrained contrastive language-image pretraining (CLIP) model. At test time, we first explore performing a zero-shot modality transfer and condition the diffusion model with a CLIP-encoded text query. However, we observe a noticeable performance drop with respect to image queries. To close this gap, we further adopt a pretrained diffusion prior model to generate a CLIP image embedding given a CLIP text embedding. Our results show the effectiveness of the proposed method, and that the pretrained diffusion prior can reduce the modality transfer gap. While we focus on text-to-audio synthesis, the proposed model can also generate audio from image queries, and it shows competitive performance against a state-of-the-art image-to-audio synthesis model in a subjective listening test. This study offers a new direction of approaching text-to-audio

synthesis that leverages the naturally-occurring audio-visual correspondence in videos and the power of pretrained language-vision models.

7.1 Introduction

With the advance of generative modeling (Radford et al., 2019; Ho et al., 2020; Rombach et al., 2022) and language-audio contrastive learning (Wu et al., 2023; Huang et al., 2022; Guzhov et al., 2022), various deep learning-based text-to-audio synthesis systems have recently emerged (Yang et al., 2022; Kreuk et al., 2023; Liu et al., 2023a; Huang et al., 2023b; Huang et al., 2023a; Agostinelli et al., 2023). However, these systems typically require a large amount of paired text-audio data for training. Despite extensive human annotation efforts, the current largest public text-audio dataset contains around 630 k text-audio pairs (Wu et al., 2023). Given the relative scarcity of text-audio data on the web as compared to text-image data, it remains unclear whether we can scale up text-audio datasets to a size comparable with large scale text-image datasets, e.g., the LAION-5B dataset (Schuhmann et al., 2022), which contains 5.85 billion text-image pairs. In this work, we approach text-to-audio synthesis without text-audio pairs through leveraging the naturally-occurring audio-visual correspondence in videos and the multimodal representation learned by pretrained language-vision models (see Figure 7.1).

The proposed CLIPSonic model is based on a conditional diffusion model (Nichol and Dhariwal, 2019), a constrastive language-image pretraining (CLIP) model (Radford et al., 2021), and a pretrained diffusion prior model (Ramesh et al., 2022), as illustrated in Figure 7.2. Given a video, CLIPSonic is trained to synthesize the mel spectrogram of the audio given a CLIP-encoded frame, randomly selected from the video. Since CLIP embeds images and texts into a cross-modal semantic space, CLIPSonic learns to map the CLIP embedding space to audio. At test time, we first explore performing a zero-shot modality transfer and conditioning the diffusion model directly with a CLIP-encoded text query. However, we observe in practice a noticeable performance drop with respect to image queries. To close this gap, we adopt a pretrained diffusion prior model to generate a CLIP image embedding given a CLIP text embedding. We note that our proposed system requires only 1) unlabeled

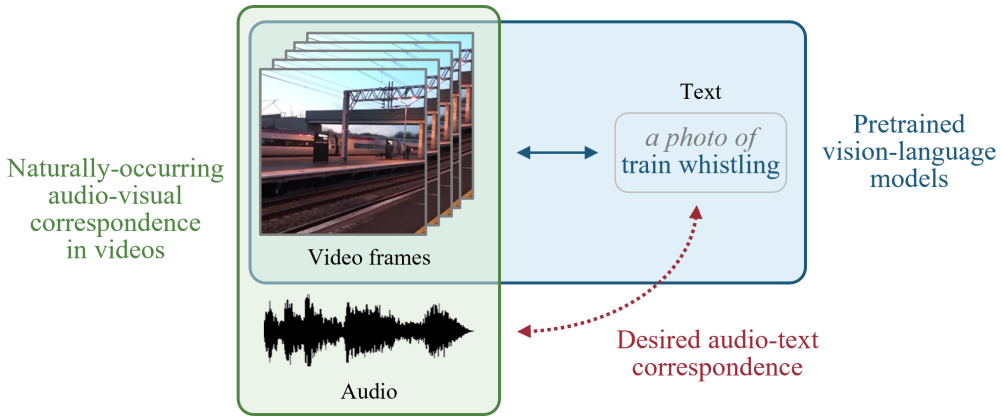


Figure 7.1. We learn the text-audio correspondence by leveraging the audio-visual correspondences in videos and the multimodal representation learned by pretrained language-vision models.

videos, for training the conditional diffusion model, and 2) image-text pairs, for pre-training the language-vision models. Through a subjective listening test and an objective evaluation, our experimental results demonstrate the effectiveness of the proposed method. Audio samples are available on our demo website.[1]

Our study differs from prior work in several ways. Existing text-to-audio models rely on large amounts of text-audio training pairs (Yang et al., 2022; Kreuk et al., 2023; Liu et al., 2023a; Huang et al., 2023b; Huang et al., 2023a; Agostinelli et al., 2023), whereas CLIPSonic learns text-queried audio synthesis without text-audio pairs. Prior work studied image-to-audio synthesis (Owens et al., 2016; Iashin and Rahtu, 2021; Sheffer and Adi, 2023), but they do not examine the zero-shot modality transfer between texts and images. CLIPSep (Dong et al., 2023d) and CLIPSynth (Dong et al., 2023c) propose to learn text-queried source separation and audio synthesis from unlabeled videos, respectively, but they do not address the issue of the zero-shot modality transfer gap. DALL-E 2 (Ramesh et al., 2022) proposes the diffusion prior model to address the zero-shot modality transfer gap in CLIP-based text-to-image synthesis, and we explore leveraging a pretrained diffusion prior model to transfer the knowledge learned from videos for text-to-audio synthesis. Other related works are AudioLDM (Liu et al., 2023a) and MusicLM (Agostinelli et al., 2023), which rely on language-audio models (Wu et al., 2023; Huang et al., 2022) to perform a zero-shot audio-to-text modality transfer, but such language-audio models are trained on audio-text pairs.

[1] https://salu133445.github.io/clipsonic/

7.2 CLIPSonic

In this section, we introduce the proposed CLIPSonic model for learning text-to-audio synthesis from unlabeled videos. As illustrated in Figure 7.2(a), CLIPSonic uses a mel spectrogram-based diffusion model for audio synthesis. We adopt the diffusion framework for its strong performance in audio synthesis (Kong et al., 2021; Pascual et al., 2023; Liu et al., 2023a). Given a video, CLIPSonic is trained to synthesize the mel spectrogram of the audio from the image in a randomly extracted video frame. Specifically, we first use a pretrained CLIP image encoder to encode the image into a query vector $\mathbf{q}_{img}$. Then, this query vector is used as a conditional signal to guide the diffusion model to generate a mel spectrogram $\hat{\mathbf{x}}_0$. We adopt a denoising diffusion probabilistic model (Nichol and Dhariwal, 2019) and classifier-free guidance (Jonathan Ho, 2021), which allows us to control the degree of conditioning signal through the guidance level variable w during inference.[2] The generated mel spectrograms are inverted back to waveforms using a separately-trained BigVGAN (Lee et al., 2023). We choose to perform diffusion on the mel spectrogram domain for its lower dimensionality than waveforms, and because BigVGAN shows good quality when synthesizing general audio from mel spectrograms.

CLIPSonic-ZS (zero-shot modality transfer). At inference time, we aim to leverage the language-vision embedding space learned by CLIP to achieve text-to-audio synthesis. CLIPSonic-ZS explores swapping the CLIP image embeddings for the CLIP text embeddings, as a way to use text queries in a zero-shot modality transfer setting. As illustrated in Figure 7.2(b), we use the CLIP text encoder to encode the input text query into a query vector $\mathbf{q}_{text}$, which is fed as a condition to the diffusion model. We refer to this model as CLIPSonic-ZS, where "ZS" stands for zero-shot modality transfer.

CLIPSonic-PD (pretrained diffusion prior). As to be shown in Section 7.4, we observe a modality gap between CLIP's text and image embedding spaces. Following DALL-E 2 (Ramesh et al., 2022), we explore relying on a diffusion prior model to bridge this gap. As illustrated in Figure 7.2(c), we first encode the input text query into a CLIP text

[2]We use the formulation: $\nabla_{\mathbf{x}} \log p_w(\mathbf{x} \mid \mathbf{q}) = (1 - w)\nabla_{\mathbf{x}} \log p(\mathbf{x}) + w\nabla_{\mathbf{x}} \log p(\mathbf{x} \mid \mathbf{q})$. A larger w leads to a stronger conditioning signal, and $w = 1$ corresponds to a conditional model without classifier-free guidance.

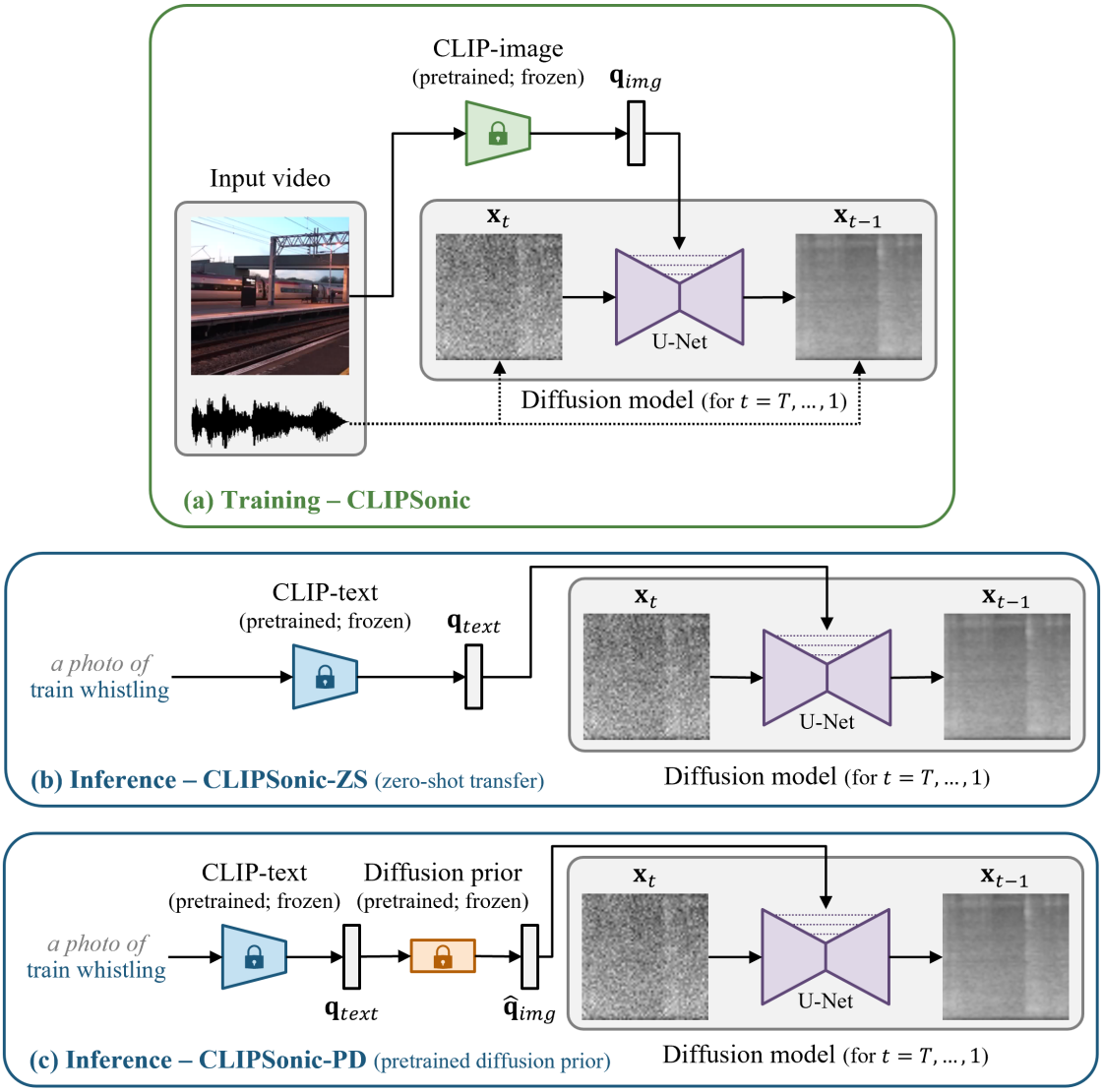


Figure 7.2. Proposed CLIPSonic model. During training, CLIPSonic learns to synthesize the audio track of a video given the image in a video frame. At inference time, we feed a text query in the form of "a photo of [label]" to approach text-to-audio synthesis or use a pretrained diffusion prior model to close the gap between the text queries (used for inference) and the image queries (used for training). $\mathbf{x}_t$ represents a noisy spectrogram at diffusion step t. The generated mel spectrogram $\hat{\mathbf{x}}_0$ is inverted back to waveform by a pretrained BigVGAN model (Lee et al., 2023).

embedding vector $\mathbf{q}_{text}$ and then generate a CLIP image embedding vector $\hat{\mathbf{q}}_{img}$ from $\mathbf{q}_{text}$ using the pretrained diffusion prior model. The generated query vector $\hat{\mathbf{q}}_{img}$ is then passed as the conditioning signal to the diffusion model. We refer to this model as CLIPSonic-PD (pretrained diffusion prior). Note that both CLIPSonic-ZS and CLIPSonic-PD require no text-audio pairs for training. Further, both the CLIP and diffusion prior models can be pretrained using only text-image pairs, hence suppressing the need for paired audio-text data.

CLIPSonic-IQ and CLIPSonic-SD. While here we focus on text-to-audio, CLIPSonic can also be used as an image-to-audio synthesis model by using $\mathbf{q}_{img}$ queries. We will refer to this variant as CLIPSonic-IQ (image-queried). Moreover, we find that it is possible to train the diffusion prior model from scratch on domain-specific datasets, and hence we also consider a variant called CLIPSonic-SD (supervised diffusion prior), where we train the diffusion prior model from scratch using text-image pairs in our datasets. As will be specified in Section 7.3, since the text data used to train the diffusion prior in CLIPSonic-SD comes from audio labels in this work, CLIPSonic-SD serves as an oracle model against CLIPSonic-PD. By comparing CLIPSonic-PD to CLIPSonic-SD, we intend to study the effectiveness of using a diffusion prior model pretrained on a massive amount of data against one trained on the target dataset.

7.3 Experimental setup

Data. We consider two datasets: VGGSound (Chen et al., 2020a) and MUSIC (Zhao et al., 2018). The VGGSound dataset consists of 171,899 10-sec YouTube videos, covering 310 classes of sounds in the wild, and we follow the train-test split provided with the dataset. The MUSIC dataset consists of 1,055 full-length YouTube videos of people playing a musical instrument, with 21 instrument types in total. We randomly split the dataset into a 9:1 train-test split. VGGSound represents a large, diverse dataset captured from unstructured sources in the wild, whereas MUSIC represents a small, curated dataset of a specific domain of interest. As both datasets come with only class labels, we convert such labels into pseudo text in the form of "a photo of [label]".

Baseline models. We compare CLIPSonic models against the following text-to-audio (TTA) and reconstruction models.

- **CLIP-TTA** is the supervised version of CLIPSonic where we use text-audio pairs for training. The pretrained CLIP-text embedding is used as conditioning.
- **CLAP-TTA** is the same as CLIP-TTA but uses pretrained CLAP-text embeddings (Wu et al., 2023), where we use a prompt in the form of "the sound of [label]". Unlike CLIP-text embeddings, CLAP-text embeddings are expected to encode audio-grounded features rather than visually-grounded features.
- **BigVGAN mel spectrogram reconstruction** are waveforms reconstructed from the ground-truth mel spectrograms by the BigVGAN model. This serves as an upper bound of spectrogram-based synthesis systems that use BigVGAN as the inversion model.

Implementation details. For mel spectrogram computation, we use a sampling rate of 16 kHz, a hop size of 512, an FFT filter size of 2048, and 64 mel bands. During training, we use mel spectrograms of size 64×64, which corresponds to two seconds of audio. For the diffusion model, we follow the network architecture proposed in (Nichol and Dhariwal, 2019) and use the open-source code in (*Improved Diffusion* n.d.). We use a cosine noise schedule with 4000 diffusion steps during training and 1000 steps at inference time. We use AdamW with a learning rate of 0.0001, a batch size of 32, and a dropout rate of 0.1 in classifier-free guidance. All diffusion models are trained for 200 k steps on MUSIC and 500 k steps on VGGSound using two NVIDIA RTX 2080 Ti GPUs, which takes a day on MUSIC and two days on VGGSound. For the pretrained CLIP model, we use the "ViT-L/14" version trained on 400 million image-text pairs (*CLIP* n.d.). We use a pretrained transformer-based diffusion prior model trained on 2 billion image-text pairs using the same backbone CLIP model (*DALLE2 PyTorch* n.d.[a]). For training the diffusion prior model CLIPSonic-SD from scratch, we follow the same architecture as in CLIPSonic-PD and use the code in (*DALLE2 PyTorch* n.d.[b]). We use AdamW with a learning rate of 0.0001 and a batch size of 32. The diffusion prior models are trained on MUSIC and VGGSound, respectively, until convergence at around 200 k steps, which takes a day on a NVIDIA RTX 2080 Ti GPU. For the CLAP model, we use the "630k-audioset-fusion" version released in (*CLAP* n.d.). For the BigVGAN model, we pretrain it on VGGSound for 500 k steps using the code in (*BigVGAN* n.d.) and use this pretrained version in all of our experiments.

Evaluation metrics. To compare the performance of our method against the baselines, we sample 512 audio samples from each model and compute the Fréchet audio distance (FAD) (Kilgour et al., 2019) and the CLAP score (Huang et al., 2023b; Wu et al., 2023). The FAD measures how close the generated audio samples are to the reference audio in terms of quality and diversity.[3] We adopt the open-source implementation provided in (*Frechet Audio Distance* n.d.) and use the VGGish (Hershey et al., 2017) as the backbone model for FAD. The CLAP score measures the relevance between the generated audio and the input query text, and it is formally defined as the cosine similarity between the CLAP embedding of the audio and that of the input text query.

Subjective test. We conduct a listening test to study the fidelity of the generated audio and their relevance to textual (text-to-audio) and visual (image-to-audio) prompts. We ask 21 expert listeners to rate the generated audio samples on a 1–5 scale in terms of fidelity and relevance. Fidelity experiments study the quality of the generated audio (without evaluating its semantic grounding) while relevance experiments study the semantic correspondence with respect to the prompt (without evaluating its audio quality). The audio samples used for this test are available on our demo website.[1]

7.4 Results

7.4.1 Objective Evaluation Results

Guidance. Figure 7.3 shows the results of the studied models as a function of the classifier-free guidance scale w. As noted in (Jonathan Ho, 2021) using conceptually-similar measures, the different curves between FAD and CLAP scores imply a trade-off between quality/diversity (represented by FAD) and query-sample relevance (represented by CLAP score). Noticeably, in terms of CLAP score, all models (except CLIPSonic-PD on MUSIC) outperform the BigVGAN reconstruction on both datasets (see Figure 7.3(b) and (d)). We attribute the higher CLAP scores to the classifier-free guidance as it is shown to improve adherence to the conditioning (Jonathan Ho, 2021) but at the cost of diversity—note the

[3] Following (Yang et al., 2022; Liu et al., 2023a) we also computed the Fréchet inception distance (FID) of the generated spectrograms, and found that the trend of FID aligned well with that of FAD. For brevity, we only report and discuss the FAD results.

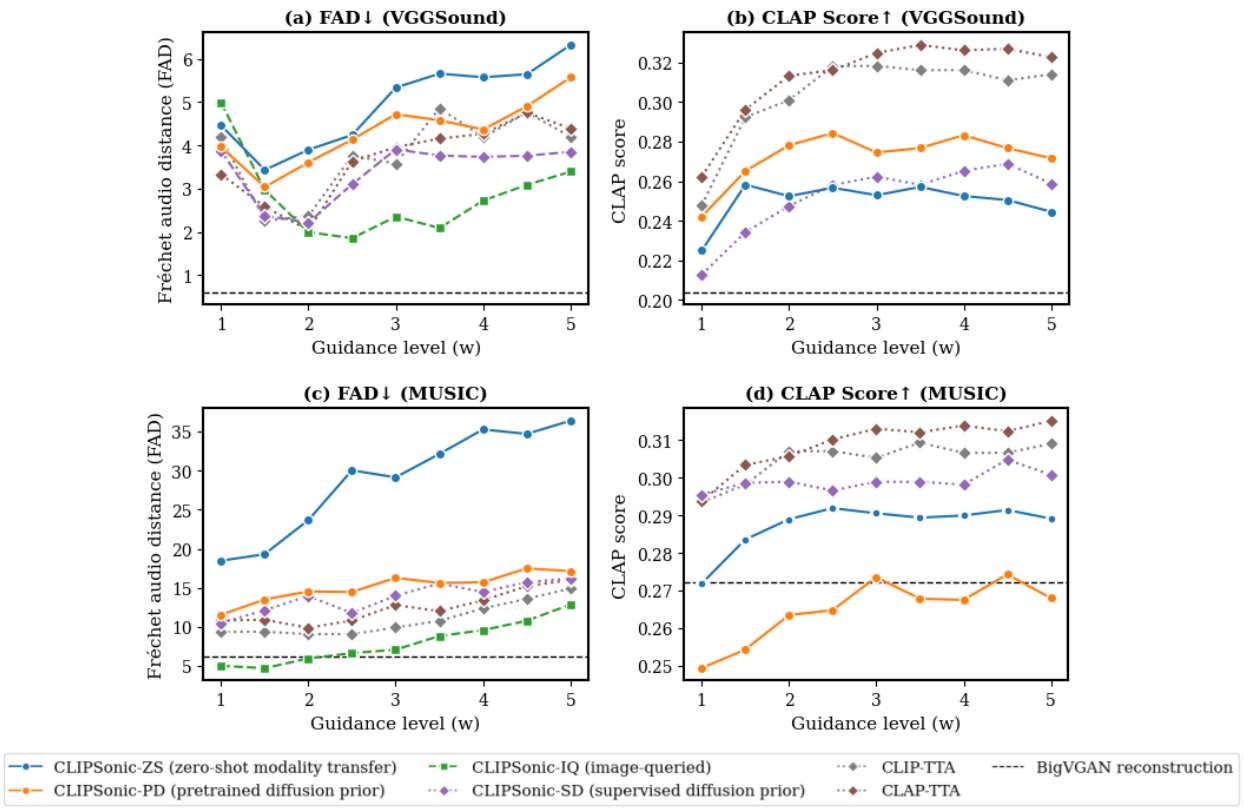


Figure 7.3. Objective evaluation results on VGGSound and MUSIC.

increasing FAD in Figure 7.3(a) and (c) as w increases. As such, practitioners can choose w based on their specific requirements. We use $w = 1.5$ as it offers a good balance between quality/diversity and relevance, and we report the results in Table 7.1.

Models without text-audio pairs. First, we discuss CLIPSonic models in Table 7.1 that do not use text-audio pairs during training. CLIPSonic-IQ (image-queried) achieves a strong performance on both datasets. Yet, when we switch to using text queries in a zero-shot setting with CLIPSonic-ZS, we observe a performance drop in terms of FAD on both datasets. This performance drop suggests a modality gap between CLIP's image (used during training) and text (used during inference) embedding spaces. In contrast, with the pretrained diffusion prior model, CLIPSonic-PD achieves a lower FAD than CLIPSonic-ZS across different w values (see also Figure 7.3). To further investigate this, we report in Table 7.2 the average cosine similarity between the query embedding ($\mathbf{q}_{text}$ or $\hat{\mathbf{q}}_{img}$) and the ground truth CLIP-image embedding $\mathbf{q}_{img}$. We note that CLIPSonic-ZS leads to a low cosine similarity, which supports our hypothesis that there is a modality gap in CLIP's embedding

Table 7.1. Evaluation results on VGGSound and MUSIC datasets, evaluated at $w = 1.5$.

Model	Without text-audio pairs	Query modality		VGGSound		MUSIC	
		Training	Inference	FAD↓	CLAP score↑	FAD↓	CLAP score↑
CLIPSonic-IQ (image-queried)	-	Image	Image	2.97	-	4.71	-
CLIPSonic-ZS (zero-shot modality transfer)	✓	Image	Text	3.43	0.258	19.30	0.284
CLIPSonic-PD (pretrained diffusion prior)	✓	Image	Text	3.04	0.265	13.51	0.254
CLIPSonic-SD (supervised diffusion prior)	✗	Image	Text	2.37	0.234	12.13	0.299
CLIP-TTA	✗	Text	Text	2.26	0.292	9.39	0.298
CLAP-TTA	✗	Text	Text	2.58	0.296	10.92	0.303
BigVGAN mel spectrogram reconstruction	-	-	-	0.60	0.204	6.21	0.272

Table 7.2. Cosine similarities between various query embeddings.

Model	Similarity computed	VGGSound	MUSIC
CLIPSonic-ZS	$\text{sim}(\mathbf{q}_{text}, \mathbf{q}_{img})$	0.205	0.245
CLIPSonic-PD	$\text{sim}(\hat{\mathbf{q}}_{img}, \mathbf{q}_{img})$	0.647	0.720
CLIPSonic-SD	$\text{sim}(\hat{\mathbf{q}}_{img}, \mathbf{q}_{img})$	0.711	0.824

space. In contrast, CLIPSonic-PD achieves a significantly higher cosine similarity, showing that the pretrained diffusion prior model can effectively bridge the modality gap. Moreover, while we observe a lower CLAP score for CLIPSonic-PD on MUSIC, we observe little difference in the relevance criterion in the listening test to be discussed in Section 7.4.2 (see Table 7.3), suggesting that all these models have passed a reasonable level of audio-text relevance.

Models using text-audio pairs. We now compare the baseline models that do use text-audio pairs for training against the previous CLIPSonic variants. First, we see that CLIPSonic-SD, with a diffusion prior trained directly on the target dataset, achieves a lower FAD than CLIPSonic-PD, which uses the pretrained diffusion prior. This is possibly due to the distribution mismatch between the target datasets and the LAION-2B dataset used to train the pretrained prior.[4] From Table 7.2, we can also see that CLIPSonic-SD can generate a CLIP-image embedding closer to the ground truth embedding on the target datasets than CLIPSonic-PD. Yet, in our subjective evaluation below we will see that CLIPSonic-PD still exhibits a favorable degree of generalization to downstream datasets since it consistently outperforms CLIPSonic-ZS. Moreover, we observe a gap between the performance of CLIPSonic-PD and

[4]We note that there is also a mismatch in the semantics of the textual queries, where the target datasets contain audio-specific labels while LAION-2B contains visually-grounded labels. However, the similar performance of CLIP-TTA and CLAP-TTA suggests that this is a minor effect.

Table 7.3. Listening test results for text-to-audio synthesis (MOS).

Model	VGGSound		MUSIC	
	Fidelity	Relevance	Fidelity	Relevance
CLIPSonic-ZS	2.55 ± 0.22	2.01 ± 0.27	2.98 ± 0.23	3.87 ± 0.24
CLIPSonic-PD	**3.04** ± **0.20**	2.86 ± 0.25	**3.67** ± **0.18**	3.91 ± 0.24
CLIPSonic-SD	2.96 ± 0.21	**3.49** ± **0.28**	3.36 ± 0.20	**4.07** ± **0.22**
Ground truth	3.78 ± 0.19	3.54 ± 0.29	3.90 ± 0.17	4.34 ± 0.18

Table 7.4. Listening test results for image-to-audio synthesis (MOS).

Model	Fidelity	Relevance
CLIPSonic-IQ (image-queried)	**3.29** ± **0.16**	3.80 ± 0.19
SpecVQGAN (Iashin and Rahtu, 2021)	2.15 ± 0.17	2.54 ± 0.23
IM2WAV (Sheffer and Adi, 2023)	2.19 ± 0.15	**3.90** ± **0.22**

that of CLIP-TTA and CLAP-TTA. However, we note that this is an unfair comparison as CLIP-TTA and CLAP-TTA are trained on audio-text pairs, while CLIPSonic-PD does not use audio-text pairs in training.

7.4.2 Subjective Listening Test Results

Text-to-audio synthesis. We conduct an ablation study to compare CLIPSonic-ZS, -PD and -SD variants on MUSIC and VGGSound. As shown in Table 7.3, CLIPSonic-ZS consistently underperforms, arguably because of the aforementioned mismatch between text and image embeddings. The two contributed variants, i.e., CLIPSonic-PD and -SD, consistently achieve higher MOS than CLIPSonic-ZS, both in terms of relevance and fidelity. Notably, the ground truth scores are relatively low (an MOS between 3 to 4), especially noticeable for VGGSound as it is noisier than the MUSIC dataset.

Image-to-audio synthesis. While our focus is to study text-to-audio synthesis, CLIPSonic-IQ can also generate audio from image queries. We compare it against SpecVQ-GAN (Iashin and Rahtu, 2021), a representative image-to-audio model, and IM2WAV (Sheffer and Adi, 2023), a state-of-the-art model for image-to-audio synthesis. All three models are trained on VGGSound and tested on out-of-distribution samples from IMAGEHEAR (Sheffer and Adi, 2023). The selected samples conform a challenging benchmark for us because they are 1) selected from IM2WAV's demo website and 2) out-of-distribution. As shown in Table 7.4,

CLIPSonic-IQ outperforms the state-of-the-art in fidelity, while remaining competitive in terms of relevance. The improved fidelity can possibly be attributed to the fact that we use a continuous representation (mel spectrogram) with a state-of-the-art inversion model (BigVGAN), as compared to the discrete VQ-VAE representation used in Im2Wav.

7.5 Conclusion

We explored approaching text-to-audio synthesis without text-audio pairs by using unlabeled videos and pretrained language-vision models. Through both objective and subjective evaluations, we showed that the proposed models can effectively learn text-to-audio synthesis without text-audio pairs, and the pretrained diffusion prior can reduce the modality transfer gap caused by the mismatch between CLIP's image (used for training) and text (used for inference) embedding spaces. Moreover, in a subjective listening test, the image-to-audio synthesis model that we base our modality transfer upon achieves competitive performance against a state-of-the-art image-to-audio synthesis model. Finally, we argue that images provide rich conditioning signals for audio synthesis, and leveraging such rich signals to improve text-to-audio synthesis is a promising research direction. Along this direction, CLIPSonic represents an example using videos and pretrained language-vision models. For future work, we intend to scale up the proposed method to a larger amount of videos, and explore using tri-modal audio-vision-language models (Guzhov et al., 2022; Wu et al., 2022a; Rouditchenko et al., 2021).

* * *

This chapter, in full, is a reprint of the material as it appears in "CLIPSonic: Text-to-Audio Synthesis with Unlabeled Videos and Pretrained Language-Vision Models" by Hao-Wen Dong, Xiaoyu Liu, Jordi Pons, Gautam Bhattacharya, Santiago Pascual, Joan Serrà, Taylor Berg-Kirkpatrick and Julian McAuley, which was published in the Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA) in 2023. The dissertation author was the primary investigator and author of this paper.

Appendices

7.A Implementation Details of the Diffusion Prior Models

The diffusion prior models used in this paper are based on the open-source implementation of DALL-E 2 in (*DALLE2 PyTorch* n.d.[b]). Specifically, the input to the models is a sequence formed in the order of the encoded CLIP text tokens, the CLIP text embedding, the diffusion step embedding, the noised CLIP image embedding, and a learnable final input embedding. This sequence is fed to a 12-layer transformer consisting of causal multi-head self-attention and feed-forward networks. The last layer's final output vector corresponding to the final input embedding serves as the prediction of the target CLIP image embedding.

For the diffusion prior model used in CLIPSonic-SD, we use a cosine noise schedule with 1000 diffusion steps during training, and 64 steps at inference time. At each diffusion step during training, we minimize the mean squared error between the predicted and the target CLIP image embeddings. Based on DALL-E 2 (Ramesh et al., 2022), we also explore the classifier-free guidance for training the diffusion prior models by randomly replacing the encoded text tokens and the CLIP text embedding with learnable placeholders 10% of the time. However, at inference time, we empirically find that using no guidance yields the best results. At inference time, for each CLIP text embedding, we generate two CLIP image embeddings from the diffusion prior model, and select the one with a higher cosine similarity to the CLIP text embedding. To train the model, we use the AdamW optimizer with a learning rate of 0.0001, a batch size of 32, a weight decay of 0.06, and we apply an exponential moving average on the model parameters with a decay factor of 0.9999. The diffusion prior models in CLIPSonic-SD are trained on MUSIC and VGGSound independently until convergence at around 200 k steps.

7.B CLAP Scores for BigVGAN Reconstructions

In Figure 7.3, we observe that the CLAP scores of the BigVGAN reconstruction using the ground truth mel spectrogram, in many cases, are lower than those of the proposed systems, which indicates lower relevance between the ground truth audio and the text query.

Table 7.5. CLAP scores computed on BigVGAN reconstructions using a sliding window.

Window size	Mode	VGGSound	MUSIC
4 sec	Max	0.273	0.280
4 sec	Mean	0.195	0.234
4 sec	Min	0.111	0.185
10 sec	-	0.204	0.272

In order to adhere to the length of the test data, the BigVGAN CLAP scores are obtained based on the entire 10-sec audio samples. However, empirical listening finds that some segments within the 10-sec samples correspond poorly to the text queries. To further investigate the correspondence, We also compute the BigVGAN CLAP scores using a 4-sec sliding window (consistent with the synthesized sample length) with a hop size of 0.5 sec, and report the maximum, mean, and the minimum scores over all 4-sec segments within a 10-sec sample as the overall score of that sample.

As shown in Table 7.5, the maximum scores on both datasets are higher than the rest, which supports our observation by listening. On VGGSound, the maximum CLAP score also exceeds those of CLIPSonic-ZS, CLIPSonic-PD, and CLIPSonic-SD (see Table 7.1). On MUSIC, there is a smaller gap between the maximum CLAP score and that obtained using the entire 10-sec audio, indicating a more uniform relevance level within a sample. However, the studied models trained on MUSIC still outperform the BigVGAN reconstruction in terms of the maximum CLAP score (except for CLIPSonic-PD, and CLIPSonic-ZS without using the classifier free guidance, see Figure 7.3). In addition to the contribution of the classifier free guidance (Section 7.4.1), the remaining reason requires further investigation. Possible directions include manually inspecting and removing samples with poor audio-text correspondence, and also finetuning CLAP on MUSIC.

7.C Limitations

We observe some limitations of the proposed method. First, as CLIPSonic is conditioned on the CLIP embedding of a single video frame, it is not readily applicable to handle more complex text queries that involve sequences of events or dynamic interactions between objects. A more powerful language-vision model that can understand videos is required to

apply our proposed method to leverage the rich temporal information in videos. Second, since the conditioning signals are extracted from videos, CLIPSonic cannot learn audio concepts that have little meaning in the visual domain, such as pitch, prosody, genre, and tempo. This represents one of the fundamental limitations of approaches that use the visual domain as a bridge to learn the text-audio correspondence. Finally, CLIPSonic offers limited controllability in generating semantically complex audio, such as speech or music given specific words or scores, respectively. However, the proposed method may serve as a pretraining approach for training language-audio models, where we can first pretrain a language-audio model on a large dataset with only unlabeled videos and later finetune the model on a small dataset with audio-text pairs.

Chapter 8

Conclusion

Works of art make rules;
rules do not make works of art.
—Claude Debussy

In this dissertation, I have presented my research in three directions: *multitrack music generation* (Chapters 2 and 3), *assistive music creation tools* (Chapters 4 and 5) and *multimodal learning for audio and music* (Chapters 6 and 7).

Multitrack music generation. In Chapter 2, I presented MusPy, a new Python library for symbolic music generation (Dong et al., 2020). With MusPy, I conducted the first large-scale experiment that measures the cross-dataset generalizability of music generation models, a process which is made easier by MusPy's dataset management system. In Chapter 3, I proposed a new deep learning model for generating multi-instrument music that achieves comparable performance against state-of-the-art systems (Dong et al., 2023a). Moreover, I also presented the first systematic analysis of musical self-attention and showed that our proposed model learns relative self-attention in certain aspects of music such as beats, positions and pitches.

Assistive music creation tools. In Chapter 4, I developed the first deep learning model for automatic instrumentation that can find applications in assistive composing tools and intelligent keyboards (Dong et al., 2021). I also demonstrated the potential for our proposed models to produce alternative convincing instrumentations for an existing arrangement. In Chapter 5, I presented the first deep neural network-based polyphonic

synthesizer that can synthesize a score into a natural, expressive performance, achieving competitive quality against the baseline model, a conditional generative audio model, in terms of pitch accuracy, timbre and noise level (Dong et al., 2022).

Multimodal learning for audio and music. In Chapter 6, I developed the first text-queried sound separation model that can be trained without any text-audio pairs, achieving competitive performance against a supervised model in some settings (Dong et al., 2023d). In Chapter 7, I presented the first text-to-audio synthesis model that requires no text-audio pairs during training (Dong et al., 2023b). Further, while we focus on text-to-audio synthesis, the proposed model can also generate audio from image queries, and it achieves competitive performance against a state-of-the-art image-to-audio synthesis model in the subjective listening test.

8.1 Future Directions

Multimodal generative AI with music and audio. Multimodal content generation has quickly become the next frontier of generative AI. Many recently-released large pretrained multimodal contrastive models lay the foundations for exciting creative applications to film, video and audiobook generation. I am particularly interested in working on multimodal generative models for background music and sound effect generation for videos, audiobooks and games. I would also like to explore fusing multiple controlling signals from different modalities (e.g., text, image, video, audio, emotion measurements, etc.) for controllable music and audio generation. My long-term goal along this direction is to *develop next-generation interfaces for music and audio editing equipped with intuitive multimodal controls*. I will seek collaborations with other faculty members in comp uter vision and natural language processing to pursue research along this direction.

Interactive AI tools for music and audio production. While recent deep learning-based music and audio generation systems can create short, plausible music excerpts, they offer limited usability and controllability for humans to step in. Instead of building a fully-automated generation system, I want to *develop interactable music and audio production tools equipped with intermediate controls that humans can interact with*. For example, recently we

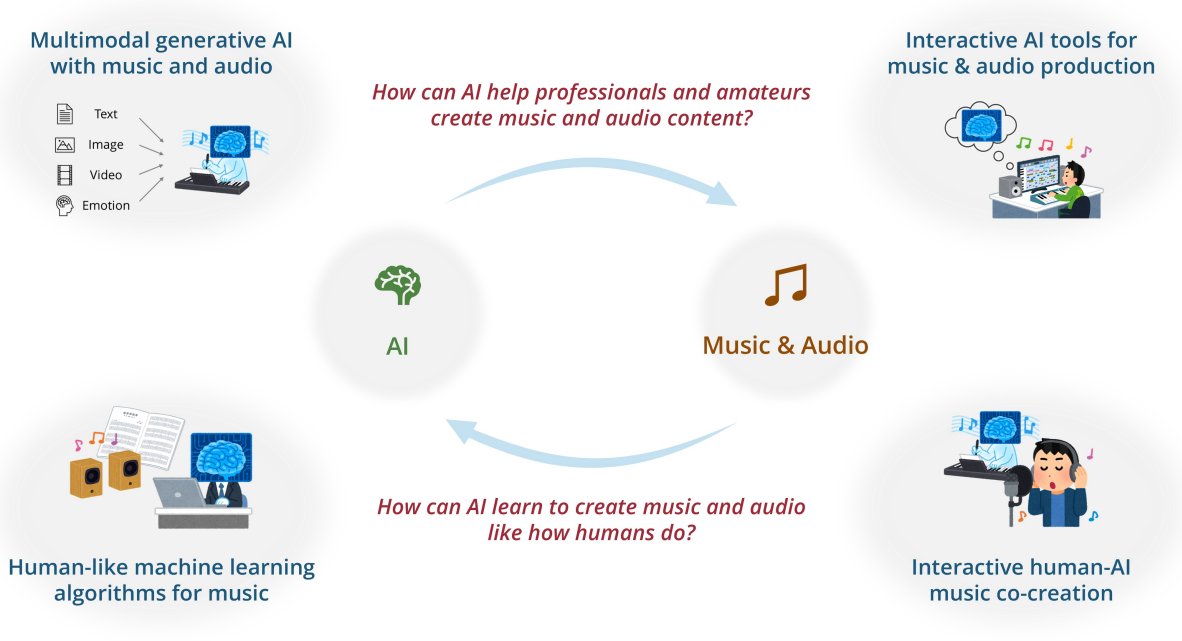


Figure 8.1. An overview of my future research directions.

have seen preliminary results on leveraging large language models (LLMs) for building a compositional, human-usable audio generation system (Liu et al., 2023b). Moreover, subject-driven personalization (Ruiz et al., 2023) and instruction-based editing (Brooks et al., 2023) has lately been attracting attentions in the image generation community, and I would like to explore opportunities in these directions with an eye to integrate these tools into professional creative workflows in music and audio production software. I will seek collaborations with other faculty members in human-computer interaction to explore new creative interfaces for music and audio production.

Human-like machine leaning algorithms for music. Richard Feynman once said: "What I cannot create, I do not understand." This echos my motivations of pursuing music generation research—generation represents the highest-level of understanding. A long-term direction of my research is to *develop human-like machine learning algorithms that can learn to create music in a way similar to how humans learn music*. For example, existing data-driven approaches for music generation usually rely on *reading* a large collection of musical scores. Unlike machines, however, humans learn music mostly through listening and practicing music rather than reading scores over and over again. I am thus interested

in exploring novel machine learning models that can learn symbolic music composition through listening to a large collection of musical audio data. Some recent work (Castellon et al., 2021) has shown preliminary results towards this direction. In my view, music possesses a unique complexity that might lead to new breakthroughs in AI and contribute towards the long-lasting pursuit of artificial general intelligence.

8.2 Broader Impacts

I envision my research to be integrated into the audio content creation workflow for professional artists and amateurs. Through providing new tools and interfaces to make music, my research could lower the barrier for music composition and empower novices to create their own music. Moreover, it could provide content creators (e.g., TikTokers, YouTubers and Twitch streamers) with royalty-free materials to avoid unintended copyright infringement. My research could also find applications in music education and therapy, where creating personalized courses can be costly. Finally, we could gain insights into the future of human-AI music co-creation though the interactions between human and automatic music composition systems. I envision this to foster the discussions in human-AI relationships in other fields.

Without deviation from the norm,

progress is not possible.

—Frank Zappa

Bibliography

Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng (2016). "TensorFlow: A system for large-scale machine learning." *USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (cited on pages 9, 12, 45).

Yuzo Abe, Yuki Murakami, and Masanobu Miura (2012). "Automatic arrangement for the bass guitar in popular music using principle component analysis." *Acoustical Science and Technology* 33.4, pp. 229–238 (cited on page 39).

Triantafyllos Afouras, Andrew Owens, Joon Son Chung, and Andrew Zisserman (2020). "Self-Supervised Learning of Audio-Visual Objects from Video." *European Conference on Computer Vision (ECCV)* (cited on page 71).

Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank (2023). "MusicLM: Generating Music From Text." *arXiv preprint arXiv:2302.03917* (cited on pages 99, 100).

Relja Arandjelović and Andrew Zisserman (2017a). "Look, listen and learn." *International Conference on Computer Vision (ICCV)* (cited on page 84).

Relja Arandjelović and Andrew Zisserman (2017b). "Objects that Sound." *European Conference on Computer Vision (ECCV)* (cited on page 84).

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton (2016). "Layer Normalization." *NeurIPS 2016 Deep Learning Symposium* (cited on page 45).

Francis R. Bach and Michael I. Jordan (2005). "Blind one-microphone speech separation: A spectral learning approach." *Advances in Neural Information Processing Systems (NeurIPS)* (cited on page 69).

Renée Baillargeon (2002). "The acquisition of physical knowledge in infancy: A summary in eight lessons." *Blackwell handbook of childhood cognitive development* (cited on page 69).

BigVGAN (n.d.). URL: https://github.com/NVIDIA/BigVGAN (cited on page 104).

Rachel M Bittner, Magdalena Fuentes, David Rubinstein, Andreas Jansson, Keunwoo Choi, and Thor Kell (2019). “mirdata: Software for Reproducible Usage of Datasets.” *International Society for Music Information Retrieval Conference (ISMIR)* (cited on page 8).

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent (2012). “Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription.” *International Conference on Machine Learning (ICML)* (cited on pages 8, 12, 13).

Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet (2017). “Deep Learning Techniques for Music Generation–A Survey.” *arXiv preprint arXiv:1709.01620* (cited on pages 1, 3, 7, 8, 12, 23, 25, 40).

Tim Brooks, Aleksander Holynski, and Alexei A. Efros (2023). “InstructPix2Pix: Learning to Follow Image Editing Instructions.” *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cited on page 115).

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei (2020). “Language Models are Few-Shot Learners.” *Advances in Neural Information Processing Systems (NeurIPS)* (cited on pages 28, 80).

Emilios Cambouropoulos (2006). “‘Voice’ separation: theoretical, perceptual and computational perspectives.” *International Conference on Music Perception & Cognition (ICMPC)* (cited on page 39).

Emilios Cambouropoulos (2008). “Voice and Stream: Perceptual and Computational Modeling of Voice Separation.” *Music Perception* 26.1 (cited on page 39).

Rodrigo Castellon, Chris Donahue, and Percy Liang (2021). “Codified audio language modeling learns useful representations for music information retrieval.” *International Society for Music Information Retrieval Conference (ISMIR)* (cited on page 116).

Chin-Jui Chang, Chun-Yi Lee, and Yi-Hsuan Yang (2021). “Variable-Length Music Score Infilling via XLNet and Musically Specialized Positional Encoding.” *International Society for Music Information Retrieval Conference (ISMIR)* (cited on page 26).

Honglie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman (2020a). “VGGSound: A Large-scale Audio-Visual Dataset.” *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cited on pages 79, 103).

Yu-Hua Chen, Yu-Siang Huang, Wen-Yi Hsiao, and Yi-Hsuan Yang (2020b). “Automatic Composition of Guitar Tabs by Transformers and Groove Modeling.” *International Society for Music Information Retrieval Conference (ISMIR)* (cited on page 26).

Ke Chen, Xingjian Du, Bilei Zhu, Zejun Ma, Taylor Berg-Kirkpatrick, and Shlomo Dubnov (2022). "Zero-shot Audio Source Separation through Query-based Learning from Weakly-labeled Data." *AAAI Conference on Artificial Intelligence (AAAI)* (cited on page 72).

Tsung-Ping Chen and Li Su (2021). "Attend to Chords: Improving Harmonic Analysis of Symbolic Music Using Transformer-Based Models." *Transactions of the International Society for Music Information Retrieval (TISMIR)* 4.1 (cited on pages 24, 32).

E Colin Cherry (1953). "Some experiments on the recognition of speech, with one and with two ears." *The Journal of the Acoustical Society of America* 25 (5) (cited on page 69).

Chung-Ming Chien, Jheng-Hao Lin, Chien-Yu Huang, Po-Chun Hsu, and Hung-Yi Lee (2021). "Investigating on Incorporating Pretrained and Learnable Speaker Representations for Multi-Speaker Multi-Style Text-to-Speech." *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cited on pages 59, 66, 67).

Shih-Chuan Chiu, Man-Kwan Shan, and Jiun-Long Huang (2009). "Automatic system for the arrangement of piano reductions." *IEEE International Symposium on Multimedia (ISM)* (cited on page 39).

Kyunghyun Cho, B van Merrienboer, Caglar Gulcehre, F Bougares, H Schwenk, and Yoshua Bengio (2014). "Learning phrase representations using RNN encoder-decoder for statistical machine translation." *Conference on Empirical Methods in Natural Language Processing (EMNLP)* (cited on page 17).

Yi-Hui Chou, I-Chun Chen, Chin-Jui Chang, Joann Ching, and Yi-Hsuan Yang (2021). "MidiBERT-Piano: Large-scale Pre-training for Symbolic Music Understanding." *arXiv preprint arXiv:2107.05223* (cited on page 26).

CLAP (n.d.). URL: https://github.com/LAION-AI/CLAP (cited on page 104).

CLIP (n.d.). URL: https://github.com/openai/CLIP (cited on page 104).

Léopold Crestel and Philippe Esling (2016). "Live orchestral piano, a system for real-time orchestral music generation." *arXiv preprint arXiv:1609.01203* (cited on page 39).

Léopold Crestel, Philippe Esling, Lena Heng, and Stephen McAdams (2017). "A database linking piano and orchestral MIDI scores with application to automatic projective orchestration." *International Society for Music Information Retrieval Conference (ISMIR)* (cited on pages 28, 30).

Michael Scott Cuthbert and Christopher Ariza (2010). "Music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data." *International Society for Music Information Retrieval Conference (ISMIR)* (cited on pages 9, 11, 13, 43, 44).

DALLE2 PyTorch (n.d.[a]). URL: https://huggingface.co/laion/DALLE2-PyTorch (cited on page 104).

DALLE2 PyTorch (n.d.[b]). URL: https://github.com/lucidrains/DALLE2-pytorch (cited on pages 104, 110).

Roger B. Dannenberg (1993). "A Brief Survey of Music Representation Issues, Techniques, and Systems." *Computer Music Journal* 17.3, pp. 20–30 (cited on page 8).

Brecht De Man, Ryan Stables, and Joshua D. Reiss (2019). *Intelligent Music Production*. Routledge (cited on page 1).

Alexandre Défossez, Neil Zeghidour, Nicolas Usunier, Léon Bottou, and Francis Bach (2018). "SING: Symbol-to-Instrument Neural Generator." *Advances in Neural Information Processing Systems (NeurIPS)* (cited on page 53).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)* (cited on page 81).

Chris Donahue, Huanru Henry Mao, Yiting Ethan Li, Garrison W. Cottrell, and Julian McAuley (2019). "LakhNES: Improving multi-instrumental music generation with cross-domain pre-training." *International Society for Music Information Retrieval Conference (ISMIR)* (cited on pages 9, 10, 14, 22, 24, 25, 40, 51).

Chris Donahue, Huanru Henry Mao, and Julian McAuley (2018). "The NES Music Database: A multi-instrumental dataset with expressive performance attributes." *International Society for Music Information Retrieval Conference (ISMIR)* (cited on pages 13, 43).

Hao-Wen Dong, Ke Chen, Shlomo Dubnov, Julian McAuley, and Taylor Berg-Kirkpatrick (2023a). "Multitrack Music Transformer." *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (cited on pages 2, 4, 6, 113).

Hao-Wen Dong, Ke Chen, Julian McAuley, and Taylor Berg-Kirkpatrick (2020). "MusPy: A Toolkit for Symbolic Music Generation." *International Society for Music Information Retrieval Conference (ISMIR)* (cited on pages 3, 4, 6, 30, 44, 59, 113).

Hao-Wen Dong, Chris Donahue, Taylor Berg-Kirkpatrick, and Julian McAuley (2021). "Towards Automatic Instrumentation by Learning to Separate Parts in Symbolic Multitrack Music." *International Society for Music Information Retrieval Conference (ISMIR)* (cited on pages 2, 4, 6, 113).

Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang (2017). "MuseGAN: Demonstration of a Convolutional GAN Based Model for Generating Multi-track Piano-rolls." *ISMIR Late-Breaking Demos* (cited on page 2).

Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang (2018a). "MuseGAN: Multi-Track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment." *AAAI Conference on Artificial Intelligence (AAAI)* (cited on pages 2, 9, 14, 25, 44).

Hao-Wen Dong, Wen-Yi Hsiao, and Yi-Hsuan Yang (2018b). “Pypianoroll: Open Source Python Package for Handling Multitrack Pianoroll.” *ISMIR Late-Breaking Demos* (cited on pages 3, 12).

Hao-Wen Dong, Xiaoyu Liu, Jordi Pons, Gautam Bhattacharya, Santiago Pascual, Joan Serrà, Taylor Berg-Kirkpatrick, and Julian McAuley (2023b). “CLIPSonic: Text-to-Audio Synthesis with Unlabeled Videos and Pretrained Language-Vision Models.” *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)* (cited on pages 3, 5, 6, 114).

Hao-Wen Dong, Gunnar A. Sigurdsson, Chenyang Tao, Jiun-Yu Kao, Yu-Hsiang Lin, Anjali Narayan-Chen, Arpit Gupta, Tagyoung Chung, Jing Huang, Nanyun Peng, and Wenbo Zhao (2023c). “CLIPSynth: Learning Text-to-audio Synthesis from Videos using CLIP and Diffusion Models.” *CVPR Workshop on Sight and Sound (WSS)* (cited on pages 3, 5, 6, 100).

Hao-Wen Dong, Naoya Takahashi, Yuki Mitsufuji, Julian McAuley, and Taylor Berg-Kirkpatrick (2023d). “CLIPSep: Learning Text-queried Sound Separation with Noisy Unlabeled Videos.” *International Conference on Learning Representations (ICLR)* (cited on pages 3, 5, 6, 100, 114).

Hao-Wen Dong and Yi-Hsuan Yang (2018). “Convolutional Generative Adversarial Networks with Binary Neurons for Polyphonic Music Generation.” *International Society for Music Information Retrieval Conference (ISMIR)* (cited on pages 2, 25).

Hao-Wen Dong, Cong Zhou, Taylor Berg-Kirkpatrick, and Julian McAuley (2022). “Deep Performer: Score-to-Audio Music Performance Synthesis.” *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (cited on pages 2, 5, 6, 114).

Douglas Eck and Jürgen Schmidhuber (2002). “Finding Temporal Structure in Music: Blues Improvisation with LSTM Recurrent Networks.” *IEEE Workshop on Neural Networks for Signal Processing (NNSP)* (cited on pages 8, 12).

Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts (2019). “GANSynth: Adversarial Neural Audio Synthesis.” *International Conference on Learning Representations (ICLR)* (cited on page 53).

Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts (2020). “DDSP: Differentiable Digital Signal Processing.” *International Conference on Learning Representations (ICLR)* (cited on pages 53, 54, 62).

Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi (2017). “Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders.” *International Conference on Machine Learning (ICML)* (cited on page 53).

Jeff Ens and Philippe Pasquier (2020). “MMM: Exploring Conditional Multi-Track Music Generation with the Transformer.” *arXiv preprint arXiv:2008.06048* (cited on pages 24, 25, 28, 29, 31, 32, 40).

Ariel Ephrat, Inbar Mosseri, Oran Lang, Tali Dekel, Kevin Wilson, Avinatan Hassidim, William T Freeman, and Michael Rubinstein (2019). “Looking to Listen at the Cocktail Party: A Speaker-Independent Audio-Visual Model for Speech Separation.” *ACM Transactions on Graphics* 37 (4) (cited on page 71).

Essen Folk Song Database (n.d.). URL: https://ifdo.ca/~seymour/runabc/esac/esacdatabase.html (cited on page 13).

FluidSynth (n.d.). URL: https://www.fluidsynth.org/ (cited on page 57).

Frechet Audio Distance (n.d.). URL: https://github.com/gudgud96/frechet-audio-distance (cited on page 105).

Ruohan Gao, Rogerio Feris, and Kristen Grauman (2018). “Learning to Separate Object Sounds by Watching Unlabeled Video.” *European Conference on Computer Vision (ECCV)* (cited on page 71).

Beat Gfeller, Dominik Roblek, and Marco Tagliasacchi (2021). “One-Shot Conditional Audio Filtering of Arbitrary Sounds.” *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cited on page 72).

Michael Good (2001). “MusicXML for Notation and Analysis.” *The Virtual Score: Representation, Retrieval, Restoration*. Chap. 8, pp. 113–124 (cited on page 8).

Patrick Gray and Razvan Bunescu (2016). “A Neural Greedy Model for Voice Separation in Symbolic Music.” *International Society for Music Information Retrieval Conference (ISMIR)* (cited on page 39).

Patrick Gray and Razvan Bunescu (2020). “From Note-Level to Chord-Level Neural Network Models for Voice Separation in Symbolic Music.” *arXiv preprint arXiv:2011.03028* (cited on page 39).

Nicolas Guiomard-Kagan, Mathieu Giraud, Richard Groult, and Florence Levé (2015). “Comparing Voice and Stream Segmentation Algorithms.” *International Society for Music Information Retrieval Conference (ISMIR)* (cited on page 39).

Andrey Guzhov, Federico Raue, Jörn Hees, and Andreas Dengel (2022). “AudioCLIP: Extending CLIP to Image, Text and Audio.” *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cited on pages 73, 99, 109).

Aristotelis Hadjakos, Simon Waloschek, and Alexander Leemhuis (2019). “Detecting Hands from Piano MIDI Data.” *Mensch und Computer 2019 - Workshopband* (cited on page 39).

Andrew Hankinson, Perry Roland, and Ichiro Fujinaga (2011). “The Music Encoding Initiative as a Document-Encoding Framework.” *International Society for Music Information Retrieval Conference (ISMIR)* (cited on page 8).

Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck (2019). “Enabling Factorized Piano

Music Modeling and Generation with the MAESTRO Dataset." *International Conference on Learning Representations (ICLR)* (cited on pages 13, 53, 54, 57).

Ben Hayes, Charalampos Saitis, and George Fazekas (2021). "Neural Waveshaping Synthesis." *International Society for Music Information Retrieval Conference (ISMIR)* (cited on pages 53, 54).

Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. (2017). "CNN Architectures for Large-scale Audio Classification." *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cited on page 105).

Walter B. Hewlett (1997). "MuseData: Multipurpose Representation." *Beyond MIDI: The Handbook of Musical Codes*. MIT Press. Chap. 27, pp. 402–447 (cited on page 8).

Jonathan Ho, Ajay Jain, and Pieter Abbeel (2020). "Denoising Diffusion Probabilistic Models." *Advances in Neural Information Processing Systems (NeurIPS)* (cited on page 99).

Sepp Hochreiter and Jürgen Schmidhuber (1997). "Long short-term memory." *Neural Computation* 9.8, pp. 1735–1780 (cited on pages 17, 38, 41).

Gen Hori, Hirokazu Kameoka, and Shigeki Sagayama (2013). "Input-output HMM applied to automatic arrangement for guitars." *Information and Media Technologies* 8.2, pp. 477–484 (cited on page 39).

Gen Hori, Yuma Yoshinaga, Satoru Fukayama, Hirokazu Kameoka, and Shigeki Sagayama (2012). "Automatic arrangement for guitars using hidden Markov model." *Sound and Music Computing Conference (SMC)* (cited on page 39).

Wen-Yi Hsiao, Jen-Yu Liu, Yin-Cheng Yeh, and Yi-Hsuan Yang (2021). "Compound Word Transformer: Learning to Compose Full-Song Music over Dynamic Directed Hypergraphs." *AAAI Conference on Artificial Intelligence (AAAI)* (cited on pages 23, 25, 26, 28, 38, 40).

Anna Huang, Monica Dinculescu, Ashish Vaswani, and Douglas Eck (2018). "Visualizing Music Self-Attention." *Advances in Neural Information Processing Systems (NeurIPS) Workshop on Interpretability and Robustness in Audio, Speech, and Language* (cited on pages 24, 32).

Cheng-Zhi Anna Huang, Hendrik Vincent Koops, Ed NewtonRex, Monica Dinculescu, and Carrie J. Cai (2020). "AI Song Contest: Human-AI Co-Creation in Songwriting." *International Society for Music Information Retrieval Conference (ISMIR)* (cited on page 1).

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 (2019). "Music Transformer: Generating music with long-term structure." *International Conference on Learning Representations (ICLR)* (cited on pages 8, 14, 23, 24, 32, 40).

Jiun-Long Huang, Shih-Chuan Chiu, and Man-Kwan Shan (2012). “Towards an automatic music arrangement framework using score reduction.” *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 8.1, pp. 1–23 (cited on page 39).

Qingqing Huang, Aren Jansen, Joonseok Lee, Ravi Ganti, Judith Yue Li, and Daniel P. W. Ellis (2022). “MuLan: A Joint Embedding of Music Audio and Natural Language.” *International Society for Music Information Retrieval Conference (ISMIR)* (cited on pages 99, 100).

Qingqing Huang, Daniel S. Park, Tao Wang, Timo I. Denk, Andy Ly, Nanxin Chen, Zhengdong Zhang, Zhishuai Zhang, Jiahui Yu, Christian Frank, Jesse Engel, Quoc V. Le, William Chan, Zhifeng Chen, and Wei Han (2023a). “Noise2Music: Text-conditioned Music Generation with Diffusion Models.” *arXiv preprint arXiv:2302.03917* (cited on pages 99, 100).

Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao (2023b). “Make-An-Audio: Text-To-Audio Generation with Prompt-Enhanced Diffusion Models.” *International Conference on Machine Learning (ICML)* (cited on pages 99, 100, 105).

Yu-Siang Huang and Yi-Hsuan Yang (2020). “Pop Music Transformer: Generating Music with Rhythm and Harmony.” *ACM International Conference on Multimedia (MM)* (cited on pages 9, 14, 23, 25–27, 40, 41).

Hsiao-Tzu Hung, Chung-Yang Wang, Yi-Hsuan Yang, and Hsin-Min Wang (2019). “Improving automatic jazz melody generation by transfer learning techniques.” *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)* (cited on page 51).

David Huron (1997). “Humdrum and Kern: Selective Feature Encoding.” *Beyond MIDI: The Handbook of Musical Codes*. MIT Press. Chap. 27, pp. 375–401 (cited on page 8).

Hymnal (n.d.). URL: https://www.hymnal.net/ (cited on page 13).

Vladimir Iashin and Esa Rahtu (2021). “Taming Visually Guided Sound Generation.” *British Machine Vision Conference (BMVC)* (cited on pages 100, 108).

Improved Diffusion (n.d.). URL: https://github.com/openai/improved-diffusion (cited on page 104).

International Piano-e-Competition (n.d.). URL: https://www.piano-e-competition.com/ (cited on page 58).

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros (2017). “Image-to-Image Translation with Conditional Adversarial Networks.” *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cited on page 62).

Andreas Jansson, Eric Humphrey, Nicola Montecchio, Rachel Bittner, Aparna Kumar, and Tillman Weyde (2017). “Singing Voice Separation with Deep U-Net Convolutional Networks.” *International Society for Music Information Retrieval Conference (ISMIR)* (cited on page 73).

Shulei Ji, Jing Luo, and Xinyu Yang (2020). “A Comprehensive Survey on Deep Music Generation: Multi-level Representations, Algorithms, Evaluations, and Future Directions.” *arXiv preprint arXiv:2011.06801* (cited on pages 23, 25).

Tim Salimans Jonathan Ho (2021). “Classifier-Free Diffusion Guidance.” *NeurIPS Workshop on Deep Generative Models and Downstream Applications* (cited on pages 101, 105).

Ioannis Karydis, Alexandros Nanopoulos, Apostolos Papadopoulos, and Emilios Cambouropoulos (2007). “VISA: The Voice Integration/Segregation Algorithm.” *International Conference on Music Information Retrieval (ISMIR)* (cited on page 39).

Ilya Kavalerov, Scott Wisdom, Hakan Erdogan, Brian Patton, Kevin Wilson, Jonathan Le Roux, and John R. Hershey (2019). “Universal Sound Separation.” *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)* (cited on pages 71, 72).

Kevin Kilgour, Beat Gfeller, Qingqing Huang, Aren Jansen, Scott Wisdom, and Marco Tagliasacchi (2022). “Text-Driven Separation of Arbitrary Sounds.” *Annual Conference of the International Speech Communication Association (INTERSPEECH)* (cited on pages 69, 72).

Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi (2019). “Fréchet Audio Distance: A Metric for Evaluating Music Enhancement Algorithms.” *Annual Conference of the International Speech Communication Association (INTERSPEECH)* (cited on page 105).

Jürgen Kilian and Holger H. Hoos (2002). “Voice Separation — A Local Optimisation Approach.” *International Conference on Music Information Retrieval (ISMIR)* (cited on page 39).

Jong Wook Kim, Rachel Bittner, Aparna Kumar, and Juan Pablo Bello (2019). “Neural Music Synthesis for Flexible Timbre Control.” *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cited on page 54).

Diederik P. Kingma and Jimmy Ba (2015). “Adam: A Method for Stochastic Optimization.” *International Conference on Learning Representations (ICLR)* (cited on pages 18, 44, 58, 86).

Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae (2020a). “HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis.” *Advances in Neural Information Processing Systems (NeurIPS)* (cited on pages 53, 54, 57–59, 66).

Qiuqiang Kong, Yuxuan Wang, Xuchen Song, Yin Cao, Wenwu Wang, and Mark D. Plumbley (2020b). “Source separation with weakly labelled data: An approach to computational

auditory scene analysis." *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cited on pages 71, 72).

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro (2021). "DiffWave: A Versatile Diffusion Model for Audio Synthesis." *International Conference on Learning Representations (ICLR)* (cited on page 101).

Bruno Korbar, Du Tran, and Lorenzo Torresani (2018). "Cooperative learning of audio and video models from self-supervised synchronization." *Advances in Neural Information Processing Systems (NeurIPS)* (cited on page 84).

Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi (2023). "AudioGen: Textually Guided Audio Generation." *International Conference on Learning Representations (ICLR)* (cited on pages 99, 100).

Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brebisson, Yoshua Bengio, and Aaron Courville (2019). "MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis." *Advances in Neural Information Processing Systems (NeurIPS)* (cited on pages 53, 57).

Sanggil Lee, Wei Ping, Boris Ginsburg, Bryan Catanzaro, and Sungroh Yoon (2023). "BigVGAN: A Universal Neural Vocoder with Large-Scale Training." *International Conference on Learning Representations (ICLR)* (cited on pages 101, 102).

Seung Hyun Lee, Wonseok Roh, Wonmin Byeon, Sang Ho Yoon, Chan Young Kim, Jinkyu Kim, and Sangpil Kim (2022). "Sound-Guided Semantic Image Manipulation." *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cited on page 73).

LilyPond (n.d.). URL: https://lilypond.org/ (cited on page 8).

Hao-Min Liu, Hao-Wen Dong, Wen-Yi Hsiao, and Yi-Hsuan Yang (2018a). "Lead sheet and Multi-track Piano-roll generation using MuseGAN." *GPU Technology Conference (GTC) Taiwan* (cited on page 2).

Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D. Plumbley (2023a). "AudioLDM: Text-to-Audio Generation with Latent Diffusion Models." *International Conference on Machine Learning (ICML)* (cited on pages 99–101, 105).

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer (2018b). "Generating Wikipedia by Summarizing Long Sequences." *International Conference on Learning Representations (ICLR)* (cited on page 28).

Xubo Liu, Haohe Liu, Qiuqiang Kong, Xinhao Mei, Jinzheng Zhao, Qiushi Huang, MarkD . Plumbley, and Wenwu Wang (2022). "Separate What You Describe: Language-Queried Audio Source Separation." *Annual Conference of the International Speech Communication Association (INTERSPEECH)* (cited on pages 69, 72).

Xubo Liu, Zhongkai Zhu, Haohe Liu, Yi Yuan, Meng Cui, Qiushi Huang, Jinhua Liang, Yin Cao, Qiuqiang Kong, Mark D. Plumbley, and Wenwu Wang (2023b). "WavJourney: Compositional Audio Creation with Large Language Models." *arXiv preprint arXiv:2307.14335* (cited on page 115).

Magenta (n.d.). URL: https://magenta.tensorflow.org/ (cited on page 9).

Rachel Manzelli, Vijay Thakkar, Ali Siahkamari, and Brian Kulis (2018). "Conditioning Deep Generative Raw Audio Models for Structured Automatic Music." *International Society for Music Information Retrieval Conference (ISMIR)* (cited on pages 53, 54).

Sébastien Marcel and Yann Rodriguez (2010). "Torchvision the Machine-Vision Package of Torch." *ACM International Conference on Multimedia (MM)* (cited on page 12).

Cory Mckay and Ichiro Fujinaga (2006). "JSymbolic: A feature extractor for MIDI files." *International Computer Music Conference (ICMC)* (cited on page 10).

Mido: MIDI Objects for Python (n.d.). URL: https://github.com/mido/mido (cited on page 11).

Yuki Mitsufuji, Giorgio Fabbro, Stefan Uhlich, Fabian-Robert Stöter, Alexandre Défossez, Minseok Kim, Woosung Choi, Chin-Yun Yu, and Kin-Wai Cheuk (2021). "Music Demixing Challenge 2021." *arXiv preprint arXiv:2108.13559* (cited on page 71).

Olof Mogren (2016). "C-RNN-GAN: Continuous recurrent neural networks with adversarial training." *NeurIPS Workshop on Constructive Machine Learning* (cited on pages 9, 14, 32).

Michael Mozer (1994). "Neural Network Music Composition by Prediction: Exploring the Benefits of Psychoacoustic Constraints and Multi-scale Processing." *Connection Science* 6, pp. 247–280 (cited on pages 8, 12).

Aashiq Muhamed, Liang Li, Xingjian Shi, Suri Yaddanapudi, Wayne Chi, Dylan Jackson, Rahul Suresh, Zachary C. Lipton, and Alexander J. Smola (2021). "Symbolic Music Generation with Transformer-GANs." *AAAI Conference on Artificial Intelligence (AAAI)* (cited on pages 23, 40).

MuseScore General SoundFont (n.d.). URL: https://musescore.org/en/handbook/3/soundfonts-and-sfz-files (cited on page 57).

Eita Nakamura and Shigeki Sagayama (2015). "Automatic piano reduction from ensemble scores based on merged-output hidden Markov model." *International Computer Music Conference (ICMC)* (cited on page 39).

Eita Nakamura and Kazuyoshi Yoshii (2018). "Statistical piano reduction controlling performance difficulty." *APSIPA Transactions on Signal and Information Processing* 7 (cited on page 39).

Alex Nichol and Prafulla Dhariwal (2019). “Improved Denoising Diffusion Probabilistic Models.” *International Conference on Machine Learning (ICML)* (cited on pages 99, 101, 104).

Nottingham Database (n.d.). URL: https://ifdo.ca/~seymour/nottingham/nottingham.html (cited on page 13).

Tsubasa Ochiai, Marc Delcroix, Yuma Koizumi, Hiroaki Ito, Keisuke Kinoshita, and Shoko Araki (2020). “Listen to What You Want: Neural Network-based Universal Sound Selector.” *Annual Conference of the International Speech Communication Association (INTERSPEECH)* (cited on pages 71, 72).

Sho Onuma and Masatoshi Hamanaka (2010). “Piano Arrangement System Based On Composers’ Arrangement Processes.” *International Computer Music Conference (ICMC)* (cited on page 39).

Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan (2020). “This Time with Feeling: Learning Expressive Musical Performance.” *Neural Computing and Applications* 32, pp. 955–967 (cited on pages 8, 14, 53).

Andrew Owens and Alexei A Efros (2018). “Audio-Visual Scene Analysis with Self-Supervised Multisensory Features.” *European Conference on Computer Vision (ECCV)* (cited on pages 71, 84).

Andrew Owens, Phillip Isola, Josh McDermott, Antonio Torralba, Edward H. Adelson, and William T. Freeman (2016). “Visually Indicated Sounds.” *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cited on page 100).

Santiago Pascual, Gautam Bhattacharya, Chunghsin Yeh, Jordi Pons, and Joan Serrà (2023). “Full-band General Audio Synthesis with Score-based Diffusion.” *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cited on page 101).

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala (2019). “PyTorch: An Imperative Style, High-Performance Deep Learning Library.” *Advances in Neural Information Processing Systems (NeurIPS)* (cited on pages 12, 87).

Christine Payne (2019). *MuseNet*. OpenAI Blog. URL: https://openai.com/blog/musenet/ (cited on pages 24, 25, 40).

Karol J. Piczak (2015). “ESC: Dataset for Environmental Sound Classification.” *ACM International Conference on Multimedia (MM)* (cited on pages 84, 90).

Ryan Prenger, Rafael Valle, and Bryan Catanzaro (2019). “WaveGlow: A Flow-based Generative Network for Speech Synthesis.” *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cited on pages 53, 57).

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever (2021). "Learning Transferable Visual Models From Natural Language Supervision." *International Conference on Machine Learning (ICML)* (cited on pages 70, 73, 74, 86, 99).

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever (2019). "Language Models are Unsupervised Multitask Learners." *Technical Report of OpenAI* (cited on pages 80, 99).

Colin Raffel (2016). "Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching." PhD thesis. Columbia University. URL: https://colinraffel.com/projects/lmd/ (cited on pages 13, 43).

Colin Raffel and Daniel P. W. Ellis (2014). "Intuitive Analysis, Creation and Manipulation of MIDI Data with pretty_midi." *ISMIR Late-Breaking Demos* (cited on pages 11, 59).

Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel P. W. Ellis (2014). "mir_eval: A Transparent Implementation of Common MIR Metrics." *International Society for Music Information Retrieval Conference (ISMIR)* (cited on page 9).

Torsten Rahne, Martin Böckmann, Hellmut von Specht, and Elyse S Sussman (2007). "Visual cues can modulate integration and segregation of objects in auditory scene analysis." *Brain research* (cited on page 69).

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen (2022). "Hierarchical Text-Conditional Image Generation with CLIP Latents." *arXiv preprint arXiv:2204.06125* (cited on pages 99–101, 110).

Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu (2019). "FastSpeech: Fast, Robust and Controllable Text to Speech." *Advances in Neural Information Processing Systems (NeurIPS)* (cited on pages 55, 58, 67).

Yi Ren, Xu Tan, Tao Qin, Jian Luan, Zhou Zhao, and Tie-Yan Liu (2020). "DeepSinger: Singing Voice Synthesis with Data Mined From the Web." *Conference on Knowledge Discovery and Data Mining (KDD)* (cited on pages 53, 54).

Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck (2018). "A hierarchical latent vector model for learning long-term structure in music." *International Conference on Machine Learning (ICML)* (cited on pages 8, 12).

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer (2022). "High-resolution image synthesis with latent diffusion models." *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cited on page 99).

Olaf Ronneberger, Philipp Fischer, and Thomas Brox (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation." *Proc. MICCAI* (cited on pages 73, 86).

Andrew Rouditchenko, Angie Boggust, David Harwath, Brian Chen, Dhiraj Joshi, Samuel Thomas, Kartik Audhkhasi, Hilde Kuehne, Rameswar Panda, Rogerio Feris, Brian Kingsbury, Michael Picheny, Antonio Torralba, and James Glass (2021). "AVLnet: Learning Audio-Visual Language Representations from Instructional Videos." *Annual Conference of the International Speech Communication Association (INTERSPEECH)* (cited on page 109).

Andrew Rouditchenko, Hang Zhao, Chuang Gan, Josh McDermott, and Antonio Torralba (2019). "Self-Supervised Audio-Visual Co-Segmentation." *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cited on page 71).

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman (2023). "DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation." *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cited on page 115).

Dimitri von Rütte, Luca Biggio, Yannic Kilcher, and Thomas Hofmann (2022). "FIGARO: Generating Symbolic Music with Fine-Grained Artistic Control." *arXiv preprint arXiv:2201.10936* (cited on pages 24–29, 31, 32).

Florin Schimbinschi, Christian Walder, Sarah M. Erfani, and James Bailey (2019). "SynthNet: Learning to Synthesize Music End-to-End." *International Joint Conference on Artificial Intelligence (IJCAI)* (cited on page 54).

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev (2022). "LAION-5B: An open large-scale dataset for training next generation image-text models." *NeurIPS 2022 Datasets and Benchmarks* (cited on page 99).

Mike Schuster and Kuldip Paliwal (1997). "Bidirectional Recurrent Neural Networks." *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681 (cited on page 41).

Wisdom Scott, Aren Jansen, Ron J. Weiss, Hakan Erdogan, and John R. Hershey (2021). "Sparse, efficient, and semantic mixture invariant training: Taming in-the-wild unsupervised sound separation." *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)* (cited on page 71).

Robert Sekuler, Allison B. Sekuler, and Renee Lau (1997). "Sound alters visual motion perception." *Nature* 385 (308) (cited on page 69).

Roy Sheffer and Yossi Adi (2023). "I Hear Your True Colors: Image Guided Audio Generation." *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cited on pages 100, 108).

Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu (2018). "Natural TTS Synthesis by Conditioning WaveNet

on Mel Spectrogram Predictions." *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cited on pages 53, 57).

Chi-Ching Shih, Pei-Ching Li, Yi-Ju Lin, Yu-Lin Wang, Alvin W. Y. Su, Li Su, and Yi-Hsuan Yang (2017). "Analysis and Synthesis of the Violin Playing Style of Heifetz and Oistrakh." *International Conference on Digital Audio Effects (DAFx)* (cited on page 62).

Yi-Jen Shih, Shih-Lun Wu, Frank Zalkow, Meinard Müller, and Yi-Hsuan Yang (2022). "Theme Transformer: Symbolic Music Generation with Theme-Conditioned Transformer." *IEEE Transactions on Multimedia (TMM)* (cited on page 26).

Shinsuke Shimojo and Ladan Shams (2001). "Sensory modalities are not separate modalities: plasticity and interactions." *Current Opinion in Neurobiology* 11 (4) (cited on page 69).

Ian Simon and Sageev Oore (2017). *Performance RNN: Generating Music with Expressive Timing and Dynamics*. Magenta Blog. URL: https://magenta.tensorflow.org/performance-rnn (cited on page 40).

Ian Simon, Adam Roberts, Colin Raffel, Jesse Engel, Curtis Hawthorne, and Douglas Eck (2018). "Learning a Latent Space of Multitrack Measures." *NeurIPS Workshop on Machine Learning for Creativity and Design* (cited on page 25).

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research (JMLR)* 15.56, pp. 1929–1958 (cited on page 44).

Fabian-Robert Stöter, Antoine Liutkus, and Nobutaka Ito (2018). "The 2018 Signal Separation Evaluation Campaign." *International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)* (cited on page 87).

Naoya Takahashi and Yuki Mitsufuji (2021). "Densely connected multidilated convolutional networks for dense prediction tasks." *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cited on page 71).

Hirofumi Takamori, Haruki Sato, Takayuki Nakatsuka, and Shigeo Morishima (2017). "Automatic arranging musical score for piano using important musical elements." *Sound and Music Computing Conference (SMC)* (cited on page 39).

Xu Tan, Tao Qin, Frank Soong, and Tie-Yan Liu (2021). "A Survey on Neural Speech Synthesis." *arXiv preprint arXiv:2106.15561* (cited on page 54).

TensorFlow Datasets (n.d.). URL: https://www.tensorflow.org/datasets (cited on page 12).

John Thickstun, Zaid Harchaoui, and Sham M. Kakade (2017). "Learning Features of Music from Scratch." *International Conference on Learning Representations (ICLR)* (cited on page 43).

Yapeng Tian, Di Hu, and Chenliang Xu (2021). “Cyclic Co-Learning of Sounding Object Visual Grounding and Sound Separation.” *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cited on page 71).

Daniel R Tuohy and Walter D Potter (2005). “A genetic algorithm for the automatic generation of playable guitar tablature.” *International Computer Music Conference (ICMC)* (cited on page 39).

Efthymios Tzinis, Scott Wisdom, Aren Jansen, Shawn Hershey, Tal Remez, Daniel P. W. Ellis, and John R. Hershey (2021). “Into the Wild with AudioScope: Unsupervised Audio-Visual Separation of On-Screen Sounds.” *International Conference on Learning Representations (ICLR)* (cited on pages 71, 72).

Reinier de Valk and Tillman Weyde (2018). “Deep neural networks with voice entry estimation heuristics for voice separation in symbolic music representations.” *International Society for Music Information Retrieval Conference (ISMIR)* (cited on pages 39, 43, 48).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). “Attention is all you need.” *Advances in Neural Information Processing Systems (NeurIPS)* (cited on pages 17, 23, 26, 28, 41, 54, 67).

Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte (2006). “Performance measurement in blind audio source separation.” *IEEE Transactions on Audio, Speech, and Language Processing* 14.4, pp. 1462–1469 (cited on page 78).

Bryan Wang and Yi-Hsuan Yang (2019). “PerformanceNet: Score-to-audio music generation with multi-band convolutional residual network.” *AAAI Conference on Artificial Intelligence (AAAI)* (cited on page 54).

DeLiang Wang and Jitong Chen (2018). “Supervised Speech Separation Based on Deep Learning: An Overview.” *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)* 26.10, pp. 1702–1726 (cited on page 71).

Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous (2017). “Tacotron: Towards End-to-End Speech Synthesis.” *Annual Conference of the International Speech Communication Association (INTERSPEECH)* (cited on page 54).

Ziyu Wang and Gus Xia (2021). “MuseBERT: Pre-training of Music Representation for Music Understanding and Controllable Generation.” *International Society for Music Information Retrieval Conference (ISMIR)* (cited on pages 25, 26, 32).

Wikifonia (n.d.). URL: http://www.wikifonia.org/ (cited on page 13).

Scott Wisdom, Efthymios Tzinis, Hakan Erdogan, Ron J. Weiss, Kevin Wilson, and John R. Hershey (2020). “Unsupervised Sound Separation Using Mixture Invariant Training.”

Advances in Neural Information Processing Systems (NeurIPS) (cited on pages 71, 72, 75, 76).

Ho-Hsiang Wu, Prem Seetharaman, Kundan Kumar, and Juan Pablo Bello (2022a). "Wav2CLIP: Learning Robust Audio Representations From CLIP." *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cited on pages 73, 109).

Shih-Lun Wu and Yi-Hsuan Yang (2020). "The Jazz Transformer on the Front Line: Exploring the Shortcomings of AI-composed Music through Quantitative Measures." *International Society for Music Information Retrieval Conference (ISMIR)* (cited on pages 14, 26, 32).

Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick, and Shlomo Dubnov (2023). "Large-scale Contrastive Language-Audio Pretraining with Feature Fusion and Keyword-to-Caption Augmentation." *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cited on pages 99, 100, 104, 105).

Yusong Wu, Ethan Manilow, Yi Deng, Rigel Swavely, Kyle Kastner, Tim Cooijmans, Aaron Courville, Cheng-Zhi Anna Huang, and Jesse Engel (2022b). "MIDI-DDSP: Detailed Control of Musical Performance via Hierarchical Modeling." *International Conference on Learning Representations (ICLR)* (cited on page 54).

Weihan Xu, Julian McAuley, Shlomo Dubnov, and Hao-Wen Dong (2023). "Equipping Pretrained Unconditional Music Transformers with Instrument and Genre Controls." *IEEE Big Data Workshop on AI Music Generation (AIMG)* (cited on page 2).

Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang (2017). "MidiNet: A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation." *International Society for Music Information Retrieval Conference (ISMIR)* (cited on pages 9, 14).

Li-Chia Yang and Alexander Lerch (2018). "On the evaluation of generative models in music." *Neural Computing and Applications* 32, pp. 4773–4784 (cited on pages 9, 14).

Chih-Hong Yang, Pei-Ching Li, Alvin W. Y. Su, Li Su, and Yi-Hsuan Yang (2016). "Automatic Violin Synthesis Using Expressive Musical Term Features." *International Conference on Digital Audio Effects (DAFx)* (cited on page 62).

Dongchao Yang, Jianwei Yu, Helin Wang, Wen Wang, Chao Weng, Yuexian Zou, and Dong Yu (2022). "Diffsound: Discrete Diffusion Model for Text-to-sound Generation." *arXiv preprint arXiv:2207.09983* (cited on pages 99, 100, 105).

Jinhyeok Yang, Jae-Sung Bae, Taejun Bak, Youngik Kim, and Hoon-Young Cho (2021). "GANSpeech: Adversarial Training for High-Fidelity Multi-Speaker Speech Synthesis." *Annual Conference of the International Speech Communication Association (INTERSPEECH)* (cited on page 62).

Yin-Cheng Yeh, Wen-Yi Hsiao, Satoru Fukayama, Tetsuro Kitahara, Benjamin Genchel, Hao-Min Liu, Hao-Wen Dong, Yian Chen, Terence Leong, and Yi-Hsuan Yang (2021). "Auto-

matic Melody Harmonization with Triad Chords: A Comparative Study." *Journal of New Music Research (JNMR)* 50.1, pp. 37–51 (cited on page 2).

Chengzhu Yu, Heng Lu, Na Hu, Meng Yu, Chao Weng, Kun Xu, Peng Liu, Deyi Tuo, Shiyin Kang, Guangzhi Lei, Dan Su, and Dong Yu (2020). "DurIAN: Duration Informed Attention Network For Multimodal Synthesis." *Annual Conference of the International Speech Communication Association (INTERSPEECH)* (cited on page 55).

Dong Yu, Morten Kolbæk, Zheng-Hua Tan, and Jesper Jensen (2017). "Permutation invariant training of deep models for speaker-independent multi-talker speech separation." *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (cited on pages 71, 76–79, 83, 88, 93–96).

Mingliang Zeng, Xu Tan, Rui Wang, Zeqian Ju, Tao Qin, and Tie-Yan Liu (2021). "MusicBERT: Symbolic Music Understanding with Large-Scale Pre-Training." *Findings of the Association for Computational Linguistics (ACL)* (cited on pages 25, 26).

Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie (2020). "Why Gradient Clipping Accelerates Training: A Theoretical Justification for Adaptivity." *International Conference on Learning Representations (ICLR)* (cited on page 86).

Hang Zhao, Chuang Gan, Wei-Chiu Ma, and Antonio Torralba (2019). "The Sound of Motions." *International Conference on Computer Vision (ICCV)* (cited on page 71).

Hang Zhao, Chuang Gan, Andrew Rouditchenko, Carl Vondrick, Josh McDermott, and Antonio Torralba (2018). "The Sound of Pixels." *European Conference on Computer Vision (ECCV)* (cited on pages 71–73, 77–79, 89, 103).