UNIVERSITY OF CALIFORNIA SAN DIEGO

Representation Learning for Music and Audio Intelligence

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

in

Computer Science

by

Ke Chen

Committee in charge:

Professor Shlomo Dubnov, Chair
Professor Taylor Berg-Kirkpatrick, Co-Chair
Professor Julian McAuley
Professor Miller Puckette

2024

The Dissertation of Ke Chen is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2024

EPIGRAPH

Music is the arithmetic of sounds as optics is the geometry of light.

Claude Debussy

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

As an incoming Ph.D. student back in 2019, I was brimming with enthusiasm for music research, imagining an expansive five-year plan for my project. Over the five-year journey spanning both my personal and academic life, I came to realize that to accomplish such a plan, delving into the research realms of audio, language, and vision are necessary and require more explorations. These diverse experiences not only imbued my music research with novel insights and methodologies, but also inspire my growth to a more professional researcher, as well as an explorer in the vast tapestry of life. While the graduation marks a milestone of my research, it embraces a new chapter of the inception in the project — it is the just the beginning.

I want to express my deep appreciation to Professor Shlomo Dubnov, Professor Taylor Berg-Kirkpatrick, Professor Julian McAuley, and Professor Miller Puckette for their support as members of my committee. Through various paper drafts and revisions, their guidance has proved to be invaluable. Their profound wisdom not only inspires me but also equips me with the knowledge to navigate research bottlenecks, refine paper writings, and deliver impactful presentations. Beyond their roles in academia, they have served as friends in a broader spectrum of my life, with invaluable communications on overcoming challenges in internships, music performance, job finding, and other life experiences. I would also like to thank Professor Gus Xia at Mohamed bin Zayed University of Artificial Intelligence (MBZUAI) and Professor Wei Li at Fudan University. Without them, my research on music and audio would not have started. It is their generous availability and support that helped me get into this field.

I want to express my heartfelt gratitude to my girlfriend, Jingyue Huang, whose support has been instrumental in reaching the final stage of my Ph.D. journey. Together, we explore countless adventures: traveling cities across the United States; sharing our love for music on different artists, singers, and compositions; collaborating on research papers and conference attendance; and cherishing every moment of our lives. Her encouragement and companionship have been invaluable in making me towards the completion of my doctoral studies. I am filled with anticipation for the many more wonderful experiences that await us.

I want to thank to all people during my Ph.D. daily life, including my roommates Zhankui He, Shihan Ran, Zesen Zhang, and Chuai Chuai; research collaborators and friends Hao-Wen Dong (UCSD), Zehao Wang (UCSD), Delong Wang (UCSD), Xingjian Du (University of Rochester), Yusong Wu (University of Montreal), Tianyu Zhang (University of Montreal), Yuchen Hui (University of Montreal), Cheng-i Wang (Audioshake), Gorden Wicherm (MERL), Jonathan Le Roux (MERL), Francois G. Germain (MERL), Beici Liang (NOMONO), Zeyu Jin (Adobe), Jiaqi Su (Adobe), Junyan Jiang (NYU Shanghai), Ziyu Wang (NYU Shanghai), Shuai Yu (Donghua University), Qiuqiang Kong (Chinese Univerity of Hong Kong), Yi Luo (Tencent), Bilei Zhu (Bytedance), Haohe Liu (University of Surrey), Zachary Novack (UCSD), Nikita Srivatsan (Carnegie Mellon University), Zexue He (UCSD), Yuheng Zhi (UCSD), Minghua Liu (UCSD), Xiyuan Zhang (UCSD), Zilong Wang (UCSD), Li Zhong (UCSD), Tianyi Shan (UCSD), Xinyue Wei (UCSD), Xiaoshuai Zhang (UCSD), Zi Lin (UCSD), Lihao Wang (Bytedance), Lintao Ying (Netease), Li Wang (Baidu); and all staffs from UCSD Music Department and Computer Science Engineering Department for support in daily affairs and communication.

Finally, I would like to thank my parents Xiumei Li and Xiaozhou Chen for their support throughout my entire life. They always support my decision, no matter whether there are unknown difficulties ahead.

Chapter 2 contains some materials (texts, tables, and figures) from a published conference paper: Ke Chen, Xingjian Du, Bilei Zhu, Zejun Ma, Taylor Berg-Kirkpatrick, Shlomo Dubnov, *HTS-AT: A Hierarchical Token-Semantic Audio Transformer for Sound Classification and Detection*, in proceedings of International Conference on Acoustics, Speech and Signal Processing, ICASSP 2022. The dissertation author was the first author of this publication.

Chapter 3 contains some materials (texts, tables, and figures) from a published conference paper: Ke Chen, Xingjian Du, Bilei Zhu, Zejun Ma, Taylor Berg-Kirkpatrick, Shlomo Dubnov, *Zero-shot Audio Source Separation through Query-based Learning from weakly labeled Data*, in proceedings of AAAI Conference on Artificial Intelligence Conference, AAAI 2022; and a preprint online paper: Qiuqiang Kong*, Ke Chen*, Haohe Liu, Xingjian Du, Taylor Berg-

Kirkpatrick, Shlomo Dubnov, Mark D Plumbley, *Universal Source Separation with Weakly-Labelled Data*, in the arXiv preprint 2305.07447. The dissertation author was the first author or the co-first-author of these publications.

Chapter 4 contains some materials (texts, tables, and figures) from a published conference paper: Yusong Wu*, Ke Chen*, Tianyu Zhang*, Yuchen Hui*, Taylor Berg-Kirkpatrick, Shlomo Dubnov, *Large-Scale Contrastive Language-Audio Pretraining with Feature Fusion and Keyword-To-Caption Augmentation*, in proceedings of International Conference on Acoustics, Speech and Signal Processing, ICASSP 2023. The dissertation author was the co-first-author of this publication.

Chapter 5 contains some materials (texts, tables, and figures) from a published conference paper: Ke Chen*, Yusong Wu*, Haohe Liu*, Marianna Nezhurina, Taylor Berg-Kirkpatrick, Shlomo Dubnov, *MusicLDM: Enhancing Novelty in Text-To-Music Generation using Beat-Synchronous Mixup Strategies*, in proceedings of International Conference on Acoustics, Speech and Signal Processing, ICASSP 2024. The dissertation author was the first author of this publication.

VITA

2019	Bachelor of Engineering, Fudan University
2021	Master of Arts, University of California San Diego
2024	Doctor of Philosophy, University of California San Diego

PUBLICATIONS

Ke Chen, Jiaqi Su, and Zeyu Jin, "MDX-GAN: Enhancing Perceptual Quality in Multi-Class Source Separation via Adversarial Training", in IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2024.

Ke Chen*, Yusong Wu*, Haohe Liu*, Marianna Nezhurina, Taylor Berg-Kirkpatrick, and Shlomo Dubnov, "MusicLDM: Enhancing Novelty in Text-to-Music Generation Using Beat-Synchronous Mixup Strategies", in IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2024.

Haohe Liu, Ke Chen, Qiao Tian, Wenyu Wang, and Mark D Plumbley, "AudioSR: Versatile Audio Super-resolution at Scale", in IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2024.

Yusong Wu*, Ke Chen*, Tianyu Zhang*, Yuchen Hui*, Taylor Berg-Kirkpatrick, and Shlomo Dubnov, "Large-Scale Contrastive Language-Audio Pretraining with Feature Fusion and Keyword to Caption Augmentation", in IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2023.

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

Ke Chen, Gordon Wicherm, Francois Germain, and Johnathan Le Roux, "Pac-HuBERT: Self-Supervised Music Source Separation via Primitive Auditory Clustering and Hidden-Unit BERT", in IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2023.

Keren Shao*, Ke Chen*, Taylor Berg-Kirkpatrick, and Shlomo Dubnov, "Towards Improving Harmonic Sensitivity and Prediction Stability for Singing Melody Extraction", in International Society for Music Information Retrieval Conference, ISMIR 2023.

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

Ke Chen, Xingjian Du, Bilei Zhu, Zejun Ma, Taylor Berg-Kirkpatrick, and Shlomo Dubnov, "HTS-AT: A Hierarchical Token-Semantic Audio Transformer for Sound Classification and Detection", in IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2022.

Ke Chen, Shuai Yu, Cheng-i Wang, Wei Li, Taylor Berg-Kirkpatrick, and Shlomo Dubnov, "TONet: Tone-Octave Network for Singing Melody Extraction from Polyphonic Music", in IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2022.

Xingjian Du, Ke Chen, Zijie Wang, Bilei Zhu, and Zejun Ma, "ByteCover2: Towards Dimensionality Reduction of Latent Embedding for Efficient Cover Song Identification", in IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2022.

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", in International Society for Music Information Retrieval Conference, ISMIR 2022.

Shlomo Dubnov, Ke Chen, and Kevin Huang, "Deep Musical Information Dynamics: Novel Framework for Reduced Neural-Network Music", in Journal of Creative Music Systems, JCMS 2022.

Ke Chen, Beici Liang, Xiaoshuan Ma, and Minwei Gu, "Learning Audio Embeddings with User Listening Data for Content-based Music Recommendation", in IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021.

Ke Chen, Cheng-i Wang, Taylor Berg-Kirkpatrick, and Shlomo Dubnov, "Music SketchNet: Controllable Music Generation via Factorized Representations of Pitch and Rhythm", in International Society for Music Information Retrieval Conference, ISMIR 2020.

Ziyu Wang*, Ke Chen*, Junyan Jiang, Yiyi Zhang, Maoran Xu, Shuqi Dai, Xianbin Gu, and Gus Xia, "POP909: A Pop-song Dataset for Music Arrangement Generation", in International Society for Music Information Retrieval Conference, ISMIR 2020.

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

ABSTRACT OF THE DISSERTATION

Representation Learning for Music and Audio Intelligence

by

Ke Chen

Doctor of Philosophy in Computer Science

University of California San Diego, 2024

Professor Shlomo Dubnov, Chair
Professor Taylor Berg-Kirkpatrick, Co-Chair

With recent breakthroughs in machine learning, the pursuit of efficient and effective feature representation has gradually taken center stage, igniting groundbreaking possibilities for various downstream applications. While significant progress has been made in the domains of natural language processing and computer vision, there arises an imperative need to construct a robust audio representation model that empowers advanced audio applications.

In this dissertation, we begin from an initial design of an innovative audio transformer as the cornerstone, HTS-AT, that employs imperative designs to capture semantic and acoustic information of audio data. We present a step-by-step demonstration on how we unleash the

power of HTS-AT to unlock a wide range of advanced audio downstream applications in audio understanding and audio generative AI. Specifically, we first adapt HTS-AT to audio event classification, assessing its prowess in comprehending the semantics of audio tracks. Subsequently, we leverage the audio embedding of HTS-AT into audio source separation, evaluating its capability to conceive the acoustic feature of audio. To embrace more applications in conjunction with other modalities, we propose a contrastive language-audio pretraining model (CLAP) that combines HTS-AT with the language understanding model to incorporate the shared information between audio and text representations. From all above explorations, we achieve the target of content creation by proposing MusicLDM, a latent diffusion model that leverages the embeddings of CLAP to perform the text-to-music generation.

Throughout all designs, experiments, and application studies, we achieve successful adaptations and superior performance of different audio downstream tasks rising from a simple audio transformer. Besides, more potential applications in the field of audio content extraction and creation are awaiting, as we will touch upon our ongoing and forthcoming endeavors in addressing their challenges and realizing their full potential.

Chapter 1

Introduction

1.1 Introduction

In the grand symphony of human existence, audio emerges as an effective medium, weaving together the expressions of humanity and the natural world. It contains various forms, each bearing its own purpose and allure. Speech, the messenger of our thoughts and emotions, conveys our workaday dialogue, leisure conversations, and cadence of formal presentations. General sound events and noises envelops us as a constant reminder of the bustling tapestry of life that surrounds us. And music, the arithmetic of sounds with attractive harmonies, lucid structures, and inspiring motivations, drives us to articulate the essence of our humanity.

Beneath such an enchanting medium, the bedrock of science appears as audio reveals its secrets through patterns and principles. From the perspective of signal processing, audio signals are the combinations of different fundamental frequencies and involves the compression and degradation through heir transmissions. From the perspective of artificial intelligence, audio data contains distinctive patterns and statistical distributions that can be predicted and normalized.

Throughout the decades, the pursuit of mankind has illuminated a convergent path, where the realms of audio signal processing and artificial intelligence intersect and intertwine. The advancements in the field of **machine learning** lead to the breakthroughs in different audio tasks, from the rudimentary tasks of audio classification and localization to the lofty ambitions of audio source separation and generation. Indeed, the ascendance of artificial intelligence

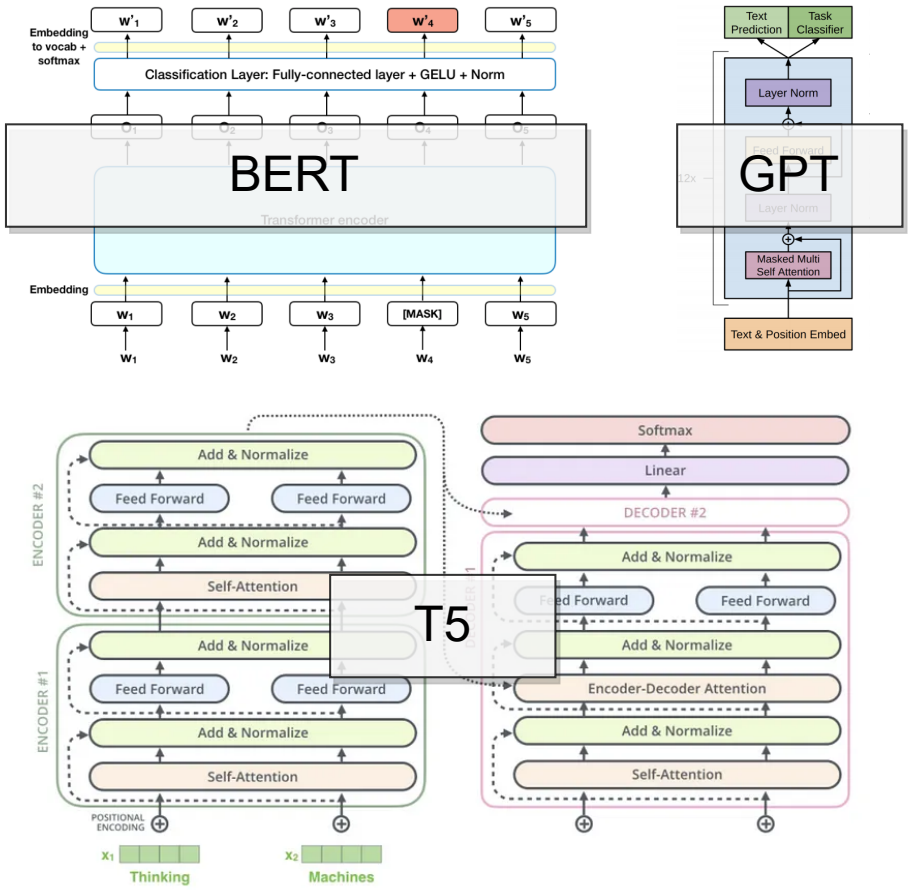


Figure 1.1. The illustration of the representation models in the field of natural language processing, namely BERT [22], GPT [11], and T5 model [86].

plays an imperative role in introducing data-driven techniques into the field of traditional signal processing. As the result, new state-of-the-art performance is achieved and even some previously insurmountable challenges can be addressed.

With recent breakthroughs in artificial intelligence, deep learning has emerged as a pivotal factor, yielding promising results across various research domains, including natural language processing, computer vision, and data mining. Neural networks, including convolutional neural network (CNN) [64], recurrent neural network (RNN [94], LSTM [48], GRU [5]), transformer [113], and mamba [39], have become crucial components within deep learning architectures. While progress in deep-learning-based audio research may appear slower compared to other fields, observed the development of various models achieving superior performance across many

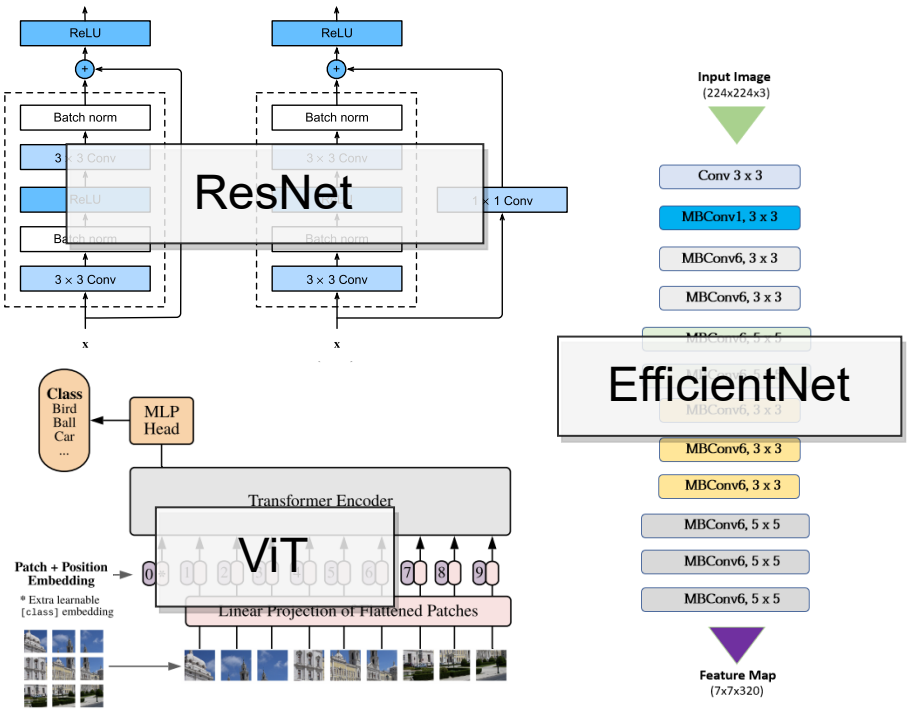


Figure 1.2. The illustration of the representation models in the field of computer vision, namely ResNet [41], EfficientNet [107], and Vision Transformer ViT [24].

audio tasks. These tasks include audio classification [60], music recommendation [15], speech source separation [73], audio generation [36], and others. At this juncture in audio intelligence, a significant challenge emerges — Can we create a unified model capable of addressing diverse downstream applications within the audio domain?

This challenge is also a focal point in representation learning [6] research. The methodology of this research views the training of deep learning models as a process to uncover efficient and effective representations of input data. Fortunately, across various artificial intelligence research domains, researchers have proposed different deep learning models as representation models that consistently excel in diverse downstream tasks.

In natural language processing, as depicted in Figure 1.1, models like BERT [22], GPT [11], and T5 [86] demonstrate exceptional generalization capabilities across tasks such as text summarization, question answering, translation, and text generation. Similarly, in computer vision, as depicted in Figure 1.2, models like ResNet [41], EfficientNet [107], and Vision Trans-

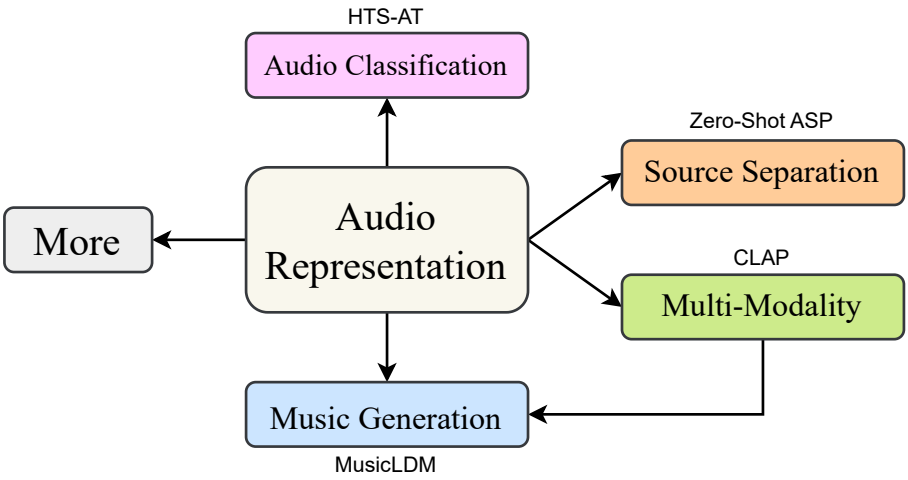


Figure 1.3. The illustration of the proposed audio representation model and its applications throughout the chapters in this dissertation, namely audio classification, source separation, multi-modal learning, and music generation.

former ViT [24] establish effective image representations for tasks including image classification, object segmentation, and image generation. The strength of representation learning lies in its ability to generalize across diverse downstream applications and its role as a foundational model [9] to foster innovation. These benefits further lead to the groundbreaking achievements in both natural language processing and computer vision fields. We have witnessed the rise of large language models (LLMs) [1] and text-to-image generation models [88, 91] as a result.

However, in the field of audio processing, representation learning remains relatively under-explored. While audio models like PANN [60] and PSLA [38] have yielded great performance in audio classification tasks, they are limited by a lack of diverse model designs and connecting modules to validate their effectiveness across other audio tasks. Furthermore, the architectures of these models are somewhat outdated, hindering their potential to achieve more promising results. Therefore, there is an urgent need to design advanced audio representation models and develop pipelines for leveraging these models across various downstream audio tasks. This is crucial for advancing the current stage of audio processing research.

1.2 Novel Contribution

In this dissertation, as depicted in Figure 1.3, we address a gap in audio research by introducing an advanced audio representation model, HTS-AT, built upon the transformer architecture [113]. We further explore the capabilities of this model by applying it to four distinct audio downstream tasks, aiming to validate its efficacy and performance as an representation model:

1. We apply HTS-AT in the audio classification task to evaluate its ability on the semantic information extraction. And we compare it with previous baselines to demonstrate its superior performance.

2. We then apply HTS-AT in the audio source separation task to address an unsolved challenge of universal source separation. We propose a zero-shot audio source separation pipeline (ZS-ASP) by leveraging HTS-AT as the representation model to separate arbitrary sources in the audio mixtures.

3. We further re-design HTS-AT by connecting it with the text encoder module to build a contrastive language-audio pretraining model (CLAP). CLAP leverages the large-scale data training and emerges great generalization ability in bridging audio and text data in the retrieval task and zero-shot audio classification task.

4. his model employs a latent diffusion approach, drawing from CLAP, Variational Auto-encoder (VAE) [57], and the vocoder Hifi-GAN [59]. We further implement techniques to enhance the novelty of generated music and mitigate the risk of plagiarism from training data.

From the above explorations, we demonstrate how HTS-AT could serve as a central audio representation model to capture both content-level details and semantic information of audio

data, Moreover, it exhibits significant generalization capabilities across various audio tasks — a goal we expect to achieve in designing an audio representation model.

1.3 Dissertation Organization

The left chapters of the dissertation are organized as follows:

- In Chapter 2, we introduce the model architecture of HTS-AT as the proposed audio representation model and demonstrate its performance on the audio classification task.
- In Chapter 3, we apply HTS-AT into the zero-shot audio source separation pipeline, and introduce its methodology and experimental results.
- In Chapter 4, we introduce the contrastive language-audio pretraining model (CLAP) based on the HTS-AT architecture.
- In Chapter 5, we introduce MusicLDM , a latent diffusion text-to-music generation model based on CLAP, VAE [57], and Hifi-GAN [59].

Preliminaries and related works pertinent to each specific audio task will be introduced in their respective chapters.

Chapter 2

Hierarchical Token-Semantic Audio Transformer

In this chapter, we begin the exploration of designing an audio representation learning model. While the architectural design of such a model is crucial, an even more fundamental question revolves around its learning target. Throughout decades of research in representation learning across various domains, certain tasks have emerged as initial targets for different types of representation models. In the field of computer vision, image classification serves as the initial task for the basis of image representation models [41, 107, 24]. In the field of natural language processing, BERT-like models [22, 70, 42] gradually lead the language modeling by preserving word embeddings that capture contextual nuances inside sentences. For audio and music processing, our aim is to identify a suitable target that bridges the gap between content-level information and semantic understanding within audio segments. Therefore, we set **the audio classification task** as our primary target, upon which we base the design of audio representation models.

2.1 Introduction

Audio classification is an audio retrieval task which aims to learn a mapping from audio samples to their corresponding labels. Depending on the audio categories, it involves sound event detection [77], music instrument classification [44], among others. It establishes a foundation for

many downstream applications including music recommendation [15], keyword spotting [119], music generation [16, 23], and others.

With burgeoning research in the field of artificial intelligence, we have seen significant promising progress in audio classification. For data collections, many datasets with different types of audio (e.g. AudioSet [35], ESC-50 [84], Speech Command [119], and others) provide platforms for the training and evaluation of models on different subtasks.

For the model design, the audio classification task is thriving based on neural-network-based models. Convolutional neural networks (CNNs) have been widely used in this field, such as DeepResNet [32], TALNet [118], PANN [60], and PSLA [38]. These models leverage CNN to capture features on the audio spectrogram, and further improve their performance through the design of the depth and breadth of the network. Recently, by introducing the transformer structure [113] into audio classification, the audio spectrogram transformer (AST) [37] further achieves the best performance through the self-attention mechanism and the pretrained model from computer vision. In this chapter, we take a further step on a transformer-based audio classification model by first analyzing remaining problems in the AST.

First, since the transformer takes the audio spectrogram as a complete sequential data, AST takes long time to train and consumes large GPU memories. In practice, it takes about one week to train on the full AudioSet dataset with four GPUs of 12 GB memories. One method to boost training speed is to use the ImageNet [20] pretrained model in computer vision. However, this also limits the model to those pretrained hyperparameters, which reduces its scalability in more audio tasks. Indeed, we find that without pretraining, AST can only achieve the baseline performance with the mean average precision of 0.366 on AudioSet, which raises our attention to its learning efficiency on the audio data. Second, AST uses a class-token (CLS) to predict labels, making it unable to predict the start and end time of events in audio samples. Most CNN-based models naturally support the frame-level localization by empirically taking the penultimate layer's output as a event presence map. This inspires us to design a module that makes every output token of an audio transformer aware of the semantic meaning of events (i.e.

a token-semantic module [34]) for supporting more audio tasks (e.g. sound event detection and localization).

In this chapter, we propose an **H**ierarchical **T**oken-**S**emantic **A**udio **T**ransformer (HTS-AT) for audio classification. Our contributions of HTS-AT can be listed as:

- HTS-AT achieves or equals SOTAs on AudioSet and ESC-50, and Speech Command V2 datasets. Moreover, the model without pretraining can still achieve the performance that is only 1%-2% lower than the best results.
- HTS-AT takes fewer parameters (31 Million vs. 87 Million), fewer GPU memories, and less training time (80 hrs vs. 600 hrs) than AST's to achieve the best performance.
- HTS-AT further enables the audio transformer to produce the localization results of event only with weakly labeled data. And it achieves a better performance than the previous CNN-based model.

2.2 Supplementary Materials

The code implementation of HTS-AT and its pretrained weights of different settings are released in https://github.com/RetroCirce/HTS-Audio-Transformer.

2.3 Model Architecture

A typical transformer structure consumes lots of GPU memories and training time, because the length of input tokens is too long and remains unchanged in all transformer blocks from beginning to end. As a result, the machine saves the output and its gradient of each block via large GPU memories, and spends much calculation time maintaining a large global self-attention matrix. To combat these problems, as depicted in Figure 2.1, we propose two key designs: a hierarchical transformer structure and a window attention mechanism.

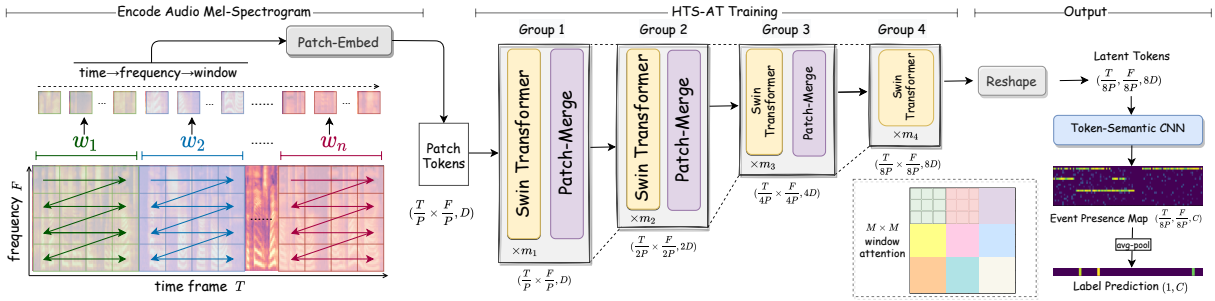


Figure 2.1. The model architecture of HTS-AT. The left part introduces the encoding process of the audio input as the mel-spectrogram. The middle part introduces the training paradigm of the proposed model. The right part introduces the training target and how we use such representations.

2.3.1 Encode the Audio Spectrogram

In the left of Figure 2.1, an audio mel-spectrogram is cut into different patch tokens with a Patch-Embed CNN of kernel size $(P \times P)$ and sent into the transformer in order. Different from images, the width and the height of an audio mel-spectrogram denote different information (i.e. the time and the frequency bin). And the length of time is usually much longer than that of frequency bins. Therefore, to better capture the relationship among frequency bins of the same time frame, we first split the mel-spectrogram into **patch windows** $w_1, w_2, ..., w_n$ and then split the patches inside each window. The order of tokens follows **time→frequency→window** as shown in Figure 2.1. With this order, patches with different frequency bins at the same time frame will be organized adjacently in the input sequence.

2.3.2 Patch-Merge and Window Attention

In the middle of Figure 2.1, the patch tokens are sent into several groups of transformer-encoder blocks. At the end of each group, we implement a Patch-Merge layer [71] to reduce the sequence size. This merge operation is applied by first reshaping the sequence to its original 2D map $(\frac{T}{P} \times \frac{F}{P}, D)$, where D is the latent state dimension. Then it merges adjacent patches as $(\frac{T}{2P} \times \frac{F}{2P}, 4D)$ and finally applies a linear layer to reduce the latent dimension to $(\frac{T}{2P} \times \frac{F}{2P}, 2D)$. As illustrated in Figure 2.1, the shape of the patch tokens is reduced by 8 times

from $(\frac{T}{P} \times \frac{F}{P}, D)$ to $(\frac{T}{8P} \times \frac{F}{8P}, 8D)$ after 4 network groups, thus the GPU memory consumption is reduced exponentially after each group.

For each transformer block inside the group, we adopt a window attention mechanism to reduce the calculation. As shown in different color boxes in the middle right of Figure 2.1, we first split the patch tokens (in 2D format) into non-overlapping $(M \times M)$ **attention windows** $aw_1, aw_2, ..., aw_k$. Then we only compute the attention matrix inside each $M \times M$ attention window. As a result, we have k window attention (WA) matrices instead of a whole global attention (GA) matrix. The computational complexities of these two mechanisms in one transformer block for $f \times t$ audio patch tokens with the initial latent dimension D are:

$$\text{GA:} \quad \mathcal{O}(ftD^2 + (ft)^2 D) \tag{2.1}$$

$$\text{WA:} \quad \mathcal{O}(ftD^2 + M^2 ftD) \tag{2.2}$$

where the window attention reduces the second complexity term by $(\frac{ft}{M^2})$ times. For audio patch tokens in a time-frequency-window order, each window attention module will calculate the relation in a certain range of continuous frequency bins and time frames. As the network goes deeper, the Patch-Merge layer will merge adjacent windows, thus the attention relation is calculated in a larger space. In the code implementation, we use the swin transformer block with a **shifted** window attention [71], a more efficient window attention mechanism. This also helps us to use the swin transformer pretrained vision model in the experiment stage.

2.3.3 Token Semantic Module

The existing AST uses a class-token (CLS) to predict the classification label, which limits it from further indicating the start and end times of events as realized in CNN-based models. In the final layer output, each token contains information about its corresponding time frames and frequency bins. We expect to convert tokens into activation maps for each label-class

(i.e. aware of semantic meaning [34]). For strong-label datasets, we can let the model directly calculate the loss in specific time ranges. For weakly labeled datasets, we can leverage the transformer to locate via its strong capability to capture the relation. In HTS-AT, as shown in the right of Figure 2.1, we modify the output structure by adding a token-semantic CNN layer after the final transformer block. It has a kernel size $(3, \frac{F}{8P})$ and a padding size $(1,0)$ to integrate all frequency bins and map the channel size $8D$ into the event classes C. The output $(\frac{T}{8P}, C)$ is regarded as a event presence map. Finally, we average the featuremap as the final vector $(1,C)$ to compute the binary cross-entropy loss with the groundtruth labels. Apart from the localization functionality, we also expect the token-semantic module to improve the classification performance, as it considers the final output by directly grouping all tokens .

2.4 Experiments

In this section, we evaluate the performance of HTS-AT in four datasets: the event classification on AudioSet [35], ESC-50 [84]; the keyword spotting on Speech Command V2 [119]; and additionally, the event detection on DESED [98].

2.4.1 Event Classification on AudioSet

Dataset and Training Detail

The AudioSet contains over two million 10-sec audio samples labeled with 527 sound event classes. In this chapter, we follow the same training pipeline in [60, 38, 37] by using the full-train set (2M samples) to train our model and evaluating it on the evaluation set (22K samples). All samples are converted to mono as 1 channel by 32kHz sampling rate. We use 1024 window size, 320 hop size, and 64 mel-bins to compute STFTs and mel-spectrograms. As a result, the shape of the mel-spectrogram is $(1024, 64)$ as we pad each 1000-frame (10-sec) sample with 24 zero-frames (T=1024, F=64). The shape of the output featuremap is $(1024, 527)$ (C=527). The patch size is 4×4, the patch window length is 256 frames, and the attention window size is 8×8. Since 8 is divisible by 64, the attention window in the first layer will not

Table 2.1. The mean average precision (mAP) performance on the AudioSet evaluation set across different models. Pretrain: if the model is pretrained on ImageNet. #Params.: the number of model parameters. Ensemble-mAP: the performance achieved by the model ensemble.

Model	Pretrain	#Params.	mAP	Ensemble-mAP
Baseline [35]	✗	2.6M	0.314	-
DeepRes [32]	✗	26M	0.392	-
PANN [60]	✗	81M	0.434	-
PSLA^{P} [38]	✓	13.6M	0.444	0.474
AST [37]	✗	87M	0.366	-
AST^{P} [37]	✓	87M	0.459	0.475 (0.485[1])
HTS-AT^{H}	✗	28.8M	0.440	-
HTS-AT^{HC}	✗	31M	0.453	-
HTS-AT^{HCP}	✓	31M	**0.471**	**0.487**

span two frames with a large time difference. The latent dimension size is D=96 and the final output latent dimension is $8D$=768, which is consistent to AST. Finally, we set 4 network groups with 2, 2, 6, 2 swin-transformer blocks respectively.

We follow [60, 38] to use the balance sampler, $\alpha = 0.5$ mix-up [127], spectrogram masking [82] with time-mask=128 frames and frequency-mask=16 bins, and weight averaging. The HTS-AT is implemented in PyTorch and trained via the AdamW optimizer (β_1=0.9, β_2=0.999, eps=1e-8, decay=0.05) with a batch size of 128 (32×4) in 4 NVIDIA Tesla V-100 GPUs. We apply a warm-up schedule by setting the learning rate as 0.05, 0.1, 0.2 in the first three epochs, then the learning rate is halved every ten epochs until it returns to 0.05. We use the mean average precision (mAP) to evaluate the classification performance.

Experimental Results

In Table 2.1, we compare our HTS-AT with different benchmark models and three self-ablated variations: (1) H: only hierarchical structure; (2) HC: with hierarchical structure and token-semantic module; and (3) HCP: (2) with pretrained vision model (the full setting). Our

[1]AST provides a second bigger ensemble result by using models with different patch settings, which is partially comparable with our settings.

best setting achieves a new SOTA mAP 0.471 in a single model as a large increment from 0.459 by AST. We also ensemble six HTS-ATs with different training random seeds in the same settings to achieve the mAP as 0.487, and outperforms AST's 0.475 and 0.485. We analyze our results in two facets.

Token Semantic Module and Pretraining

PSLA, AST and HTS-AT adopt the ImageNet-pretrained model, where PSLA uses the pretrained EfficientNet [107], AST uses DeiT [109], and our HTS-AT uses the swin-transformer in Swin-T/C24 setting[2] for 256×256 images. We can see that the unpretrained single HTS-AT can achieve an mAP as 0.440. It is improved to 0.453 by the addition of token semantic module, 1.8% lower than 0.471. Finally the pretrained HTS-AT achieves the new best mAP as 0.471. However, the unpretrained single AST only reflects 0.366, 9.3% lower than 0.459. These indicate that: (1) the pretrained model definitely improves the performance by building a solid prior on pattern recognition; and (2) HTS-AT shows a far better scalability to different hyperparameters than AST, since its unpretrained model can still achieve the third best performance.

Parameter Size and Training Time

When comparing the parameter size of each model, the AST has 87M parameters. And HTS-AT is more lightweight with 31M parameters, which is even compatible with CNN-based models. As for the estimated training time, PANN takes about 72 hours to converge and HTS-AT takes about $20 \times 4 = 80$ hours in V-100 GPUs; and AST takes about $150 \times 4 = 600$ hours in 4 TITAN RTX GPUs[3]. The speed improvement corresponds to the less calculation and GPU memory consumption of HTS-AT, as we could feed 128 samples instead of only 12 samples in AST per batch. Therefore, we conclude that HTS-AT consumes less training time and has fewer parameters than AST's.

[2]https://github.com/microsoft/Swin-Transformer

[3]We make memories not exceed 12GB in V-100 in line with TITAN RTX.

Table 2.2. The accuracy performance of audio classification on the ESC-50 dataset and the Speech Command V2 dataset.

Model	ESC-50 Acc.(%)	Model	SCV2 Acc.(%)
PANN [60]	90.5	RES-15 [116]	97.0
AST [37]	95.6 ± 0.4	AST [37]	$\mathbf{98.1 \pm 0.05}$
ERANN [115]	96.1	KWT-2 [7]	97.3 ± 0.03
HTS-AT	$\mathbf{97.0 \pm 0.2}$	HTS-AT	$\mathbf{98.0 \pm 0.03}$

2.4.2 Evaluations on ESC-50 and Speech Command V2

Dataset and Training Detail

The ESC-50 dataset contains 2000 5-sec audio samples labeled with 50 environmental sound classes in 5 folds. We train the model for 5 times by selecting 4-fold (1600 samples) as training set and the left 1-fold (400 samples) as test set. And we repeat this experiment 3 times with different random seeds to get the mean performance and deviation. The Speech Command V2 contains 105,829 1-sec spoken word clips labled with 35 common word classes. It contains 84843, 9981, and 11005 clips for training, validation and evaluation. Similarly, we train our HTS-AT for 3 times to obtain the prediction results. We use the mean accuracy score (acc) for the evaluation on both datasets. For the data processing, we resample the ESC-50 samples into 32kHz and the Speech Command clips 16kHz. And we follow the same setting.

Experimental Results

We use our best AudioSet-pretrained HTS-AT to train on these two dataset respectively and compare it with benchmark models (also in AudioSet or extra data pretraining). Since 1-sec and 5-sec does not take the full 10-sec input trained on AudioSet, we repeat the 1-sec and 5-sec by 10 and 2 times to make it 10-sec. As shown in Table 2.2, the results shows that our HTS-AT achieves a new SOTA as 97.0% on ESC-50 dataset and equals the SOTA 98.0% on Speech Command V2. Our deviations are relatively smaller than AST's, indicating that HTS-AT is more stable after convergence.

Table 2.3. The event-based F1-scores of each class on the DESED test set. Models with * are from DCASE 2021 [2], which are partial references since they use extra training data and are evaluated on DESED test set and its another private subset.

Model	Alarm	Blender	Cat	Dishes	Dog	Shaver	Frying	Water	Speech	Cleaner	Average
PANN [60]	34.3	42.4	36.3	17.6	35.8	23.8	9.3	30.6	69.7	51.0	35.1
HTS-AT	**48.6**	52.9	67.7	25.0	48.0	42.9	60.3	43.0	46.8	49.1	48.4
HTS-AT - Ensemble	47.5	55.1	**72.4**	30.9	**49.7**	41.9	**63.2**	**44.3**	51.3	50.6	50.7
Zheng et al.* [**?**]	41.4	54.1	**72.4**	29.4	47.8	**61.01**	49.2	33.7	**69.5**	65.5	**52.4**
Kim et al.* [**?**]	34.7	**59.8**	71.6	40.4	47.3	26.2	61.8	32.8	64.9	**66.7**	50.6
Lu et al.* [**?**]	37.1	41.4	62.5	**40.6**	39.7	46.5	46.5	34.5	54.5	46.9	45.0

2.4.3 Localization Performance on DESED

We additionally evaluate HTS-AT's capability to localize the sound event as start and end time in given audio samples. We use the DESED test set [98], which contains 692 10-sec test audio samples in 10 classes with the strong labels. We mainly compare our HTS-AT with PANN. We do not include AST and PSLA since AST does not directly support the event localization and the PSLA's code is not published. We also compare it partially with models in DCASE 2021 [**?**], nevertheless they use extra training data and are evaluated on DESED test set and its another private subset. We use the event-based F1-score on each class as the evaluation metric, implemented by a Python library `psds_eval`[4].

The F1-scores on all 10 classes in the DESED by different models are shown in Table 2.3. We find that HTS-AT achieves better F1-scores on 8 classes and a better average F1-score 50.7% than PANN. When compared among leaderboard models, our model still achieves some highest scores of certain classes. However, the F1-scores on Speech and Cleaner are relatively low, indicating that there are still some improvements for a better localization performance. From the above experiments, we can conclude that HTS-AT is able to produce the specific localization output via the token-semantic module, which extends the functionality of the audio transformer.

[4]https://github.com/audioanalytic/psds_eval

2.5 Conclusion

In this chapter, we propose HTS-AT, a hierarchical token-semantic transformer for audio classification, as the design of the audio presentation learning model. It achieves a new SOTA on multiple datasets of different audio classification scenarios, demonstrating its strong capability to bridge between content-level audio information and semantic audio information. Furthermore, the token-semantic module enables HTS-AT to locate the events start and end time. Experiments show that HTS-AT is a high performance, high scalability, and lightweight audio transformer. In the next chapter, we will continue to leverage HTS-AT, as the core network of our audio representation model, into an more advanced application, audio source separation.

This chapter contains some materials (texts, tables, and figures) from a published conference paper: Ke Chen, Xingjian Du, Bilei Zhu, Zejun Ma, Taylor Berg-Kirkpatrick, Shlomo Dubnov, *HTS-AT: A Hierarchical Token-Semantic Audio Transformer for Sound Classification and Detection*, in proceedings of International Conference on Acoustics, Speech and Signal Processing, ICASSP 2022. The dissertation author was the first author of this publication.

Chapter 3

Zero-Shot Audio Source Separation

From the Chapter 2, HTS-AT serves as an effective audio representation learning model to achieve superior performance on audio classification. However, the effectiveness of the audio representation model will not solely work on a single task. It is essential to recognize that the effectiveness of an audio representation model extends beyond a single task. To comprehensively assess HTS-AT, we aim to extend its applicability to various tasks in audio and music signal processing.

Upon deeper analysis, the intrinsic goal of audio classification emerges as a semantic extraction task from audio signals. However, the successful performance on audio classification does not guarantee that the representation of HTS-AT effectively conceives audio content information from the original inputs. Aggressive compression or abstraction of audio signals by the model can partially lead to the high performance on audio classification [60], which inadequately capture essential audio content information. This could potentially limit the utility of such representation models in downstream applications that require such content.

In order to further evaluate HTS-AT, we introduce it into the field of audio source separation, as **a content extraction task**. This exploration aims to further determine whether HTS-AT can effectively address challenges in audio source separation tasks and ascertain its efficacy as an audio representation model capable of preserving both semantic and content-level information from audio signals.

3.1 Introduction

Audio source separation is a core task in the field of audio processing using artificial intelligence. The goal is to separate one or more individual constituent sources from a single recording of a mixed audio piece. Audio source separation can be applied in various downstream tasks such as audio extraction, audio transcription, and music and speech enhancement. Although there are many successful backbone architectures, such as Wave-U-Net, TasNet, and D3Net [103, 73, 106]), fundamental challenges and questions remain — how can the models be made to better generalize to multiple, or even unseen, types of audio sources when supervised training data is limited?

The challenge is known as **universal source separation**, meaning that we only need a single model to separate as many sources as possible. Most models mentioned above require training a full set of model parameters for each target type of audio source. As a result, training these models is both time and memory intensive. There are several heuristic frameworks [**?**] that leverage meta-learning to bypass this problem, but they have difficulty generalizing to diverse types of audio sources. In other words, these frameworks succeeded in combining several source separators into one model, but the number of sources is still limited.

One potential approach to overcome this challenge is to train a model with an audio separation dataset that contains a very large variety of sound sources. The more sound sources a model can see, the better it will generalize. However, the scarcity of the supervised separation datasets makes this process challenging. Most separation datasets contain only a few source types. For example, MUSDB18 [87] and DSD100 [72] contain music tracks of only four source types (vocal, drum, bass, and other) with a total duration of 5-10 hours. MedleyDB [8] contains 82 instrument classes but with a total duration of only 3 hours. There exists some large-scale datasets such as AudioSet [35] and FUSS [122], but they contain only weakly labeled data. AudioSet, for example, contains 2.1 million 10-sec audio samples with 527 sound events. However, only 5% of recordings in Audioset have a localized event label [45]. For the remaining 95% of recordings,

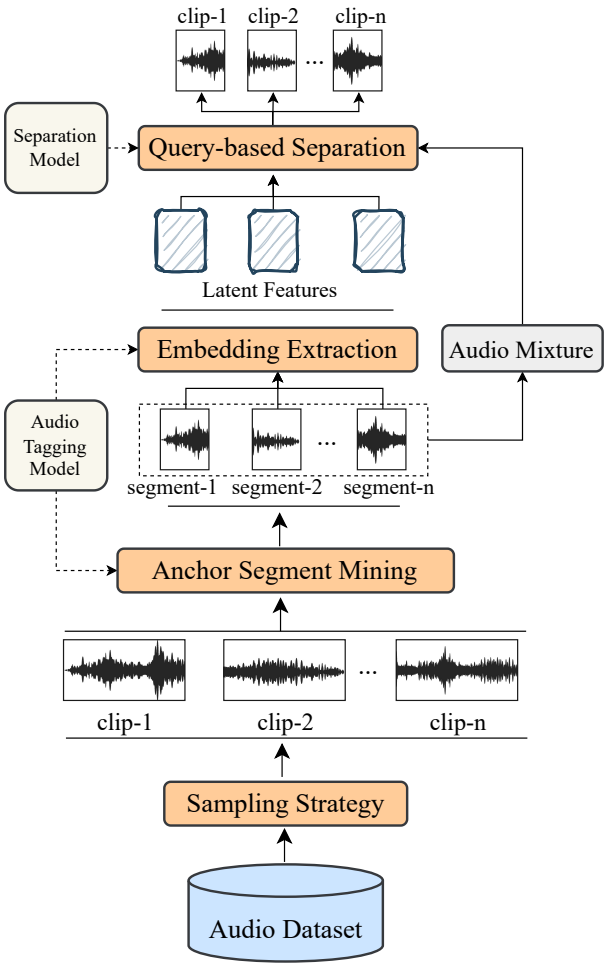


Figure 3.1. The architecture of zero-shot audio source separation trained from weakly-labelled data, including datasets, sampling strategies, anchor segment mining, embedding extraction, and query-based audio source separation module.

the correct occurrence of each labeled sound event can be anywhere within the 10-sec sample. In order to leverage this large and diverse source of weakly labeled data, we first need to localize the sound event in each audio sample, which is referred as an audio tagging task [29].

In this chapter, as illustrated in Figure 3.1, we devise a pipeline that comprises of three components:

1. A data simulation process using a large-scale weakly labeled audio dataset and a sampling strategy to create training data of source separation in multiple source targets.

2. An audio representation learning model, proposed as HTS-AT in Chapter 2, for performing

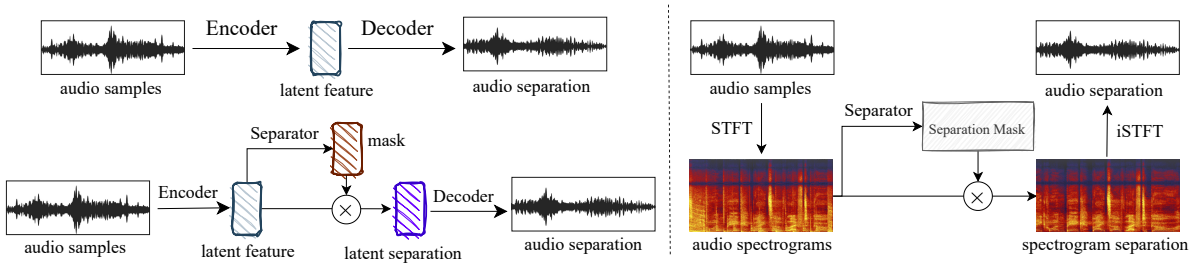


Figure 3.2. The standard architecture of deep-learning-based audio source separation models. Left top: synthesis-based separation model. Left bottom: mask-based separation model. Right: the general type of frequency-domain separation model.

audio classification and localization tasks to contribute to the final source separation target, during the anchor segment mining and embedding extraction steps.

3. A query-based U-Net source separator, trained by the mixture data from the proposed sampling strategy and the separation condition from HTS-AT.

The HTS-AT can localize the occurrences of sound events from weakly labeled audio samples and encode them as latent source embeddings. The separator learns to separate out a target source from an audio mixture given a corresponding target source embedding query, which is produced by the embedding processor. Further, the embedding processor enables zero-shot generalization by forming queries for new audio source types that were unseen at training time. In the experiment, we find that our model can separate unseen types of audio sources, including musical instruments and held-out AudioSet's sound classes, effectively by achieving the SDR performance on par with existing state-of-the-art (SOTA) models.

3.2 Supplementary Materials

The code implementation of zero-shot audio source separation and its pretrained weights of different settings are released in https://github.com/RetroCirce/Zero_Shot_Audio_Source_Separation.

3.3 Audio Source Separation via Neural Networks

Deep learning methods for audio source separation have outperformed traditional methods such as Non-negative Matrix Factorization [65]. Figure 3.2 introduces the source separation models in the time domain (left) and in the frequency domain (right).

3.3.1 Time-domain Separation Models

A neural network time-domain separation model f is typically constructed as an encoder-decoder architecture, as shown in the left of Figure 3.2. Formally, given a single-channel audio clip $x \in \mathbb{R}^L$ and a separation target $s \in \mathbb{R}^L$, where L is sample length, the separator f contains two types: a synthesis-based separation system that directly outputs the waveform of the target source, and a mask-based separation that predict a mask that can be multiplied to the mixture to output the target source.

Separation models such as Demucs [93] and Wave-U-Net [103], f directly estimates the final separation target: $\hat{s} = f(x)$. Mask-based separation models such as TasNet [73] and ConvTasNet [74] predict masks in the latent space produced by the neural network. The masks control how much of sources should remain from the mixture. Then, a decoder is designed to reconstruct the separated waveform from the masked latent feature produced by the neural network.

3.3.2 Frequency-domain Separation Models

In contrast to time-domain models, frequency-domain models leverage a spectrogram, such as a short-time Fourier transform (STFT), to facilitate the separation process. Harmonic features have more patterns in the frequency domain than those in the time domain. This might help improve separation performance in source separation tasks, such as music source separation and environmental sound separation [75].

Formally, given a mono audio clip x, we denote the STFT of x as a complex matrix

$X \in \mathbb{C}^{T \times F}$, where T is the number of time frames and F is the number of frequency bins. We denote the magnitude and the phase of X as $|X|$ and $\angle X$, respectively. The right part of Figure 3.2 shows a frequency-domain separation system f predicting a magnitude ideal ratio mask (IRM) [80] $M \in \mathbb{R}^{T \times F}$ or a complex IRM (cIRM) [121] $M \in \mathbb{C}^{T \times F}$ that can be multiplied by the STFT of the mixture to obtain the STFT of the separated source. The complex STFT of the separated source $\hat{S} \in \mathbb{C}^{T \times F}$ can be calculated by:

$$\hat{S} = M \odot X. \tag{3.1}$$

where $\odot$ is the element-wise complex multiplication. Then, the separated source $\hat{s} \in \mathbb{R}^{L}$ can be obtained by applying an inverse STFT on $\hat{S}$.

Frequency domain models include fully connected neural networks [125], recurrent neural networks (RNNs) [49, 105, 112], and convolutional neural networks (CNNs) [13]. UNets [53, 43] are variants of CNN that contain encoder and decoder layers for source separation. Band-split RNNs (BSRNNs) [75] apply RNNs along both the time and frequency axes to capture time and frequency domain dependencies. There are also approaches such as hybrid Demucs [93] which combine time and frequency domain systems to build source separation systems.

3.3.3 Datasets of Source Separation

Many previous source separation systems require clean source data to train source separation systems. However, the collection of clean source data is difficult and time-consuming. Table 3.1 summarizes datasets that can be used for source separation. We can observe that previous clean source datasets have duration of around tens of hours, while weakly labeled datasets (detail in section 3.5.1) are extremely larger than clean source datasets. Among the large-scale datasets, AudioSet [**?**] is a representative weakly labeled dataset containing over 5,800 hours of 10-second audio clips, with an ontology of 527 sound classes. The ontology of AudioSet has a tree structure, where each audio clip may contain multiple tags. Such data

Table 3.1. The illustration of source separation datasets.

Dataset	Duration (hours)	Classes	Type
Voicebank-Demand [114]	19	1	Clean
MUSDB18 [87]	6	4	Clean
UrbanSound8K [95]	10	10	Clean
FSDKaggle 2018 [30]	20	41	Clean
FUSS [122]	23	357	Clean
AudioSet [35]	5,800	527	weakly labeled

scale inspires us to use AudioSet, and explore how to deal with the weakly labeled AudioSet in addressing challenges of universal source separation.

3.3.4 Universal Source Separation

Universal source separation attempts to employ a single model to separate different types of sources. Currently, the meta-learning model MetaTasNet [96] can separate up to four sources in MUSDB18 dataset in the music source separation task. SuDoRM-RF [111], the Uni-ConvTasNet [54], the PANN-based separator [61], and MSI-DIS [67] extend the universal source separation to speech separation, environmental source separation, speech enhancement and music separation and synthesis tasks. However, most existing models require a separation dataset with clean sources and mixtures to train, and only support a limited number of sources that are seen in the training set. An ideal universal source separator should separate as many sources as possible even if they are unseen or not clearly defined in the training. In this chapter, by continuing from HTS-AT, we move further by proposing a pipeline that can use audio event samples for training a separator that generalizes to diverse and unseen sources.

3.4 Audio Classification and Localization

As introduced in Chapter 2, the audio classification task is to classify one or more target sound events in given audio signals. The audio localization task further requires the model to output the specific time-range of events on the audio timeline. Currently, the convolutional

neural network (CNN) is being widely used to detect sound events. The Pretrained Audio Neural Networks (PANN) [60] and the PSLA [38] achieve the current CNN-based SOTA for the sound event detection, with their output featuremaps serving as an empirical probability map of events within the audio timeline. For the transformer-based structure, the audio spectrogram transformer (AST) [37] re-purposes the visual transformer structure ViT [24] and DeiT [109] to use the transformer class-token to predict the sound event. It achieves the best performance on the sound event detection task in AudioSet. However, it cannot directly localize the events because it outputs only a class-token instead of a featuremap. In this chapter, we leverage HTS-AT to detect and localize the sound event. Moreover, we use HTS-AT to process the weakly labeled data that is sent downstream into the following separator.

3.5 Model Architecture and Pipeline

3.5.1 Weakly Labeled Data

In contrast to clean source data, weakly labeled data only contain the labels of what sound classes are present in an audio recording. Weakly labeled data may also contain interfering sounds. There are no time stamps for sound classes or clean sources. We denote the n-th audio clip in a weakly labeled dataset as a_n where a is the abbreviation for the *audio*. The tags of a_n is denoted as $y_n \in \{0,1\}^K$, where K is the number of sound classes. The value $y_n(k) = 1$ indicates the presence of a sound class k while $y_n(k) = 0$ indicates the absence of a sound class k. We denote a weakly labeled dataset as $D = \{a_n, y_n\}_{n=1}^N$, where N is the number of training samples in the dataset. The weakly labeled audio recording also contains unknown interference sounds, i.e., $y_n(k) = 0$ may contain missing tags for some sound class k.

The goal of a weakly labeled USS system is to separate arbitrary sounds trained with only weakly labeled data. Recapping Figure 3.1, we depict the architecture of our proposed zero-shot audio source separation system, containing four steps:

1. We apply a sampling strategy to sample audio clips of different sound classes from a

weakly labeled audio dataset.

2. We define an anchor segment as a short segment that is most likely to contain a target sound class in a long audio clip. We apply an anchor segment mining algorithm to localize the occurrence of events/tags in the weakly labeled audio tracks.

3. We use pretrained audio classification model (i.e., HTS-AT) to predict the tag probabilities or embeddings of anchor segments.

4. We mix anchor segments as input mixtures, train a query-based separation network to separate the mixture into one of the target source queried by the sound class condition.

3.5.2 Audio Clips Sampling

The original AudioSet is highly unbalanced — sound classes such as "Speech" and "Music" have almost 1 million audio clips, while sound classes such as "tooth breath" have only tens of training samples. Without balanced sampling, the neural network may never see infrequent sound classes in a relatively short training period, resulting an under-optimization of the source separation in certain sources.

Following the training scheme of audio classification systems [60, 37], we apply a balanced sampling strategy to construct training data of source separation, which samples audio clips from different sound classes to constitute a mini-batch to ensure the clips contain balanced sound classes (i.e., each sound class is sampled evenly from the unbalanced dataset). We denote a mini-batch of sampled audio clips as $\{a_i\}_{i=1}^{B}$, where B is the mini-batch size.

3.5.3 Anchor Segment Mining

Anchor segment mining is the core part of Universal or Zero-shot audio source separation systems trained on weakly labeled data. Since the weakly labeled audio tracks do not always contain the labeled sound class throughout its timeline, we need to extract a short audio segment inside this track to create source data for training the separation model.

We define anchor segment mining, as a procedure of sound event localization, to localize anchor segments in an audio clip. This can help us extract audio clips with relatively clean sound sources from weakly labeled audio samples.

Formally, given an audio clip $a_i \in \mathbb{R}^L$, an anchor segment mining algorithm extracts an anchor segment $s_i \in \mathbb{R}^{L'}$ from a_i, where $L' < L$ is the samples number of the anchor segment. For each audio clip in mini-batch $\{a_i\}_{i=1}^B$, we apply a pretrained audio localization model to detect an anchor segment s_i, where the center of s_i is the time stamp where the sound class label is most likely to occur in terms of probability.

Specially, an audio localization model outputs two prediction results:

1. The event classification prediction $p_e \in [0,1]^K$, where K is the number of sound classes.

2. The framewise event prediction $p_{\text{frame}} \in [0,1]^{T \times K}$, where T the number of frames.

Typically, the event prediction p_e is usually calculated by summarizing the averaged value and the maximum value along the framewise event prediction p_{frame} (detail in Chapter 2).

During the anchor segment mining step, we use the audio localization model to perform framewise event prediction of an audio clip. Then, for the target sound class k of this audio clip, we denote the anchor segment score of the sound class k as:

$$q_k(t) = \sum_{t-\tau/2}^{t+\tau/2} p_{\text{frame}}(t,k), \tag{3.2}$$

where τ is the duration of anchor segments. Then, the center time t of the optimal anchor segment is obtained by:

$$t_{\text{anchor}} = \underset{t}{\text{argmax}}\ q_k(t). \tag{3.3}$$

We apply the anchor segment mining strategy as described in (3.2) and (3.3) process a mini-batch of the audio clips $\{a_1, ..., a_B\}$ into a mini-batch of anchor segments $\{s_1, ..., s_B\}$.

Algorithm 1. The data sampling and anchor segment mining steps for data preparation.

1: **Inputs:** dataset $D = \{a_n, y_n\}_{n=1}^{N}$ containing K sound classes, mini batch size B.
2: **Outputs:** a mini-batch of mixture and anchor segment pairs $\{(s, s_i)\}_{i=1}^{B}$.
3: **Step 1: Balanced Sampling**: Uniformly sample B sound classes from sound classes $\{1, ..., K\}$ without replacement. Sample one audio clip $a_b, b = 1, ..., B$ for each selected sound class. We denote the mini-batch of audio clips as $\{a_1, ..., a_B\}$.
4: **Step 2: Anchor Segment Detection**: Apply a pretrained audio localization model on each a_i to detect the optimal anchor time stamp t_i by (3.2)(3.3). Extract the anchor segment $s_i \in \mathbb{R}^{L'}$ whose center is t_i.
5: **Step 3 Segment Re-verification**: Use the same localization model to predict the event classification prediction of s_i and apply threshold $\theta \in [0,1]^K$ to get binary results $r_i \in \{0,1\}^K$, where where $r_i(k)$ is set to 1 if the presence probability is larger than $\theta(k)$. Permute $\{s_i\}_{i=1}^{B}$ so that $r_i \wedge r_{(i+1)\%B} = \mathbf{0}$, where $\%$ is the modulo operator.
6: **Step 4: Mixing**: Create mixture source pairs $\{(s, s_i)\}_{i=1}^{B}$.

Together, Algorithm 1 illustrates the procedure for creating training data. Step 1 describes audio clip sampling and Step 2 describes anchor segment mining. To further avoid two anchor segments containing the same classes being mixed, we propose an Step 3 to further filter the mined anchor segments from a mini-batch of audio clips $\{s_1, ..., s_B\}$ to constitute mixtures. Step 4 describes mixing detected anchor segments into mixtures to train the separation system.

3.5.4 Audio Models in Use: PANN and HTS-AT

Audio classification models, including HTS-AT as our proposed audio representation model, are capable of localizing the time stamps of audio events, even though they are only trained on weakly labeled datasets such as AudioSet [60, 38, 37, 118]. To verify if the proposed HTS-AT serves as an advanced audio representation model to previous classification baselines. We apply both PANN [60] and HTS-AT to perform the anchor segment mining procedure.

Figure 3.3 shows the model architectures of both PANN and HTS-AT. The event presence map in both PANN and HTS-AT outputs denote the the framewise prediction $p_{\text{frame}} \in [0,1]^{T \times K}$. Additionally, the output of the penultimate layer with a size of (T, H) can be used to obtain its averaged vector with a size of H as a latent source embedding for conditional source separation as our last step, where H is the dimension of the latent embedding.

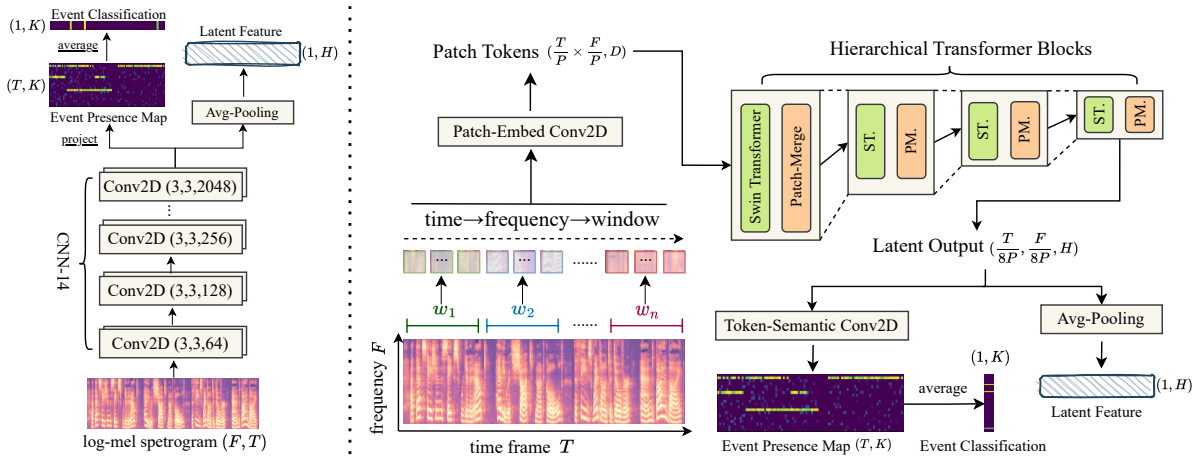


Figure 3.3. Two employed models for audio classification and localization. Left: Pretrained Audio Neural Networks (PANN) in CNN14 architecture. Right: Hierarchical Token-Semantic Transformer (HTS-AT) in 4-block architecture as proposed in Chapter 2.

The detailed architecture of HTS-AT has been introduced in Chapter 2. It applies Swin-Transformer [71] blocks to process the patch-level audio inputs as the sequence. A token-semantic 2D-CNN [34] is implemented at the end to further process the reshaped output $(\frac{T}{8P}, \frac{F}{8P}, H)$ into the framewise event presence map (T, K) ,which can be averaged to an event classification vector K. The latent embedding, at the same time, is produced by averaging the reshaped output into a H-dimension vector with an average-pooling layer.

For the Pretrained Audio Neural Networks (PANN), as shown in the left of Figure 3.3, it contains CNN blocks constructed by the VGG architecture [99] to convert an audio mel-spectrogram into a (T, K) featuremap, where T is the number of time frames and K is the number of sound event classes. The model averages the featuremap over the time axis to obtain a final probability vector $(1, K)$ and computes the binary cross-entropy loss between it and the groudtruth label. Since CNNs can capture the information in each time window, the featuremap (T, K) is empirically regarded as a presence probability map of each sound event at each time frame. When determining the latent source embedding for the following pipeline, the penultimate layer's output (T, H) can be used to obtain its averaged vector $(1, H)$ as the latent source embedding.

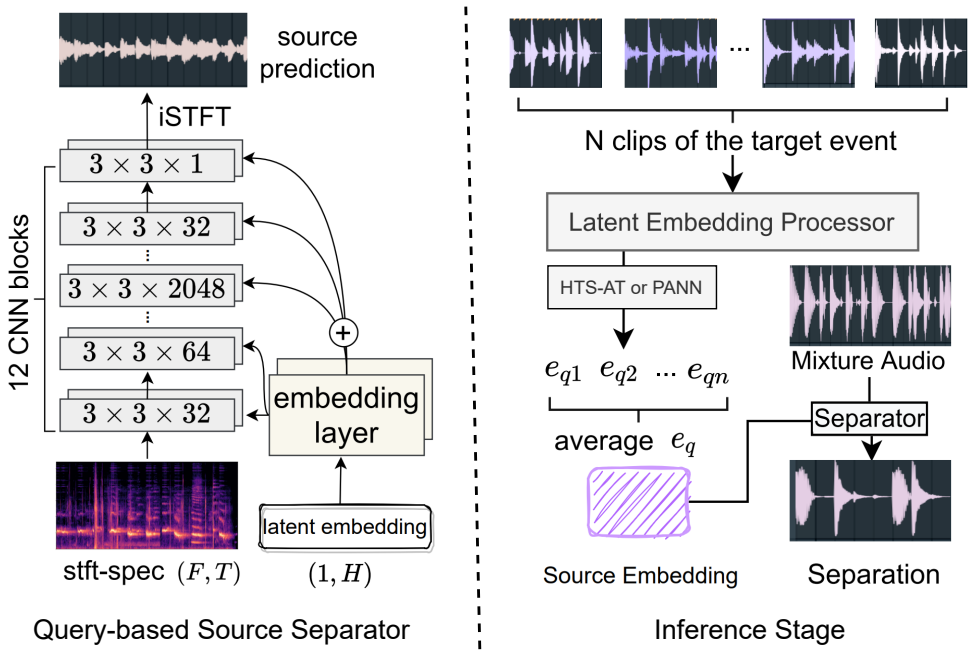


Figure 3.4. The model architecture of query-based source separator (Left) and the paradigm of zero-shot audio source separation in the inference stage (Right).

3.5.5 Query-based Source Separator

By the utilized audio localization models (PANN or HTS-AT), we can localize the most possible occurrence of a given sound event in audio samples. Finally, we could get two 2-sec audio segments s_1, s_2 as the most possible occurrences of the sound classes k_1, k_2 by the anchor segment mining on two original audio clips a_1, a_2.

Subsequently, we resend two segments s_1, s_2 into the SED system to obtain two source embeddings e_1, e_2. Each latent source embedding $(1, H)$ is incorporated into the source separation model to specify which source needs to be separated.

Specifically, we collect s_1, s_2, e_1, e_2, we mix two clips with energy normalization. We first calculate the energy of a signal s_i by $E = ||s_i||_2^2$. We denote the energy of s_i and s_{i+1} as E_i and E_{i+1}. We apply a scaling factor $\alpha_i = \sqrt{E_i / E_{i+1}}$ to s_{i+1} when creating the mixture s:

$$s = s_i + \alpha s_{i+1}. \tag{3.4}$$

By this means, both anchor segments s_1 and s_2 have the same energy which is beneficial to the optimization of the separation system. On the one hand, we match the energy of anchor segments to let the neural network learn to separate the sound classes. On the other hand, the amplitude diversity of sound classes is increased.

Then we send two training triplets $(s, s_1, e_1), (s, s_2, e_2)$ into the separator f, respectively. We let the separator to learn the following regression:

$$f(s, e_j) \mapsto s_j, j \in \{1, 2\}. \tag{3.5}$$

As shown in the left part of Figure 3.4, we base on U-Net [92] to construct our source separator, which contains a stack of downsampling and upsampling CNNs. The mixture clip s is converted into the spectrogram by Short-time Fourier Transform (STFT). In each CNN block, the latent source embedding e_j is incorporated by two embedding layers producing two featuremaps and added into the audio featuremaps before passing through the next block. Therefore, the network will learn the relationship between the source embedding and the mixture, and adjust its weights to adapt to the separation of different sources. The output spectrogram of the final CNN block is converted into the separate waveform $s\prime$ by inverse STFT (iSTFT). Suppose that we have n training triplets $\{(s^1, s_j^1, e_j^1), (s^2, s_j^2, e_j^2), ..., (s^n, s_j^n, e_j^n)\}$, we apply the Mean Absolute Error (MAE) to compute the loss between separate waveforms $S' = \{s^{1\prime}, s^{2\prime}, ..., s^{n\prime}\}$ and the target source clips $S_j = \{s_j^1, s_j^2, ..., s_j^n\}$:

$$L = MAE(S_j, S') = \frac{1}{n} \sum_{i=0}^{n} |s_j^i - s^{i\prime}| \tag{3.6}$$

Combining these two components together, we could utilize more datasets (i.e. containing sufficient audio samples but without separation data) in the source separation task. Indeed, it also indicates that we no longer require clean sources and mixtures for the source separation task [61] if we succeed in using these datasets to achieve good performance.

3.5.6 Zero-shot Learning via Latent Source Embeddings

As shown in the right part of Figure 3.4, after the training stage, the audio localization models can also be utilized to obtain the latent source embedding e of given clips s, and send the embedding into the separator to perform the audio source separation. And in the inference stage, we utilize this model to separate more sources that are unseen or undefined in the training set.

Formally, suppose that we need to separate an audio a_q according to a query source class k_q. In order to get the latent source embedding e_q, we first need to collect N clean clips of this source $\{s_{q1}, s_{q2}, ..., s_{qN}\}$. Then we feed them into the audio representation model to obtain the latent embeddings $\{e_{q1}, e_{q2}, ..., e_{qN}\}$. The e_q is obtained by taking the average of them:

$$e_q = \frac{1}{N}\sum_{i=1}^{N} e_{qi} \tag{3.7}$$

Then, we use e_q as the query for the source k_q and separate a_q into the target track $f(x_q, e_q)$. A visualization of this process is depicted in Figure 3.4.

The 527 classes of Audioset are ranged from ambient natural sounds to human activity sounds. Most of them are not clean sources as they contain other backgrounds and event sounds. After training our model on Audioset, we find that the model is able to achieve a good performance on separating unseen sources. According to [117], we declare that this follows a Class-Transductive Instance-Inductive (CTII) zero-shot setting by training the separation model with certain types of sources and using unseen queries to generalize the model.

3.6 Experiments

There are two experimental stages for us to train a zero-shot audio source separator. First, we need to train the audio classification and localization model as the first component. Then, we train an audio source separator as the second component based on the processed data. In the following subsections, we will introduce the experiments in these two stages.

Table 3.2. The mean average precision (mAP) performance of audio classification from different baselines on the Audioset evaluation set.

Model	mAP
AudioSet Baseline [35]	0.314
DeepRes. [32]	0.392
PANN. [60]	0.434
PSLA. [38]	0.444
AST. (single) w/o. pretrain [37]	0.366
AST. (single) [37]	0.459
768-d HTS-AT	**0.471**
768-d HTS-AT w/o. pretrain	0.453
2048-d HTS-AT w/o. pretrain	0.459

3.6.1 Audio Classification and Localization

Dataset and Training Details

Similar to Chapter 2, we train both HTS-AT and PANN on AudioSet, with 2 million 10-sec audio samples and labeled with a set of 527 sound labels. We use the full=train set of AudioSet (2M samples) for training and the evaluation set (22K samples) for evaluation. For the pre-processing of audio, all samples are converted to mono as 1 channel by 32,000 Hz sampling rate. To compute STFTs and mel-spectrograms, we use 1024 window size and 320 hop size. As a result, each frame is $\frac{320}{32000} = 0.01$ sec. The number of mel-frequency bins is $F = 64$. Each 10-sec sample constructs 1000 time frames and we pad them with 24 zero-frames ($T = 1024$). The shape of the output featuremap is $(1024, 527)$ ($K = 527$). The patch size is 4×4 and the time window is 256 frames in length. Different from Chapter 2, we further propose two settings for HTS-AT with a latent dimension size H of 768 or 2048. We adopt the 768-d model to make use of the swin-transformer ImageNet-pretrained model for achieving a potential best result. And we adopt the 2048-d model in the following separation experiment because it shares the consistent latent dimension size with PANN's. We set 4 network groups in the HTS-AT, containing 2,2,6, and 2 swin-transformer blocks respectively.

We implement both PANN and HTS-AT in PyTorch, train it with a batch size of 128 and the AdamW optimizer (β_1=0.9, β_2=0.999, eps=1e-8, decay=0.05) [56] in 8 NVIDIA Tesla V-100 GPUs in parallel. We adopt a warm-up schedule by setting the learning rate as 0.05, 0.1, 0.2 in the first three epochs, then the learning rate is halved every ten epochs until it returns to 0.05.

AudioSet Results

Following the standard evaluation pipeline, we use the mean average precision (mAP) to verify the classification performance on Audioset evaluation set. In Table 3.2, we compare the HTS-AT with previous SOTAs including PANN, PSLA, and AST. Among all models, PSLA, AST, and our 768-d HTS-AT apply the ImageNet-pretrained models. Specifically, PSLA uses the pretrained EfficientNet [107]; AST uses the pretrained DeiT; and 768-d HTS-AT uses the pretrained swin-transformer in Swin-T/C24 setting[1]. We also provide the mAP result of the 768-d HTS-AT without pretraining for comparison. For the 2048-d HTS-AT, we train it from zero because there is no pretrained model. For the AST, we compare our model with its single model's report instead of the ensemble one to ensure the fairness of the experiment. All HTS-ATs are converged around 30-40 epochs in about 20-hour training.

Table 3.2 presents the mAP results of different models on the AudioSet evaluation set, which is consistent to Table 2.1. The 768-d HTS-AT achieves a new mAP SOTA as 0.471 on Audioset, additionally, the newly-trained 2048-d HTS-AT without pretraining weights can also achieve the pre-SOTA mAP as 0.459. This indicates that both HTS-ATs in 768-d and 2048-d settings are capable of being utilized in the audio source separation process to provide the audio embedding as the separation condition.

[1] https://github.com/microsoft/Swin-Transformer

3.6.2 Audio Source Separation

Dataset and Training Details

We train our audio separator in AudioSet full-train set, validate it in Audioset evaluation set, and evaluate it in MUSDB18 test set as following the 6th community-based Signal Separation Evaluation Campaign (SiSEC 2018). MUSDB18 contains 150 songs with a total duration of 3.5 hours in different genres. Each song provides a mixture track and four original stems: vocal, drum, bass, and other. All SOTAs are trained with MUSDB18 training set (100 songs) and evaluated in its test set (50 songs). Different from these SOTAs, we train our model only with Audioset full-train set other than MUSDB and directly evaluate it in MUSDB18 test set.

Since Audioset is not a natural separation dataset (i.e., no mixture data), to construct the training set and the validation set, during each training step, we sample two classes from 527 classes and randomly take each sample a_1, a_2 from two classes in the full-train set. We implement a balanced sampler that all classes will be sampled equally during the whole training. During the validation stage, we follow the same sampling paradigm to construct 5096 audio pairs from Audioset evaluation set and fix these pairs. By setting a fixed random seed, all models will face the same training data and the validation data.

We compare two audio localization system in the source separation pipeline: PANN or HTS-AT . The separator comprises 6 encoder blocks and 6 decoder blocks. In encoder blocks, the numbers of channels are namely 32, 64, 128, 256, 512, 1024. In decoder blocks, they are reversed (i.e., from 1024 to 32). There is a final convolution kernel that converts 32 channels into the output audio channel. Batch normalization [52] and ReLU non-linearity [3] are used in each block. The final output is a spectrogram, which can be converted into the final separate audio $c\prime$ by iSTFT. Similarly, we implement our separator in PyTorch and train it with the Adam optimizer (β_1=0.9, β_2=0.999, eps=1e-8, decay=0), the learning rate 0.001 and the batch size of 64 in 8 NVIDIA Tesla V-100 GPUs in parallel.

Evaluation Metrics

We use source-to-distortion ratio (SDR) as the metric to evaluate our separator. For the validation set, we compute three SDR metrics between the prediction and the groundtruth in different separation targets:

- The target of mixture-SDR: $f(s, e_j) \mapsto s_j$
- The target of clean-SDR: $f(s_j, e_j) \mapsto s_j$
- The target of silence-SDR: $f(s_{\neg j}, e_j) \mapsto \mathbf{0}$

Where the symbol $\neg j$ denotes any clip which does not share the same class with the j-th clip. In our setting, $\neg 1 = 2$ and $\neg 2 = 1$. The clean SDR is to verify if the model can maintain the clean source given the self latent source embedding. The silence SDR is to verify if the model can separate nothing if there is no target source in the given audio. These help us understand if the model can be generalized to more general separation scenarios only by using the mixture training. For the testing, we only compute the mixture SDR between each stem and each original song in MUSDB18 test set. Each song is divided into 1-sec clips. The song's SDR is the median SDR over all clips. And the final SDR is the median SDR over all songs.

The Choice of Source Embeddings

We choose three source embeddings for our separator: (1) the 527-d presence probability vector from PANN, referring to [61]; (2) the 2048-d latent embedding from PANN penultimate layer; and (3) the 2048-d latent embedding from HTS-AT. This helps to verify if the latent source embedding can perform a better representation for separation, and if the embedding from HTS-AT is better than that from PANN.

In the training and validation stage, we get each latent source embedding directly from each 2-sec clip according to the pipeline in Figure 3.1. After picking the best model in the validation set, we follow Figure 3.4 to get the query source embeddings in MUSDB18. Specifically,

Table 3.3. The SDR performance of different models with different source embeddings in the validation set.

Validation Set: AudioSet Evaluation Set			
Metric-SDR: dB	mixture	clean	silence
527-d PANN-SEP [61]	7.38	8.89	11.00
2048-d PANN-SEP	9.42	13.96	15.89
2048-d HTS-AT-SEP	**10.55**	**27.83**	**16.64**

we collect all separate tracks in the highlight version of MUSDB10 training set (30 secs in each song, 100 songs in total) and take the average of their embeddings on each source as four queries: vocal, drum, bass, and other.

Separation Results

Table 3.3 shows the SDRs of two models in the validation set. We could clearly figure out that when using the 2048-d latent source embedding, PANN achieves better performance in increasing three types of SDR by 2-4 dB than that of 527-d model. A potential reason is that the extra capacity of the 2048-d embedding space helped the model better capture the feature of the sound comparing to the 527-d probability embedding. In that, the model can receive more discriminative embeddings and perform a more accurate separation.

Then we pick the best models of 527-d PANN-SEP, 2048-d PANN-SEP, 2048-d HTS-AT-SEP and evaluate them in MUSDB18. As shown in Table 3.4, there are three categories of models: (1) Standard Model: these models can only separate one source, in that they need to train 4 models to separate each source in MUSDB18. (2) Query-based Model: these models can separate four sources in one model. Both models in (1) and (2) require the training data in MUSDB training set and cannot generalize to separate other sources. And (3) Zero-shot Model: our proposed models can separate four sources in one model without any MUSDB18 training data. Additionally, they can even separate more sources. Specifically, for our proposed 2048-d HTS-AT model, we repeat the training three times with different random seeds.

From Table 3.4 our proposed model 2048-d HTS-AT-SEP outperforms PANN-SEP

Table 3.4. The SDR performance in MUSDB18 test set. All models are categorized into three slots.

Standard SOTA Model				
Median SDR	vocal	drum	bass	other
WaveNet [103]	3.25	4.22	3.21	2.25
WK [120]	3.76	4.00	2.94	2.43
RGT1 [90]	3.85	3.44	2.70	2.63
SpecUNet [69]	5.74	4.66	3.67	3.40
MMDLSTM [105]	**6.60**	6.41	5.16	4.15
Open Unmix [104]	6.32	5.73	5.23	4.02
Query-based Model w/. MUSDB18 Training				
Median SDR	vocal	drum	bass	other
AQMSP [66]	4.90	4.34	3.09	3.16
Meta-TasNet [96]	6.40	5.91	5.58	**4.19**
Zero-shot Model w/o. MUSDB18 Training				
Median SDR	vocal	drum	bass	other
527-d PANN-SEP	4.16	0.95	-0.86	-2.65
2048-d PANN-SEP	6.06	5.00	3.38	2.86
2048-d HTS-AT-SEP	6.15	5.44	3.80	3.05

models in all SDRs (6.15, 5.44, 3.80, 3.05). The deviation of SDR performance on four stems are $\pm$0.22, $\pm$0.32, $\pm$0.23, and $\pm$0.20. The SDRs in vocal, drum, and bass are compatible with standard and query-based SOTAs. However, we observe a relatively low SDR in the "other" source. One possible reason is that the "other" embedding we calculate for MUSDB18 is not general because it denotes different instruments and timbres in different tracks. Another possible reason is that the separation quality is related to the random combination of training data, and different orders may cause differences on some specific types of sounds. These sub-topics can be further researched in the future.

In summary, the most novel and surprising observation is that our proposed audio separator succeeds in separating 4 sources in MUSDB18 test set without any of its training data but only Audioset. The model performs as a zero-shot separator by using any latent source embedding collected from accessible data, to separator any source it faces.

Table 3.5. The SDR performance of the 2048-d HTS-AT-SEP in the zero-shot verification experiment.

Class	Conversation	Whisper	Clap	Cat	Orchestra	Aircraft	Engine	Pour	Scratch	Creak
Mixture-SDR	9.08	8.04	9.67	9.49	9.18	8.47	8.31	7.92	8.42	6.56
Clean-SDR	17.44	10.50	17.78	15.01	10.06	13.09	14.85	14.28	15.52	13.79
Silence-SDR	14.05	13.86	14.45	17.63	12.08	11.97	11.56	12.76	13.95	13.61

3.6.3 Zero-Shot Verification

In this section, we conduct another experiment to separate sources that are held-out from training. We first select 10 sound classes in Audioset. Then during the training, we remove all data of these 10 classes. The model only learns how to separate clips mixed by the left 517 classes. During the evaluation, we construct 1000 (100×10) mixture samples in Audioset evaluation set whose constituents only belong to these 10 classes. Then we calculate the mixture SDR, the clean SDR, and the silence SDR of them.

Table 3.5 shows the results by the 2048-d HTS-AT model. We can find that the model can still separate the held-out sources well by achieving the average mixture SDR, clean SDR, and silence SDR as 8.52 dB, 14.23 dB, and 13.59 dB (calculated under 10 classes). The detailed SDR distribution of these 1000 samples is depicted in the open source repository. The intrinsic reason for this good performance is that the SED system captures many features of 517 sound classes in its latent space. And it generalizes to regions of the embedding space it never saw during training, which the unseen 10 classes lie in. Finally, the separator utilizes these features in the embedding to separate the target source. The zero-shot setting of our model is essentially built by a solid feature extraction mechanism and a latent source separator.

3.6.4 Visualization of Hierarchical Separation

One application of the proposed zero-shot audio source separation model is to separate arbitrary audio recordings into individual sources with sound ontology. Figure 3.5 shows the automatically detected and separated waveforms of a movie clip from *Harry Potter and the Sorcerer's Stone* from the AudioSet sound ontology from level 1 to level 3. Level 1 indicates

coarse sound classes and level 3 indicates fine sound classes. In level 1, the model successfully separates human sounds, music and sounds of things. In level 2, the model further separates human group actions, vehicle, and animals. In level 3, the model separates fine-grained sound classes such as bell, bird, crowd, and scary music. We can observe that the proposed separation model works well on different hierarchies of sound classes without knowing any source information of the movie soundtrack. This shows a broad potential application of such models in the practical usages of audio analysis, understanding and editing on arbitrary soundtracks.

3.7 Conclusion

In this chapter, we further explore the effectiveness of HTS-AT on addressing the challenges in the field of audio source separation. We introduce a zero-shot audio source separator capable of leveraging weakly labeled data for training, targeting different sources for separation, and accommodating unseen sources based on content-level information from the audio representation model. Training the entire system on Audioset and evaluating it on the MUSDB18 dataset, experimental results demonstrate that our system achieves performance comparable to standard supervised models. Additionally, We verify our system in a complete zero-shot setting to prove its generalization ability. With this system, more weakly labeled audio data can be utilized for source separation, enabling separation of additional sources within this pipeline. These results further establish the efficacy of HTS-AT as the proposed audio representation model, capable of capturing both content-level and semantic information of audio signals, and its significant contribution to other fields of music and audio signal processing.

This chapter contains some materials (texts, tables, and figures) from a published conference paper: Ke Chen, Xingjian Du, Bilei Zhu, Zejun Ma, Taylor Berg-Kirkpatrick, Shlomo Dubnov, *Zero-shot Audio Source Separation through Query-based Learning from weakly labeled Data*, in proceedings of AAAI Conference on Artificial Intelligence Conference, AAAI 2022; and a preprint online paper: Qiuqiang Kong*, Ke Chen*, Haohe Liu, Xingjian Du, Taylor

Berg-Kirkpatrick, Shlomo Dubnov, Mark D Plumbley, *Universal Source Separation with Weakly-Labelled Data*, in the arXiv preprint 2305.07447. The dissertation author was the first author or the co-first-author of these publications.

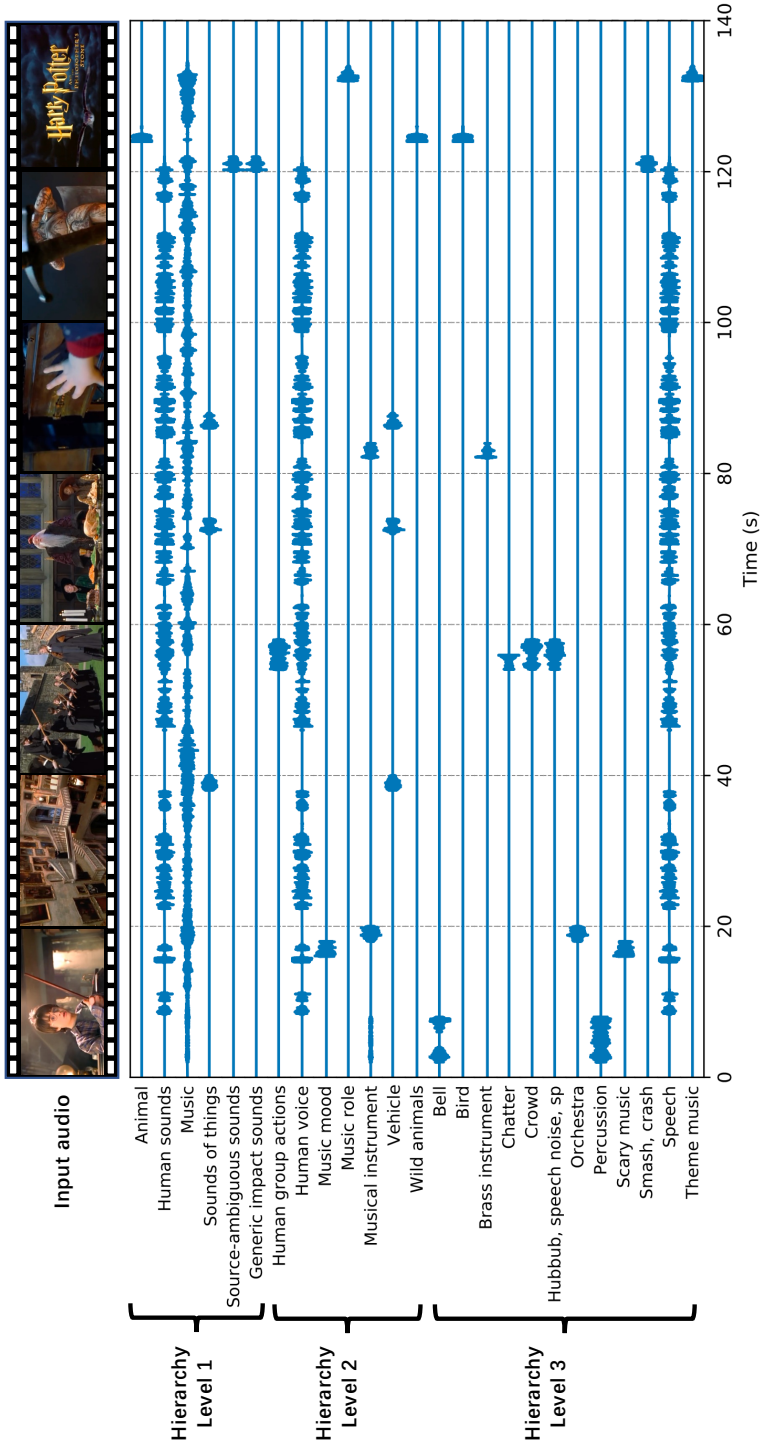


Figure 3.5. The visualization of zero-shot audio source separation performed on the trailer of "Harry Potter and the Sorcerer's Stone": https://www.youtube.com/watch?v=VyHV0BRtdxo

Chapter 4

Contrastive Language-Audio Pretraining

In Chapter 3, HTS-AT proves to be an efficient and effective audio representation model. It excels not only in capturing semantic information for tasks like audio classification and localization but also in preserving content-level information crucial for addressing challenges in zero-shot audio source separation. The experimental results underscore our confidence in leveraging the potential of HTS-AT to enhance various audio and music applications, spanning classification, separation, recommendation, and even generation tasks.

However, the new challenges come out as we dive deeply into the nuanced designs and the applications of HTS-AT. While the ability to capture both semantic and content-level information from audio signals marks a promising start for an audio representation model, a significant limitation persists in its capacity to bridge **other modalities**. It is noted that many applications of audio and music processing extend beyond a singular modality (i.e., audio itself). For example, content creators often leverage text descriptions and images as "hints" to guide the music generation and editing process. Similarly, users often engage in conversations with AI-agents to gradually uncover their music preference in the recommendation scenarios. Such applications are not currently doable within the single architecture of HTS-AT, which relies solely audio information and is not able to accept inputs from other modalities like texts or images. Prior to exploring further applications of HTS-AT, it is imperative to revisit the design of HTS-AT and equip it with the capability to incorporate audio with other modalities.

Among all modalities in the world, we focus on one of the primary resources — language. Language serves as a central communication medium for human beings, and large language models (LLMs) such as ChatGPT [1] have emerged as increasingly promising tools for bridging new knowledge. In this chapter, we explore how we can integrate HTS-AT with the language modality to spark innovation in audio and music technologies and applications.

4.1 Introduction

Audio is one of the most common information types in the world alongside text and image data. However, different audio tasks typically require finely-annotated data, which limits the amount of available audio data due to the labor-intensive collection procedure. Consequently, designing an effective audio representation for many audio tasks without requiring a lot of supervision remains a challenge.

The contrastive learning paradigm is a successful solution for training a model on large-scale noisy data collected from internet. The recently proposed Contrastive Language-Image Pretraining (CLIP) [85] learns the correspondence between text and image by projecting them into a shared latent space. The training is conducted by regarding the ground-truth image-text pair as the positive sample and left as negative. In contrast to training on uni-modal data, CLIP is not constrained by data annotation and shows great robustness by achieving high accuracy in a zero-shot setting on out-of-domain variations of ImageNet dataset [20]. Additionally, CLIP shows great success in downstream tasks such as text-to-image retrieval and text-guided captioning. Similar to vision, audio and natural languages also contain overlapping information. In audio event classification task, for instance, some text descriptions of an event can be mapped to the corresponding audio. These text descriptions share a similar meaning that could be learned together with the related audio to form an audio representation of crossmodal information. Additionally, training such a model requires simply paired audio and text data, which is easy to collect.

Several recent studies [40, 26, 21, 124, 81, 76, 123] have presented the prototype of the contrastive language-audio pretraining model for the text-to-audio retrieval task. [124] utilizes Pretrained Audio Neural Network (PANN) [60] as the audio encoder, BERT [22] as the text encoder, and several loss functions to evaluate the text-to-audio retrieval performance. [21] further ensemble our proposed HTS-AT and RoBERTa [70] into the encoder list to further enhance performance. Then, [26] investigates the effectiveness of the learned representation in the downstream task of audio classification. Some other studies, such as AudioClip [40] and WaveCLIP [123], focus more on the contrastive image-audio (or image-audio-language) pretraining model. All these models show great potential for contrastive learning in the audio domain.

Nonetheless, current studies have not shown the full strength of the language-audio contrastive learning. First, the models mentioned above are trained on relatively small datasets, showing that large-scale data collection and augmentation for training are needed. Second, prior work lacks a full investigation of selections and hyperparameter settings of audio/text encoders, which is essential for determining the basic contrastive language-audio architecture. Third, the model struggles to accommodate varied audio lengths, particularly for the transformer-based audio encoder. There should be a solution to handle audio inputs of variable-length. Finally, the majority of language-audio model studies focuses solely on text-to-audio retrieval without assessing their audio representations in downstream tasks. As a representation model, we expect more discoveries of its generalization ability to more downstream tasks.

In this chapter, we make contributions to improve the dataset, model design and the experiment setting from above concerns based on the HTS-AT representation model:

- We release LAION-Audio-630K, currently the largest public audio caption dataset of 633,526 audio-text pairs. To facilitate the learning process, we employ the keyword-to-caption model to augment labels of AudioSet [35] into corresponding captions. This dataset can also contribute to other audio tasks.

- We construct a pipeline of contrastive language-audio pretraining, as CLAP. Two audio encoders (HTS-AT, PANN) and three text encoders (BERT,RoBERTa, CLIP-Transformer) are selected for testing. We employ feature fusion mechanisms to enhance the performance and enable our model to handle variable-length inputs.

- We conduct comprehensive experiments on the model, including the text-to-audio retrieval task, as well as zero-shot and supervised audio classification downstream tasks. We demonstrate that scaling of the dataset, keyword-to-caption augmentation, and feature fusion can improve the performance of the model in different perspectives. It achieves the state-of-the-art (SOTA) in the text-to-audio retrieval and audio classification tasks, even comparable to the performance of supervised models.

4.2 Supplementary Materials

The code implementation of CLAP and its pretrained weights of different settings are released in https://github.com/LAION-AI/CLAP. The dataset LAION-Audio-630K is released in https://github.com/LAION-AI/audio-dataset.

4.3 LAION-Audio-630K and Training Dataset

4.3.1 LAION-Audio-630K

We collect LAION-Audio-630K, a large-scale audio-text dataset consisting of 633,526 pairs with the total duration of 4,325.39 hours. It contains audios of human activities, natural sounds and audio effects, consisting of 8 data sources from publicly available websites. We collect these datasets by downloading audios and relevant text descriptions. Based on our current knowledge, LAION-Audio-630K is the largest audio-text dataset publicly available and a magnitude larger than previous audio-text datasets as shown in Table 4.1.

Table 4.1. The illustration of LAION-Audio-630K dataset and its comparison to existing datasets.

Dataset	Pairs	Duration (hours)
Clotho [25]	5,929	37.00
SoundDescs [58]	32,979	1060.40
AudioCaps [55]	52,904	144.94
LAION-Audio-630K (ours)	633,526	4325.39

4.3.2 Training Dataset

To test how model performance will scale on different sizes and types of dataset, we use three training set setting in the paper, varying from small to large size. These settings employ three datasets:

1. AudioCaps and Clotho (**AC+CL**) [55, 25] consist of 55K training samples of audio-text pairs.

2. LAION-Audio-630K (**LA.**) consists of around 630K audio-text pairs.

3. Audioset [35] consists of around 2 million audio samples with only labels available for each audio track.

When processing these datasets, we exclude all overlapping data in evaluation sets. More details of the training datasets can be found in section 4.8.1.

4.3.3 Dataset Format and Preprocessing

All audio files used in this work are preprocessed to mono channel at a sample rate of 48,000 Hz in FLAC format. For datasets with only tags or labels available, we extend labels into captions using the template "The sound of `label-1`, `label-2`, ..., and `label-n`" or the keyword-to-caption model (detail in section 4.5.1). As a result, we can leverage more data into the training of the contrastive language-audio pretraining model. Combining all the datasets, we increase the total number of audio samples with text caption to around 2.6 million.

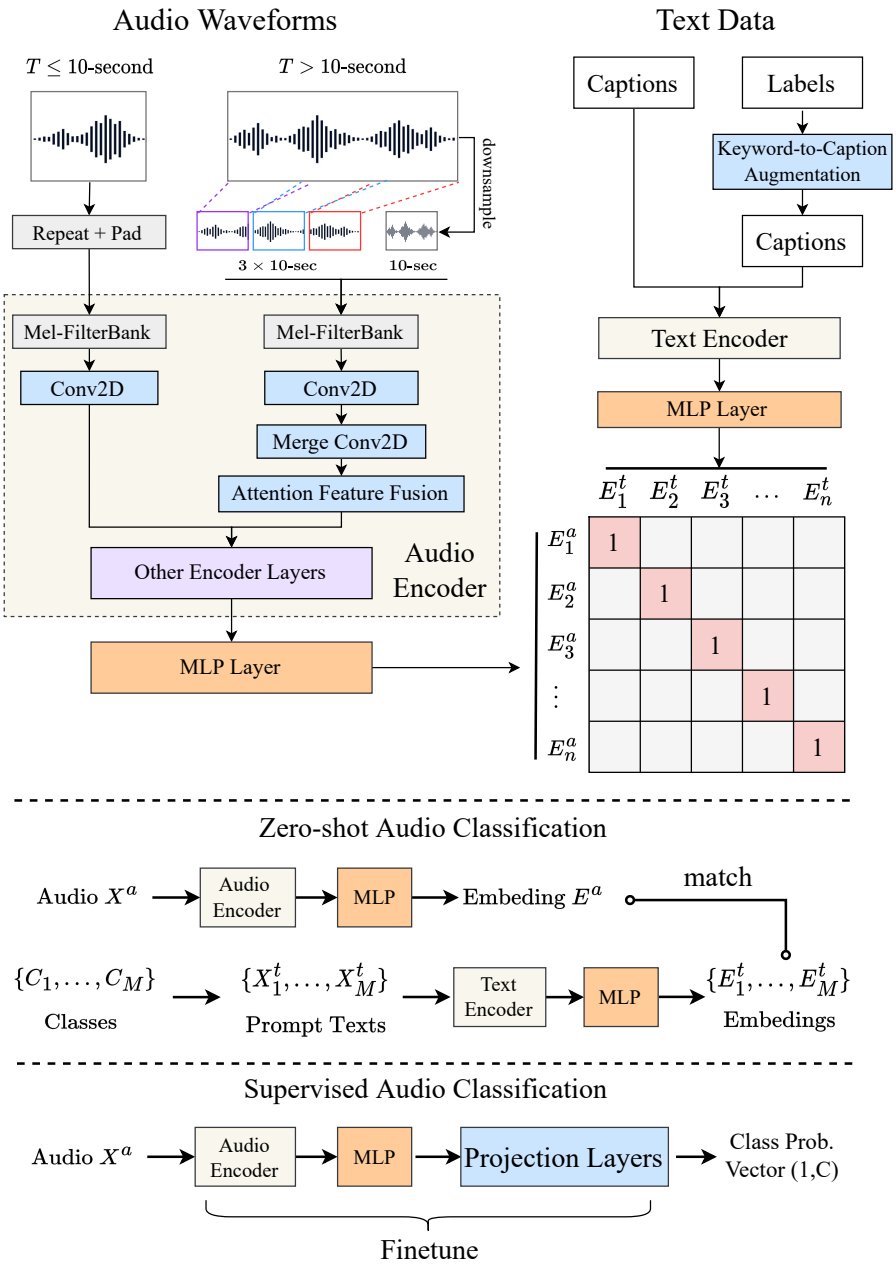


Figure 4.1. The architecture of our proposed contrastive language-audio pretraining model (CLAP) based on HTS-AT, including audio encoders, text encoders, feature fusion, and keyword-to-caption augmentation.

4.4 Model Architecture

4.4.1 Contrastive Language-Audio Pretraining

Figure 4.1 depicts the general architecture of our proposed contrastive language-audio encoder model. Similar to CLIP [85], we have two encoders to separately process the input of audio data X_i^a and text data X_i^t, where (X_i^a, X_i^t) is one of audio-text pairs indexed by i. The audio embedding E_i^a and the text embedding E_i^t are respectively obtained by the audio encoder $f_{audio}(\cdot)$ and the text encoder $f_{text}(\cdot)$, with projection layers:

$$E_i^a = MLP_{audio}(f_{audio}(X_i^a)) \tag{4.1}$$

$$E_i^t = MLP_{text}(f_{text}(X_i^t)) \tag{4.2}$$

Where the audio/text projection layer is a 2-layer multi-layer perceptron (MLP) with ReLU [3] as the activation function to map the encoder outputs into the same dimension D (i.e., $E_i^a, E_i^t \in \mathbb{R}^D$).

The model is trained with the contrastive learning paradigm between the audio and text embeddings in pair, following the same loss function (i.e., CLIP-Loss) in [85]:

$$L = \frac{1}{2N}\sum_{i=1}^{N}(\log\frac{\exp(E_i^a \cdot E_i^t/\tau)}{\sum_{j=1}^{N}\exp(E_i^a \cdot E_j^t/\tau)} + \log\frac{\exp(E_i^t \cdot E_i^a/\tau)}{\sum_{j=1}^{N}\exp(E_i^t \cdot E_j^a/\tau)}) \tag{4.3}$$

Where τ is a learnable temperature parameter for scaling the loss. Two logarithmic terms consider either audio-to-text logits or text-to-audio logits. N is usually the number of data, but during the training phase, N is used as the batch size, as we cannot compute the whole matrix of all data but update the model by batch gradient descent.

After we train the model, the embeddings (E^a, E^b) can be used for different tasks as shown in Figure 4.1 and listed in the below subsection.

4.4.2 Downstream Tasks in Inference Stage

Text-to-Audio Retrieval

The target audio embedding E_p^a can find the nearest text embedding E_q^t among M texts $E^t = \{E_1^t, ..., E_M^t\}$ by the cosine similarity function, determining the best match.

Zero-shot Audio Classification

For M audio classes $C = \{C_1, ..., C_M\}$, we can construct M prompt texts $X^t = \{X_1^t, ..., X_M^t\}$ (e.g., "the sound of `class-name`"). For a given audio X_p^a, we determine the best match X_q^t among X^t by the cosine similarity function over their embeddings. One advantage of using the contrastive language-audio pretraining is that the categories of audio are unrestricted (i.e., zero-shot) since the model can convert the classification task into the text-to-audio retrieval task.

Supervised Audio Classification

After training the model, for a given audio X_p^a, its embedding E_p^a can be further mapped into a fixed-category classification task by adding a projection layer at the back and finetuning (i.e., the non-zero-shot setting).

4.4.3 Audio Encoders and Text Encoders

Similar to Chapter 3, we select two models, PANN [60] and the proposed HTS-AT, to construct the audio encoder. PANN is a CNN-based audio classification model with 7 downsampling CNN blocks and 7 upsampling blocks. HTS-AT is a transformer-based model with 4 groups of swin-transformer blocks [71] and a token-semantic module [34]. For both of them, we use the output of the penultimate layer, a H-dimension vector as the output sent to the projection MLP layer, where $H_{PANN} = 2048$ and $H_{HTS-AT} = 768$.

We select three models, CLIP transformer [85] (text encoder of CLIP), BERT [22], and RoBERTa [70], to construct the text encoder. The output dimension of text encoders is respectively $H_{CLIP} = 512$. $H_{BERT} = 768$, and $H_{RoBERTa} = 768$. We apply both 2-layer MLPs

with ReLU activation [3] to map both audio and text outputs into 512 dimensions, which is the size of audio/text representations when training with the contrastive learning paradigm.

4.4.4 Feature Fusion for Variable-Length Audio

Unlike image data that can be resized to a unified resolution, audio has a nature of variable length. Conventionally, one would input the full audio into the audio encoder and take the average of per-frame or per-chunk audio embeddings as output (i.e., slice & vote). However, the conventional method is computationally inefficient on long audio.

As shown in the left of Figure 4.1, we train and inference on different lengths of audio inputs in constant computation time by combining both coarsely global and randomly sampled local information. For an audio in T seconds and a fixed chunk duration $d = 10$ seconds:

- $T \leq d$: we first repeat the input, then pad it with zero values. For example, a 3-second input will be repeated as $3 \times 3 = 9$-second and padded with 1-second zero values.

- $T > d$: we first downsample the input from T to d-second as a global input. Then we randomly slice three d-second clips, respectively from the front $\frac{1}{3}$, middle $\frac{1}{3}$ and back $\frac{1}{3}$ of the input, as local inputs. We send these $4 \times d$ inputs into the first layer of audio encoder to get the initial features, then three local features will be further converted to one feature by another 2D-Convolution layer with 3-stride in the time axis. Finally, the local feature X^a_{local} and the global feature X^a_{global} will be fused as:

$$X^a_{fusion} = \alpha X^a_{global} + (1 - \alpha) X^a_{local} \tag{4.4}$$

  Where $\alpha = f_{AFF}(X^a_{global}, X^a_{local})$ is a factor obtained by attention feature fusion (AFF) [19], a two-branch CNN model for learning the fusion factor of two inputs. Comparing with the "slice & vote" method, the feature fusion also saves the training time as we only process audio slices in the first few layers.

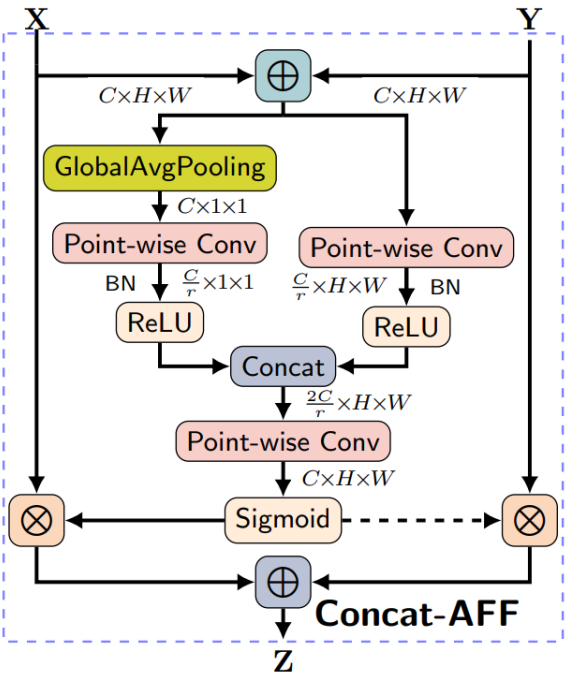


Figure 4.2. The model architecture of attentional feature fusion. This illustration is referred from [19].

4.5 Attentional Feature Fusion

The attentional feature fusion (AFF), a two-branch CNN network, to combine the global information and the local information of input audios (originally the images in the computer vision research) together.

As shown in Figure 4.2, the fusion architecture accepts two inputs: X is the global information (X^a_{global}), and Y is the merged local information (X^a_{local}) Two inputs are sent to two CNN networks to generate the coefficient α, then X and Y are added by this coefficient as presented in Equation 4.4.

No.	Keywords	T5_raw_sentence	T5_post_sentence
1	"washing machine door", "drop lid close", "thud", "hollow metal impacts", "hits"	a woman closes her eyes and thuds the lid of a washing machine after an impact with metal.	a person closes their eyes and thuds the lid of a washing machine after an impact with metal.
2	"Tools", "misc-tools", "canon calculator", "desktop electronic type with roll printer", "buttons click without printing"	A man is typing on a computer desktop with canons and other electronic tools and clicking on the buttons.	A person is typing on a desktop with a canon and clicking on the buttons.
3	"Tools", "hand-tools", "rock chiseling", "sharp metal hits", "hammer impacts on chisel", "various types of rock", "clinking", "ringing"	A man chiseling metal with a hammer and various types of tools hits a rock and clinks it.	A person chiseling metal with a hammer and various types of tools hits a rock and clinks it.
4	"spago", "las vegas restaurant", "balcony", "footsteps and shuffling movements", "crowd walla", "reverberant", "plates and glasses clanking", "phone ringing", "loop"	a woman shuffling plates and glasses with a reverberant ringing phone in a restaurant in las vegas	a person shuffling plates and glasses with a reverberant ringing phone in a restaurant in las vegas
5	"Materials", "rope", "foley", "rope", "whoosh", "spin", "twirl", "reel", "whip spin"	A man whooshs his rope and spins it around on a reel.	A person whoosh spins a reel of rope.
6	"library main entrance ambience", "busy", "footsteps", "voices", "walla", "distant door open and close", "large", "reverberant", "loop"	a woman opens a door and closes it with a large reverberant voice in the distance	a person opens a door and closes it with a large reverberant voice in the distance
7	"Guns", "bullets", "bullet drops", ".45 cartridge drops to concrete", "metal clinks"	A man drops a bullet from a gun. A man drops a cartridge from a tin to the concrete and clinks it to the metal.	A person drops a bullet into a crowd. A person drops a cartridge from a tin to the concrete and clinks it to the metal.
8	"Fire", "misc-fire", "science fiction blow torch cutting flame sizzle", "blow torch flame sizzle", "tool"	a man blows a torch and blows off the flames with science fiction tools.	a person blows a torch and blows it with flames that sizzle in science fiction
9	"bathroom stall door", "metal door bump", "push", "thump"	A woman pushes a metal thump on the door of a bathroom.	A person pushes a metal thump on the door of a bathroom.
10	"ambience", "dungeon", "screams", "chains", "water drips", "light wind", "wind"	a woman opens a door and closes it with a large reverberant voice in the distance	a person opens a door and closes it with a large reverberant voice in the distance

Figure 4.3. Examples of keyword-to-caption augmentation from AudioSet labels and the de-biased version for the model training.

4.5.1 Keyword-to-Caption Augmentation

As mentioned in section 4.3.1, some datasets contains reasonable labels or tags as keywords of the corresponding audios. As shown in the right of Figure 4.1, we used a pre-trained language model T5 [86] to make captions on top of these keywords. We also de-bias the output

sentence as post-processing. For example, we replace “woman” and “man” with ‘person’ as gender de-biasing. Some examples of keyword-to-caption by T5 model from AudioSet labels are provided in Table 4.3.

Additionally, when applying keyword to caption, we excluded samples shorter than 2 seconds, as we found in such case the audio is merely a single event, thus matching poorly with the caption generated. When using keyword to caption in training dataset including audioset, we use only the captions generated by keyword to caption and exclude the captions generated by template.

4.6 Experiments

In this section, we conduct three experiments on our proposed model. First, we train with different audio and text encoders to find the best baseline combination. Then, we train our model on various dataset size, with the feature fusion and keyword-to-caption augmentation to verify the efficacy of the proposed methods.

For the first two experiments, we evaluate our model via recall and mean average precision (mAP) on audio-to-text and text-to-audio retrieval. Lastly, we use the best model to conduct zero-shot and supervised audio classification experiments to evaluate the generalization ability to the downstream tasks.

4.6.1 Hyperparameters and Training Details

As mentioned in section 4.3.2, we use AudioCaps, Clotho, LAION-Audio-630K, along with the additional dataset — AudioSet by keyword-to-caption augmentation, to train our model. For the audio data, we use 10-second input length, 480 hop size, 1024 window size, 64 mel-bins to compute STFTs and mel-spectrograms. As the result, each input sent to the audio encoder is of the shape $(T = 1024, F = 64)$. For the text data, we use a maximum token length of 77.

When training the model without the feature fusion, the audio longer than 10-second will be randomly chunked to a 10-second segment. During training, we use the Adam [56] optimizer

with $\beta_1 = 0.99$, $\beta_2 = 0.9$ with a warm-up and cosine learning rate decay at a basic learning rate of 10^{-4}. We train the model using a batch size of 768 on **AudioCaps+Clotho** dataset, 2304 on training dataset containing LAION-Audio-630K, and 4608 on training dataset containing **AudioSet**. We train the model for 45 epochs.

4.6.2 Evaluation Metrics

The primary focus on assessing the efficacy of the models in terms of retrieval performance utilizes the metrics such as R@1, R@5, R@10 and Mean Average Precision (mAP). The Clotho and AudioCaps datasets, in particular, are characterized by the presence of five text ground-truths per audio sample. Therefore, in evaluating the retrieval performance on these datasets, we adopt the same metrics as used in previous studies, specifically, those outlined in [76, 81] in https://github.com/XinhaoMei/audio-text_retrieval/blob/main/tools/utils.py#L74.

For text-to-audio retrieval, we treat each text from an audio as independent test sample, and calculate the average of text-to-audio retrieval metrics on test samples that are five times the size of test set. In evaluating audio-to-text recall, the recall for each audio is calculated by taking the best audio-to-text retrieval result from the five text ground-truths. Additionally, audio-to-text Mean Average Precision (mAP) is calculated as

$$mAP@10 = \frac{1}{R}\sum_{r=1}^{10}(P(r) * rel(r)) \tag{4.5}$$

where $P(r)$ represents the precision at recall level r, and $rel(r)$ is a binary indicator of whether the text at recall level r is relevant or not.

In the case of other datasets, such as Freesound, in which there is only one text associated with each audio sample, the recall and mean average precision (mAP) are measured in the standard manner.

Table 4.2. The text-to-audio retrieval result (mAP@10) of using different audio/text encoder on AudioCaps and Clotho.

Model	AudioCaps (mAP@10)		Clotho (mAP@10)	
	A→T	T→A	A→T	T→A
PANN + CLIP Transformer	4.7	11.7	1.9	4.4
PANN + BERT	34.3	44.3	10.8	17.7
PANN + RoBERTa	37.5	45.3	11.3	18.4
HTS-AT + CLIP Transformer	2.4	6.0	1.1	3.2
HTS-AT + BERT	43.7	49.2	**13.8**	**20.8**
HTS-AT + RoBERTa	**45.7**	**51.3**	**13.8**	20.4

4.6.3 Text-to-Audio Retrieval

Audio and Text Encoders

We first conduct experiments to choose the best audio encoder and text encoder for the text-to-audio retrieval task. We combine two audio encoders with three text encoders in section 4.4.3 where both are loaded from pretrained checkpoints as the same to [81, 76, 21]. In this experiment, we only train on AudioCaps and Clotho datasets ($\sim$55K data), and report the best mAP@10 on audio-to-text (A→T) and text-to-audio (T→A) perspectives.

According to the results in Table 4.2, for audio encoder, HTS-AT performs better than PANN combined with the RoBERTa or BERT text encoder. For the text encoder, RoBERTa achieves better performance than BERT while the CLIP transformer performs the extremely worst. This coincides with the choice of text encoder in previous works [76, 26]. When further analyzing the loss convergence trends of CLIP transformer model, we find that RoBERTa is less over-fitting, while CLIP transformer is of high-over-fitting, thus resulting its low generalization performance.

Dataset Scale

Consequently, we apply HTS-AT-RoBERTa as our best model setting to conduct the text-to-audio retrieval experiments as a comprehensive evaluation in Table 4.3. We adopt the same metrics in [76, 81] to compute recall scores at different ranks in this task. In the training

Table 4.3. The text-to-audio retrieval performance on AudioCaps and Clotho datasets, where "LA." refers to LAION-Audio-630K, "template" refers to the text prompting by templates, "K2C aug." refers to the keyword-to-caption augmentation, and "fusion" refers to the feature fusion.

Model	Training Set	AudioCaps Eval.						Clotho Eval.					
		T-A Retrieval			A-T Retrieval			T-A Retrieval			A-T Retrieval		
		R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
MMT [81]	AudioCaps or Clotho	36.1	**72.0**	**84.5**	39.6	76.8	86.7	6.7	21.6	33.2	7.0	22.7	34.6
ML-ACT [76]	AudioCaps or Clotho	33.9	69.7	82.6	39.4	72.0	83.9	14.4	36.6	49.9	16.2	37.6	50.2
CLAP-HTS-AT [21]	AudioCaps + Clotho + WT5K	34.6	70.2	82.0	41.9	73.1	84.6	16.7	41.1	54.1	20.0	44.9	58.7
HTS-AT-RoBERTa	AudioCaps + Clotho	**36.7**	70.9	83.2	45.3	78.0	87.7	12.0	31.6	43.9	15.7	36.9	51.3
HTS-AT-RoBERTa	AudioCaps + Clotho + LA.	32.7	68.0	81.2	43.9	77.7	87.6	15.6	38.6	52.3	23.7	48.9	59.9
HTS-AT-RoBERTa (fusion)	AudioCaps + Clotho + LA.	36.2	70.3	82.5	45.0	76.7	88.0	**17.2**	**42.9**	**55.4**	24.2	**51.1**	**66.9**
HTS-AT-RoBERTa	ACaps. + Clotho + LA. + AudioSet (template)	34.7	70.5	83.2	45.3	79.5	89.2	16.4	39.0	51.0	21.8	44.6	60.1
HTS-AT-RoBERTa	ACaps. + Clotho + LA. + AudioSet (K2C aug.)	36.1	71.8	83.9	**46.8**	**82.9**	**90.7**	16.1	38.3	51.1	22.7	48.5	60.8
HTS-AT-RoBERTa (fusion)	ACaps. + Clotho + LA. + AudioSet (K2C aug.)	35.1	71.9	83.7	44.2	80.8	90.3	16.9	41.6	54.4	**24.4**	49.3	65.7

set, we gradually increase the scale of the dataset. We find that scaling up the dataset from "AudioCaps + Clotho" to "LA." does not improve the result on AudioCaps evaluation set but gets better performance on Clotho evaluation set, which is similar to the comparison between MMT [81] and CLAP-HTS-AT [21]. One reason is that AudioCaps contains audios similar to AudioSet on which the audio encoder's loaded checkpoint is pretrained. When the model receives more data from other sources, it increases its generalization but moves the distribution out of AudioSet data. Therefore, the performance on AudioCaps drops but that on Clotho increases a lot, demonstrating a trade-off of the model to keep the performance among different types of audios.

Keyword-to-Caption and Feature Fusion

When adding the feature fusion mechanism and keyword-to-caption augmentation to the model, we can observe that either of them improves the performance. The feature fusion is effective especially in Clotho dataset because it contains longer audio data ($>$ 10-second). When we add AudioSet into the training set with either template prompting or keyword-to-caption augmentation, we can see the performance increases again on AudioCaps while decreases on Clotho. This further confirms the trade-off performance between AudioCaps and Clotho datasets mentioned above. And the keyword-to-caption augmentation does bring in better performance than the simple template text prompting method on most metrics.

As the result, our best model outperforms previous methods on most metrics (mainly R@1=36.7% on AudioCaps and R@1=18.2% on Clotho) in the text-to-audio retrieval tasks. We show that training on large-scale datasets (LAION-Audio-630K and AudioSet with keyword-to caption augmentation), and feature fusion can effectively improve model performance.

Table 4.4. The zero-shot (ZS.) and supervised (SV.) audio classification results. The SoTA of each dataset/setting is denoted by the reference after the number.

Model	Audio Classification Dataset & Setting				
	ESC-50	US8K	VGGSound		FSD50K
	ZS.	ZS.	ZS.	SV.	SV.
Wav2CLIP [123]	41.4	40.4	10.0	46.6	43.1
AudioClip [40]	69.4	65.3	-	-	-
Microsoft [21]	82.6	73.2	-	-	58.6
CLAP	89.1	73.2	29.1	**75.4**	64.9
CLAP+Fusion	88.0	75.8	26.3	75.3	64.4
CLAP+K2C Aug.	**91.0**	**77.0**	**46.2**	75.3	59.7
SoTA*	82.6 [21]	73.2 [21]	10.0 [123]	64.1 [79]	**65.6** [62]

4.6.4 Zero-shot and Supervised Audio Classification

Zero-shot Audio Classification

To study the model generalization and robustness, we conduct zero-shot audio classification experiments on three top-performing models in previous experiments. We evaluate models on three audio classification dataset, namely ESC-50 [84], VGGSound [14], and Urbansound8K (US8K) [95]. We use **top-1 accuracy** as the metric. We classify audio by performing audio-to-text retrieval with each text corresponds to the text prompt converted from class label via "This a sound of `label`.". We noticed a dataset overlap between our training data and the zero-shot dataset we are evaluating on. We **excluded all the overlap samples** and perform zero-shot evaluation on the whole remaining dataset.

Supervised Audio Classification

We perform supervised audio classification by fine-tuning the audio encoder on VGGSound and FSD50K [28] datasets. We do not conduct this experiment on ESC-50 and Urbansound8K because the potential data leakage issue in those dataset will makes the results incomparable with the previous methods. Specially, mAP is used as the metric to evaluate FSD50K.

As shown in the in Table 4.4, our models achieves new SoTAs of zero-shot audio classification across all three datasets, demonstrating the high generalization ability of our model to unseen data. Keyword-to-Caption augmentation increases the performance of VGGSound and US8K a lot as it adds more text captions to "enrich" the text embedding space. Feature fusion not only enables the model to handle variable-length input, but also achieves better performance than previous models. Our best supervised audio classification result outperforms the current state-of-the-art on VGGSound dataset and is close to state-of-the-art on FSD50K dataset. The results verify that the proposed model also learns efficient audio representation during contrastive learning paradigm.

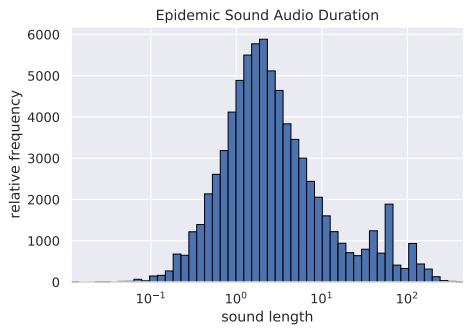
4.7 Conclusion

In this chapter, we propose a large-scale audio-text dataset and improvements on current language-audio contrastive learning paradigm. We show that LAION-Audio-630, AudioSet with keyword-to-caption augmentation, and feature fusion effectively leads to better audio understanding, task performance, and enables effective learning on variable-length data. As the result, we successfully extend the proposed HTS-AT into a more advanced audio representation model – CLAP, that is able to bridge both audio and language modalities beyond the singular audio information. With CLAP, we can anticipate more advanced applications of audio and music processing along with textual inputs.

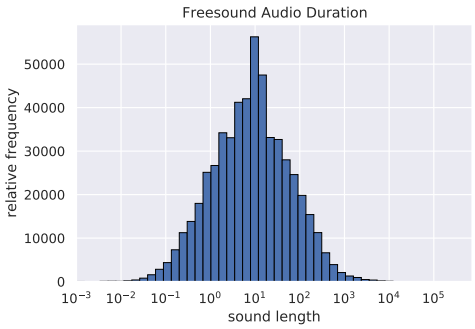
4.8 Additional Information

4.8.1 Details of LAION-AUDIO-630K

- We list the specifications of website/sources from which we collect the audio samples and text captions for LAION-Audio-630K in Table 4.5.
- We list the details of three datasets in Table 4.6. We use the combination of them to train the model in the section 4 of the submission.



(a) The audio length distribution of Epidemic Sound.



(b) The audio length distribution of Freesound.

Figure 4.4. The audio length distribution of Epidemic Sound and Freesound.

- We present the distribution of audio length on Epidemic Sound and Freesound [31] in Figure 4.4, as parts of LAION-Audio-630K, to demonstrate the existence of variable-length problem in audio data processing and model training.

The samples in Freesound dataset are collected from Freesound [31]. All audio clips from Freesound are released under Creative Commons (CC) licenses, while each clip has its own license as defined by the clip uploader in Freesound, some of them requiring attribution to their original authors and some forbidding further commercial reuse. Specifically, here is the statistics about licenses of audio clips involved in LAION-Audio-630K:

- CC-BY: 196,884
- CC-BY-NC: 63,693
- CC0: 270,843
- CC Sampling+: 11,556

We listed the licenses for each sample in our dataset release page at https://github.com/LAION-AI/audio-dataset/tree/main/laion-audio-630k.

Table 4.5. The dataset resource of LAION-Audio-630k.

Data Source	Number of Samples	Duration	Data Type
BBC sound effects	15,973	463.48hrs	1 caption per audio, audio
Free To Use Sounds	6,370	175.73hrs	Filename as caption, audio
Sonniss Game effects	5,049	84.6hrs	Filename as caption, audio
We Sound Effects	488	12.00hrs	Filename as caption, audio
Paramount Motion Sound Effects	4,420	19.49hrs	Filename as caption, audio
Audiostock	10,000	46.30hrs	1 caption per audio, audio
Freesound [31]	515,581	3003.38rs	1-2 captions per audio, audio
Epidemic Sound	75,645	220.41hrs	2 captions per audio, audio

Table 4.6. All datasets used for the training of CLAP.

Data Source	Number of Samples	Duration	Data Type
AudioCaps + Clotho			
AudioCaps	49,274	136.87hrs	1 caption per audio, audio
Clotho	3,839	23.99hrs	5 captions per audio, audio
LAION-Audio-630K			
BBC sound effects	15,973	463.48hrs	1 caption per audio, audio
Episodesound	75,645	220.41hrs	2 captions per audio, audio
freesound	414,127	2,528.15hrs	1-2 captions per audio, audio
Free To Use Sounds	6,370	175.73hrs	Filename as caption, audio
Sonniss Game effects	5,049	84.6hrs	Filename as caption, audio
We Sound Effects	488	12.00hrs	Filename as caption, audio
Paramount Motion Sound Effects	4,420	19.49hrs	Filename as caption, audio
Audiostock	10,000	46.30hrs	1 caption per audio, audio
FSD50K	36,796	70.39hrs	1 caption per audio, audio
MACS	3,537	9.825hrs	Several (2∼) captions per audio, audio
Wavtext5K	4,072	23.2hrs	1 caption per audio, audio
AudioSet			
AudioSet	1,912,024	463.48hrs	2 captions per audio,audio

4.8.2 Additional Experiment on Freesound Dataset

To further evaluate the efficacy of feature fusion, apart from AudioCaps and Clotho datasets, we further evaluate our model on Freesound evaluation set, which contains more than 10-sec audio samples (similar to Clotho dataset).

The result is shown in Table 4.7, the notation is the same as the Table 4.3 in our submission paper. The performance on Freesound dataset shares a similar trend with that on Clotho dataset:

• The performance trained on "AudioCaps + Clotho + LA." is better than that trained on

Table 4.7. The text-to-audio retrieval performance on Freesound evaluation set.

Model	Training Set	Freesound (mAP@10)	
		A→T	T→A
HTS-AT-RoBERTa	AudioCaps + Clotho + LA.	25.9	24.5
HTS-AT-RoBERTa (fusion)	AudioCaps + Clotho + LA.	**26.4**	**24.9**
HTS-AT-RoBERTa	AudioCaps + Clotho + LA. + AudioSet (K2C Aug.)	22.9	21.8
HTS-AT-RoBERTa (fusion)	AudioCaps + Clotho + LA. + AudioSet (K2C Aug.)	24.6	22.9

"AudioCaps + Clotho + LA. + AudioSet". Similar to Clotho, the Freesound dataset contains audio samples that are different from AudioSet, adding the AudioSet into the training will move the model's distribution out of general audio data to AudioSet-like audio data, such decreasing the performance.

- The performance with feature fusion is better than that without feature fusion, as the Freesound dataset contains the samples larger than 10-secs, which is the same to Clotho dataset. Their performance trend are similar.

From the above experiment, we can further conclude that the feature fusion can improve the performance of text-to-audio task (i.e., generate better audio representations) on the variable-length audio samples.

4.9 Experiment Settings on Data Exclusion

We excluded all the overlap samples and perform zero-shot evaluation on the whole remaining dataset. Table 4.8 shows the detail of it.

4.10 Reference of CLAP

This chapter contains some materials (texts, tables, and figures) from a published conference paper: Yusong Wu*, Ke Chen*, Tianyu Zhang*, Yuchen Hui*, Taylor Berg-Kirkpatrick, Shlomo Dubnov, *Large-Scale Contrastive Language-Audio Pretraining with Feature Fusion and Keyword-To-Caption Augmentation*, in proceedings of International Conference on Acoustics, Speech and Signal Processing, ICASSP 2023. The dissertation author was the co-first-author of this publication.

Table 4.8. The overlaps between the training data and the zero-shot evaluation data, we excluded all these overlaps from the evaluation sets to calculate the audio classification metrics.

Datasource A	Datasource B	Number of samples from Datasource A that are also in Datasource B
ESC50-all	Clotho-train	94
ESC50-all	Clotho-valid	27
ESC50-all	Clotho-test	34
ESC50-all	FSD50K-train	399
ESC50-all	FSD50K-valid	60
ESC50-all	FSD50K-test	171
US8K-all	Clotho-train	411
US8K-all	Clotho-valid	150
US8K-all	Clotho-test	209
US8K-all	FSD50K-train	697
US8K-all	FSD50K-valid	180
US8K-all	FSD50K-test	341
Clotho-test	FSD50K-train	54
Clotho-test	FSD50K-valid	15
Clotho-test	FSD50K-test	33
FSD50K-test	Clotho-train	137
FSD50K-test	Clotho-valid	31
FSD50K-test	Clotho-test	33
Clotho-valid	FSD50K-train	53
Clotho-valid	FSD50K-valid	10
FSD50K-valid	Clotho-train	38
FSD50K-valid	Clotho-valid	10
AudioCaps-test	Audioset-unbalanced-train	4,875
AudioCaps-test	Audioset-balanced-train	0
AudioSet-test	AudioCaps-train	0
AudioSet-test	AudioCaps-valid	0

Chapter 5

MusicLDM: Text-To-Music Generation

In Chapter 4, we analyze the limitation of the proposed HTS-AT model when it encounters the application requiring the information from multiple modalities. The refinement of HTS-AT involves connecting it with a text encoder to construct CLAP, a contrastive language-audio pretraining model capable of bridging both audio and text information. These efforts lead to a groundbreaking development in the generation task — text-to-music generation. Previously, a single HTS-AT model was inadequate for serving as an audio representation model for this application due to the absence of text input, which is addressed by CLAP.

In this chapter, we introduce the combination of the CLAP model, the variational auto-encoder module, and the audio synthesis vocoder to create a text-to-music generation model, MusicLDM. Besides, we delve into a detailed analysis of methods to prevent plagiarism in the music generation task to ensure the novelty of the generated music.

5.1 Introduction

Text-guided generation tasks have gained increasing attention in recent years and have been applied to various modalities, including text-to-image, text-to-video, and text-to-audio generation. Text-to-image generation has been used to create both realistic and stylized images based on textual descriptions, which can be useful in various scenarios including graphic design. Text-to-audio generation is a relatively new, but rapidly growing area, where the goal

is to generate audio pieces, such as sound events, sound effects, and music, based on textual descriptions. Diffusion models have shown superior performance in these types of cross-modal generation tasks, including systems like DALLE-2 [88] and Stable Diffusion [91] for text-to-image; and AudioGen [63], AudioLDM [68], and Make-an-Audio [51] for text-to-audio.

As a special type of audio generation, *text-to-music* generation has many practical applications [10, 27]. For instance, musicians could use text-to-music generation to quickly build samples based on specific themes or moods and speed up their creative process. Amateur music lovers could leverage generated pieces to learn and practice for the purpose of musical education.

However, text-to-music generation presents several specific challenges. One of the main concerns is the limited availability of text-music parallel training data [4]. Compared to other modalities such as text-to-image, there are relatively few text-music pairs available, making it difficult to train a high-quality conditional model. Large and diverse training sets may be particularly imperative for music generation, which involves many nuanced musical concepts, including melody, harmony, rhythm, and timbre. Further, the effectiveness of diffusion models trained on more modest training sets has not been fully explored. Finally, a related concern in text-to-music generation is the risk of plagiarism or lack of novelty in generated outputs [4]. Music is often protected by copyright laws, and generating new music that sounds too similar to existing music can lead to legal issues. Therefore, it is important to develop text-to-music models that can generate novel and diverse music while avoiding plagiarism, even when trained on relatively small training datasets.

In this chapter, we focus on both these challenges: we develop a new model and training strategy for learning to generate novel text-conditioned musical audio from limited parallel training data. Currently, since there is no open-source model for text-to-music generation, we first construct a state-of-the-art text-to-music generation model, MusicLDM, which adapts the Stable Diffusion [91] and AudioLDM [68] architectures to the music domain. Next, to address the limited availability of training data and to encourage novel generations, we adapt an idea from past work in other modalities: mixup [127], which has been applied to computer vision and audio

retrieval tasks, augments training data by recombining existing training points through linear interpolation. This type of augmentation encourages models to interpolate between training data rather than simply memorizing individual training examples, and thus may be useful in addressing data limitations and plagiarism in music generation. However, for music *generation*, the naive application of mixup is problematic. Simply combining waveforms from two distinct musical pieces leads unnatural and ill-formed music: tempos and beats (as well as other musical elements) are unlikely to match. Thus, we propose two novel mixup strategies, specifically designed for music generation: beat-synchronous audio mixup (BAM) and beat-synchronous latent mixup (BLM), which first analyze and beat-align training samples before interpolating between audio samples directly or encoding and then interpolating in a latent space, respectively.

We design new metrics that leverage a pretrained text and audio encoder (CLAP) to test for plagiarism and novelty in text-to-music generation. In experiments, we find that our new beat-synchronous mixup augmentation strategies, by encouraging the model to generate new music within the convex hull of the training data, substantially reduce the amount of copying in generated outputs. Further, our new model, MusicLDM, in combination with mixup, achieves better overall musical audio quality as well as better correspondence between output audio and input text. In both automatic evaluations and human listening tests, MusicLDM outperforms state-of-the-art models at the task of text-to-music generation while only being trained on 9K text-music sample pairs.

5.2 Supplementary Materials

The code implementation of MusicLDM and its pretrained weights of different settings are released in https://github.com/RetroCirce/MusicLDM. Music samples and qualitative results are available at https://musicldm.github.io.

5.3 Related Work

5.3.1 Text-to-Audio Generation

Text-to-audio generation (TTA) [68, 63, 126] is a type of generative task that involves creating audio content from textual input. In years past, text-to-speech (TTS) [89, 108] achieved far better performance than other types of audio generation. However, with the introduction of diffusion models, superior performance in various generation tasks became more feasible. Recent work has focused on text-guided generation in general audio, with models such as Diffsound [126], AudioGen [63], AudioLDM [68], and Make-an-Audio [51] showing impressive results. In the domain of music, text-to-music models include the retrieval-based MuBERT [78], language-model-based MusicLM [4], diffusion-based Riffusion [33] and Noise2Music [50]. However, a common issue with most recent text-to-audio/music models is the lack of open-source training code. Additionally, music models often requires large amounts of privately-owned music data that are inaccessible to the public, which makes it difficult for researchers to reproduce and build upon their work.

5.3.2 Plagiarism on Diffusion Models

Diffusion models have been shown to be highly effective at generating high-quality and diverse samples for text-to-image tasks. However, a potential issue with these models is the risk of plagiarism [100, 12], or the generation novelty. As stated by [100], diffusion models are capable of memorizing and combining different image objects from training images to create replicas, which can lead to highly similar or even identical samples to the training data. [12] explores different methods that could extract the training data with a generate-and-filter pipeline, showing that new advances in privacy-preserving training of diffusion models are required. Such issues are especially concerning in the domain of music, where copyright laws are heavily enforced and violations can result in significant legal and financial consequences. Therefore, there is a need to develop strategies to mitigate the risk of plagiarism in text-to-music generation

using diffusion models.

5.3.3 Mixup on Data Augmentation

Mixup [127] is a widely used data augmentation technique that has shown remarkable success in improving model generalization and mitigating overfitting. The basic principle of mixup is to linearly combine pairs of training samples to effectively construct new samples that lie on the line connecting the original samples in the feature space, encouraging the model to learn a more continuous and robust decision boundary. In this chapter, we explore the mixup technique in the context of text-to-music generation based on latent diffusion models. Different from the mixup in other modalities, music mixup involves a delicate balance of musical elements to prevent the mixed music from being chaotic noise. Moreover, in diffusion models, mixup also can refer to the combination of latent features, rather than music signals. We propose two mixup strategies tailored for music latent diffusion models and explore their potential benefits for data augmentation and generation performance.

5.4 Model Architecture

5.4.1 MusicLDM

As illustrated in Figure 5.1, MusicLDM has similar architecture as AudioLDM: a contrastive language-audio pretraining (CLAP) model in Chapter 4, an audio latent diffusion model [68] with a pretrained variational auto-encoder (VAE) [57], and a Hifi-GAN neural vocoder [59].

Formally, given an audio waveform $\boldsymbol{x} \in \mathbb{R}^T$ its corresponding text, where T is the length of samples, we feed the data into three modules:

1. We pass $\boldsymbol{x}$ through the audio encoder of CLAP (i.e., HTS-AT) $f_{audio}(\cdot)$, to obtain the semantic audio embedding $\boldsymbol{E}_x^a \in \mathbb{R}^D$, where D is the embedding dimension.

2. We pass the text of x through the text encoder of CLAP (i.e., RoBERTa) $f_{text}(\cdot)$, to to obtain the semantic text embedding $\boldsymbol{E}_x^t \in \mathbb{R}^D$.

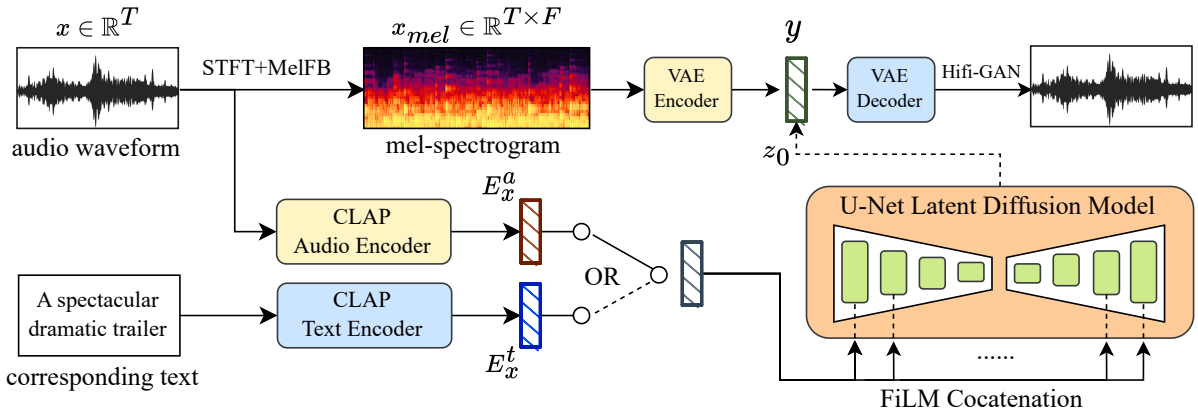


Figure 5.1. The architecture of MusicLDM, which contains a contrastive language-audio pretraining (CLAP) model, an audio latent diffusion model with VAE, and a Hifi-GAN neural vocoder.

3. We transform $\boldsymbol{x}$ into in the mel-spectrogram $\boldsymbol{x}_{mel} \in \mathbb{R}^{T \times F}$. Then we pass $\boldsymbol{x}_{mel}$ into the VAE encoder, to obtain an audio latent representation $\boldsymbol{y} \in \mathbb{R}^{C \times \frac{T}{P} \times \frac{F}{P}}$, where T is the mel-spectrogram frame size, F is the number of mel bins, C is the latent channel size of VAE, and P is the downsampling rate of VAE. The VAE is pretrained to learn to encoder and decode the mel-spectrogram of music data.

In MusicLDM, the latent diffusion model has a UNet architecture where each encoder or decoder block is composed of a ResNet layer [41] and a spatial transformer layer [91]. During the training, the semantic embedding of the input audio $\boldsymbol{E}_x$ is concatenated with the latent feature of each UNet encoder and decoder block by the FiLM mechanism [83]. The output of the diffusion model is the estimated noise $\boldsymbol{\varepsilon}_\theta(\boldsymbol{z}_n, n, \boldsymbol{E}_x)$ from n to $(n-1)$ time step in the reverse process, where θ is the parameter group of the diffusion model, and $\boldsymbol{z}_n$ is the n-step feature generated by the forward process. We adopt the training objective [46] as the mean square error (MSE) loss function:

$$L_n(\theta) = \mathbb{E}_{\boldsymbol{z}_0, \boldsymbol{\varepsilon}, n} ||\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\boldsymbol{z}_n, n, \boldsymbol{E}_x)||_2^2 \tag{5.1}$$

where $\boldsymbol{z}_0 = \boldsymbol{y}$ is the audio latent representation from VAE (i.e., the groundtruth), and $\boldsymbol{\varepsilon}$ is the target noise for training. More details regarding the training and the architecture of the latent diffusion model can be referred in section 5.5.2.

For MusicLDM, we make two changes from the text-to-audio generation scenarios to the text-to-music scenarios.

First, since the original contrastive language-audio pretraining (CLAP) model is pretrained on text-audio pair datasets dominated by sound events, sound effects and natural sounds, we retrained the CLAP on text-music pair datasets (details in section 5.5.2) to improve its understanding of music data and corresponding texts. We also retrained the Hifi-GAN vocoder on music data to ensure high-quality transforms from mel-spectrograms to music waveforms.

Second, in the original AudioLDM, the model is only fed with audio embeddings as the condition during the training process, i.e., $\boldsymbol{E}_x = \boldsymbol{E}_x^a$; and it is fed with text embeddings to perform the text-to-audio generation, i.e., $\boldsymbol{E}_x = \boldsymbol{E}_x^t$. This approach leverages the alignment of text and audio embeddings inside CLAP to train the latent diffusion model with more audio data without texts. However, this audio-to-audio training $\boldsymbol{\varepsilon}_\theta(\boldsymbol{z}_n, n, \boldsymbol{E}_x^a)$ is essentially an approximation of the text-to-audio generation $\boldsymbol{\varepsilon}_\theta(\boldsymbol{z}_n, n, \boldsymbol{E}_x^t)$. Although CLAP is trained to learn joint embeddings for text and audio, it does not explicitly enforce the embeddings to be distributed similarly in the latent space, which can make it challenging for the model to generate coherent text-to-audio outputs solely with audio-to-audio training. This problem becomes more severe when the available text-music pair data is limited. Moreover, relying solely on audio embeddings ignores the available text data, which means that we are not leveraging the full potential of our dataset. Consequently, generating accurate and realistic text-to-audio generations may not be effective.

To further investigate this task, we introduce two additional training approaches for comparison:

1. Train the MusicLDM directly using the text embedding as the condition, i.e., $\boldsymbol{\varepsilon}_\theta(\boldsymbol{z}_n, n, \boldsymbol{E}_x^t)$

2. Train the MusicLDM using the audio embedding as the condition, then finetune it with text

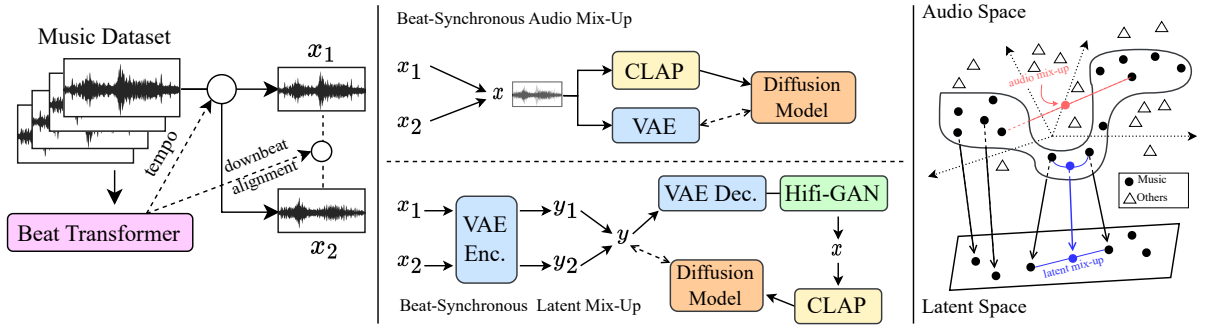


Figure 5.2. Mixup strategies. Left: tempo grouping and downbeat alignment via Beat Transformer. Middle: BAM and BLM mixup strategies. Right: How BAM and BLM are applied in the feature space of audio signals and audio latent variables.

embedding, i.e.,$\boldsymbol{\varepsilon}_\theta(\boldsymbol{z}_n, n, \boldsymbol{E}_x^a) \rightarrow \boldsymbol{\varepsilon}_\theta(\boldsymbol{z}_n, n, \boldsymbol{E}_x^t)$

The first approach follows the original target of text-to-audio, serving as a comparison with audio-to-audio training. The second approach is proposed as an improvement on audio-to-audio generation, as we shift the condition distribution from the audio embedding back to the text embedding during the training of the diffusion model.

In section 5.5.3, we compared the above two approaches with the original audio-to-audio training approaches to determine the best approach for generating high-quality and highly correlated text-to-music outputs.

5.4.2 Beat-Synchronous Mixup

As shown in Figure 5.2, we propose two mixup strategies to augment the data during the MusicLDM training: Beat-Synchronous Audio Mixup (BAM) and Beat-Synchronous Latent Mixup (BLM).

Beat-tracking via Beat Transformer

Musical compositions are made up of several elements, such as chord progressions, timbre, and beats. Of these, beats play a crucial role in determining the musical structure and alignment. In most audio retrieval tasks, mixup is a popular technique that involves randomly mixing pairs of audio data to augment the training data. However, when mixing two

music samples that have different tempos (beats per minute), the mixture can be chaotic and unappealing.

To avoid this, we use a state-of-the-art beat tracking model, Beat Transformer [128], to extract the tempo and downbeat map of each music track, as shown in the left of Figure 5.2. We categorize each music track into different tempo groups and during training, we only mixed tracks within the same tempo group to ensure the tracks were in similar tempos. Furthermore, we align the tracks by comparing their downbeat maps and selecting a certain downbeat to serve as the starting position for the mixup track. This preprocessing approach allows us to better select the music data available for mixup, resulting in mixup tracks that are neatly ordered in terms of tempo and downbeats.

Beat-Synchronous Audio Mixup

As depicted in the upper part of Figure 5.2, once we select two aligned music tracks $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, we mix them by randomly selecting a mixing ratio from the beta distribution $\lambda \sim \mathscr{B}(5,5)$, as:

$$\boldsymbol{x} = \lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2 \tag{5.2}$$

We then use the mixed data $\boldsymbol{x}$ to obtain the CLAP embedding $\boldsymbol{E}_x$ and the audio latent variable $\boldsymbol{y}$. We train the latent diffusion model using the standard pipeline. This beat-synchronous audio mixup strategy is referred to as BAM.

Beat-Synchronous Latent Mixup

As depicted in the lower part of Figure 5.2, in the latent diffusion model, the mixup process can also be applied on the latent variables, referred as beat-synchronous latent mixup (BLM). After selecting two aligned music tracks $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, we feed them into the VAE encoder

to obtain the latent variables $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$. We then apply the mixup operation to the latent variables:

$$\boldsymbol{y} = \lambda \boldsymbol{y}_1 + (1-\lambda)\boldsymbol{y}_2 \tag{5.3}$$

In contrast to BAM, BLM applies the mixup operation to the latent space of audio, where we cannot ensure that the mixture of the latent variables corresponds to the actual mixture of the music features in the appearance. Therefore, we first generate a mixed mel-spectrogram $\boldsymbol{x}_{mel}$ by feeding the mixed latent variable $\boldsymbol{y}$ into the VAE decoder. Then, we feed $\boldsymbol{x}_{mel}$ to the Hifi-GAN vocoder to obtain the mixed audio $\boldsymbol{x}$ as the input music. With $\boldsymbol{x}$ and $\boldsymbol{y}$, we follow the pipeline to train the MusicLDM.

What are BAM and BLM doing?

As shown in the right of Figure 5.2, we demonstrate the interpolation between the feature space of audio when using BAM and BLM. In the feature space of audio signals, the "•" represents the feature point of music data, while the "△" denotes the feature point of other audio signals, such as natural sound, audio activity, and noise. During the pretraining process of VAE, a latent space is constructed for encoding and decoding the music data. The VAE aims to learn the distribution of the latent variables that can best represent the original data and transform the original feature space into a lower-dimensional manifold. This manifold is designed to capture the underlying structure of the music data. Therefore, any feature point within this manifold is considered to be a valid representation of music.

BAM and BLM are concerned with augmenting the data at different levels of feature space. As shown in right of Figure 5.2, BAM linearly combines two points in audio space to form a new point on the red line. BLM, represented by the blue line, performs a similar operation, but result in a new point in the VAE-transformed latent space, which will be decoded back onto the music manifold of audio space.

Both BAM and BLM offer unique advantages and disadvantages. BAM applies mixup in

the original feature space, resulting in a smooth interpolation between feature points. However, BAM cannot ensure a reasonable music sample that lies within the music manifold. This issue is more problematic using the simple audio mixup strategy without tempo and downbeat alignments. BLM, conversely, augments within the music manifold, fostering robust and diverse latent representations. However, BLM is computationally more expensive as it requires computing the latent feature back to audio via VAE decoder and Hifi-GAN. Furthermore, when the ill-defined or collapsed latent exists in VAE, BLM may be out of effectiveness.

Both BAM and BLM are effective data augmentation techniques that encourage the model to learn a more continuous and robust decision boundary on the audio feature space, or implicitly from the latent space to the audio space, which can improve the model's generalization performance and mitigate overfitting. In the context of text-to-music generation, these mixup strategies can have a potential to mitigate the limitations of data size and help avoid plagiarism issues. By introducing small variations through mixup, the model can touch a more rich space of music data and generate music samples that are correlated to the texts but show differences to the original training data. Later in section 5.5.3, we evaluated whether these strategies mitigate the data limitation and plagiarism issues.

5.5 Experiments

In this section, we conducted three experiments on our proposed method. First, we trained MusicLDM with different mixup strategies and compared them with available baselines. Second, we evaluated MusicLDM in terms of text-music relevance, novelty and plagiarism risk via metrics based on CLAP scores. Finally, we conducted a subjective listening test to give an additional evaluation.

Table 5.1. Comparison of zero-shot classification performance of the CLAP (trained on more music data) with previous baselines.

	ESC-50	US8K	VGGSound	GTZAN
Wav2CLIP [123]	41.4	40.4	10.0	-
audioCLIP [40]	68.6	68.8	-	-
CLAP (Elizalde et al. [26])	82.6	73.2	-	25.2
CLAP (ours in Chapter 4)	91.0	77.0	46.2	71.0
CLAP (ours on music data)	90.1	80.6	46.6	71.0

5.5.1 CLAP Setup

Hyperparameters

We finetuned our pretrained CLAP model on music datasets in addition to its original training data, allowing it to better understand the relation between music and textual descriptions. The audio encoder of CLAP is HTS-AT and the text encoder is RoBERTa-base [70]. The HTS-AT has an embedding dimension of 768 and has 4 groups of swin-transformer blocks, each group has depth of $[2,2,12,2]$ and number of head in $[4,8,16,32]$. The RoBERTa consists of a transformer model with 12 layers, 8 heads, and a inner width of 512. The audio embedding and the text embedding have the dimension size $D = 512$.

Dataset and Training Details

The new CLAP model is trained on dataset of 2.8 Million text-audio pairs, including extra music data at https://github.com/LAION-AI/audio-dataset/blob/main/data_collection, with an approximate total duration of 20,000 hours. We use the batch size of 2304 and the Adam [56] optimizer with $\beta_1 = 0.99$, $\beta_2 = 0.9$ with a warm-up and cosine learning rate decay at a basic learning rate of 0.0001.

Zero-shot Classification Performance

Similarly to Chapter 4, we evaluate the performance of the newly-trained CLAP on the zero-shot audio classification tasks, namely on the benchmark datasets of ESC-50 [84], Urbansound 8K [95], and VGGSound [14]. To demonstrate that the retrained CLAP involves

more understandings of music data, we further add a music genre classification benchmark dataset GTZAN [110] into the evaluation. As shown in Table 5.1, our retained CLAP achieves best performance acoustic event classification in Urbansound 8K and VGGSound dataset, while still maintaining comparable performance in ESC-50 dataset and on par performance in GTZAN music classification dataset. Although the performance on GTZAN music dataset is not improved, the extra data used for training CLAP might result in a better representation space which is beneficial for text-to-music generation model as verified by the outperforming results on other datasets.

5.5.2 MusicLDM Setup

Hyperparameters

For audio signal processing, we use the sampling rate of 16 kHz to convert all music samples for the training of MusicLDM. Each input data is a chunk of 10.24 seconds randomly selected from the dataset, i.e., $L = 163840$. We use the hop size 160, the window size 1024, the filter length 1024, and the number of mel-bins 128 to compute the short-time Fourier transform (STFT) and mel-spectrograms. As the result, the input mel-spectrogram has the time frame $T = 1024$ and the mel-bins $F = 128$.

We adopt a convolutional VAE as the latent audio representation model, consisting of a 4-block downsampling encoder and a 4-block upsampling decoder. The downsampling and upsampling rate $P = 8$ and the latent dimension $C = 16$, i.e., the bottleneck latent variable y has a shape of $(C \times \frac{T}{P} \times \frac{F}{P}) = (16 \times 128 \times 16)$. For the latent diffusion model, we refer the UNet latent diffusion model[1]. It contains 4 encoder blocks, 1 bottleneck block, and 4 decoder blocks. Each block contains 2 residual CNN layers and 1 spatial transformer layer [113]. The channel dimensions of encoder blocks are 128, 256, 384, and 640 and reversed in decoder blocks. For Hifi-GAN, we adopt its official repository [2] along with the configuration [3]. We change the

[1] https://huggingface.co/spaces/multimodalart/latentdiffusion
[2] https://github.com/jik876/hifi-gan
[3] https://github.com/jik876/hifi-gan/blob/master/config_v1.json

number of mel-bins to 128 to fit the processing of MusicLDM.

Dataset and Training Details

We used the Audiostock dataset in LAION-Audio-630K (in Chapter 4 for training MusicLDM, which provides correct text descriptions for correspoinding music tracks. Specifically, the Audiostock dataset contains 9,000 music tracks for training and 1000 tracks for testing with the total duration of 455.36 hours.

We trained all MusicLDM modules with music clips of 10.24 seconds at 16 kHz sampling rate. In both VAE and diffusion model, music clips are represented as mel-spectrograms with T=1024 frames and F=128 mel-bins. Unlike AudioLDM, the VAE module of MusicLDM utilizes a downsampling rate of P=8 and a latent dimension of C=16. The architecture and training process of MusicLDM follow those of AudioLDM [68].

For the training of VAE, we use the Adam optimizer [56] with a learning rate of 4.5×10^{-6} with a batch size of 24. We apply the mel-spectrogram loss, adversarial loss, and a Gaussian constraint loss as the training object of VAE. For the training of Hifi-GAN, we use the batch size of 96 and the AdamW optimizer with $\beta_1 = 0.8$, $\beta_2 = 0.99$ at the learning rate of 2×10^{-4}. For the training of MusicLDM, we use the batch size of 24 and the AdamW optimizer with the basic learning of 3×10^{-5}. In the forward process, we use 1000-step of a linear noise schedule from $\beta_1 = 0.0015$ to $\beta_{1000} = 0.0195$. In the sampling process, we use the DDIM [101] sampler with 200 steps. We adopt the classifier-free guidance [47] with a guidance scale $w = 2.0$. When applying the mixup strategy, we use the mixup rate $p = 0.5$. The CLAP model is trained on 24 A100 GPUs. The VAE and Hifi-GAN model are trained on 4 A6000 GPUs. Last, the latent diffusion model is trained on single NVIDIA A6000 GPU. All models are converged at the end of the training.

Table 5.2. The evaluation of generation quality among MusicLDMs and baselines. AA-Train. and TA-Train. refer to the audio-audio training scheme and the text-audio training scheme. MusicGen and MusicLDM are works in the same period.

Model	Training Data Size	AA-Train.	TA-Train.	FD_{pann} ↓	FD_{vgg} ↓	Inception Score ↑	KL Div. ↓
Riffusion [33]	—	✗	✓	68.95	10.77	1.34	5.00
MuBERT [78]	—	—	—	31.70	19.04	1.51	4.69
AudioLDM (w/. original CLAP) [68]	455 hours	✓	✗	38.92	3.08	1.67	3.65
Moûsai [97]	2500 hours	✗	✓	30.73	10.59	1.50	3.88
MusicGen* [18]	20000 hours	✗	✓	**25.19**	**2.17**	**1.82**	**3.10**
MusicLDM	455 hours	✓	✗	26.67	2.40	**1.81**	3.80
MusicLDM (Only TA-Training)		✗	✓	32.40	2.51	1.49	3.96
MusicLDM w/. mixup		✓	✗	30.15	2.84	1.51	3.74
MusicLDM w/. BAM		✓	✗	28.54	**2.26**	1.56	3.50
MusicLDM w/. BLM		✓	✗	**24.95**	2.31	1.79	**3.40**
MusicLDM w/. Text-Finetune	455 hours	✓	✓	27.81	1.75	1.76	3.60
MusicLDM w/. BAM & Text-Finetune		✓	✓	28.22	1.81	1.61	3.61
MusicLDM w/. BLM & Text-Finetune		✓	✓	**26.34**	**1.68**	**1.82**	**3.47**

Implementation of Comparison Model

For generating from Riffusion and MuBERT, we use the official API of Riffusion [4] and MuBERT [5].

5.5.3 Generation Quality

Following evaluation techniques used in past work on audio generation [68], we use frechet distance (FD), inception score (IS), and kullback-leibler (KL) divergence to evaluate the quality of generated musical audio outputs. Frechet distance evaluates the audio quality by using an audio embedding model to measure the similarity between the embedding space of generations and that of targets. In this dissertation, we use two standard audio embedding models: VGGish [99] and PANN [60]. The resulting distances we denote as FD_{vgg} and FD_{pann}, respectively. Inception score measures the diversity and the quality of the full set of audio outputs, while KL divergence is measured on individual pairs of generated and groundtruth audio samples and averaged. We use the `audioldm_eval` library [6] to evaluate all the metrics mentioned above, comparing the groundtruth audio from the Audiostock 1000-track test set with the 1000 tracks of music generated by each system based on the corresponding textual descriptions.

Table 5.2 presents the results for our models in comparison with baselines. We sent textual descriptions from the test set to the official APIs of Riffusion, MuBERT, Moûsai, and MusicGen to generate corresponding audio results. Both Riffusion and MuBERT were unable to achieve results comparable to the remaining models. The sub-optimal performance of Riffusion resulted from poor music generation quality, while MuBERT generated high-quality pieces from real sample libraries but fell short in replicating the distribution of Audiostock dataset. Moûsai and MusicGen yielded much better generation quality by leveraging advanced model architectures as well as the large scale of internal training data. We also retrained the original AudioLDM model on the Audiostock dataset but rely on the original CLAP models for condition

[4] https://huggingface.co/riffusion/riffusion-model-v1
[5] https://github.com/MubertAI/Mubert-Text-to-Music
[6] https://github.com/haoheliu/audioldm_eval

embeddings. Throughout all models, we observed that MusicLDM variants, trained on only 455 hours music tracks, are able to achieve competitive FD_{pann}, FD_{vgg}, and IS scores with only a slightly inferior on the KL divergence score to MusicGen. This underscores the efficacy of CLAP model pretrained for music, providing more suitable music embeddings as conditioning information.

We also observe the inferior results of "MusicLDM (Only TA-Training)" in comparison to audio-to-audio training variants. This suggests that a gap exists between distributions of text and audio embeddings, making it challenging to generate high-quality audio solely from text embedding.

This hypothesis is further supported by the results of combining audio-to-audio training and text-to-audio fine-tuning. We observe a significant decrease in FD_{vgg} with small changes in FD_{pann} and IS, indicating a substantial improvement in generation quality, driven by leveraging both audio and text embeddings during training.

Last, we compared MusicLDM with different mixup strategies, namely simple mixup [127], BAM, and BLM. The comparison reveals the negative impact of the simple mixup on all metrics. This degradation in generated sample quality, characterized by instrumental interference and noise, is attributed to the simple mixup's inability to guarantee musicality in the mix. Similar observations are evident in the BAM results, indicated by a drop in FD_{pann} and IS. However, the tempo and downbeat alignments of BAM (and BLM), along with the original mixup benefits, counterbalance this defect to a certain extent, enhancing the model's generalization ability and improving certain metrics. BLM, as a latent space mixing method, aligns with our hypothesis that latent space mixup yield audio closely resembling music. This technique allows us to largely bypass the potential confusion issues tied to audio mixing, thus capitalizing on mixup's ability to drive generalization and prevent copying via data augmentation. Furthermore, incorporating text-finetuning results in a comprehensive improvement of music generation quality, solidifying BLM as the most effective strategy.

Table 5.3. The objective metrics to measure the relevance and novelty (plagiarism). And the subjective listening test to evaluate the quality, relevance, and musicality.

Model	Objective Metrics			Subjective Listening Test		
	Relevance	Novelty and Plagiarism Risk				
	Text-Audio Similarity↑	SIM_{AA}@90 ↓	SIM_{AA}@95 ↓	Quality↑	Relevance↑	Musicality↑
Test Set (Ref.)	0.325	—	—	—	—	—
Retrieval Max (Ref.)	0.423	—	—	—	—	—
MuBERT [78]	0.131	0.107	0	2.02	1.50	**2.33**
MusicLDM (original)	**0.281**	0.430	0.047	1.98	2.17	2.19
MusicLDM w/. mixup	0.234	**0.391**	0.028	—	—	—
MusicLDM w/. BAM	0.266	0.402	0.027	2.04	2.21	2.01
MusicLDM w/. BLM	0.268	0.401	**0.020**	**2.13**	**2.31**	2.07

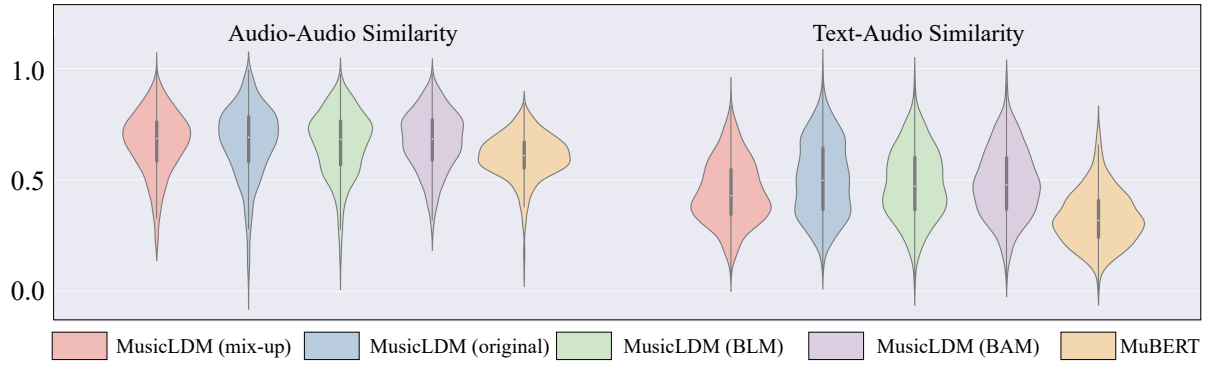


Figure 5.3. The violin plot of the audio-audio similarity, and the text-to-audio similarity.

5.5.4 Text-Audio Relevance, Novelty and Plagiarism

We proposed two metric groups, **text-audio similarity** and **nearest-neighbor audio similarity ratio** to assess text-audio relevance, novelty, and plagiarism risk in various models.

First, text-audio similarity measures the relevance between the text and the audio. It is defined as the dot product between the groundtruth text embedding $\boldsymbol{E}^t_{gd}$ from the test set and the audio embedding $\boldsymbol{E}^a$ from music generated by models, i.e., $\boldsymbol{E}^t_{gd} \cdot \boldsymbol{E}^a$. The embeddings from both text and audio are normalized in CLAP model, thus the dot product computes the cosine similarity between text and audio embeddings.

Second, we would also like to measure the extent to which models are directly copying samples from the training set. We verify this by first computing the dot products between the audio embedding of each generated music output to all audio embeddings from the Audiostock

training set and returning the maximum – i.e., the similarity of the nearest-neighbor in the training set. Then, we compute the fraction of generated outputs whose nearest-neighbors are above a threshold similarity. We refer this as the nearest-neighbor audio similarity ratio, providing SIM_{AA}@90 where the threshold is 0.9, and SIM_{AA}@95 with 0.95.

The lower this fraction, the lower the risk of plagiarism – i.e., fewer samples have very close training neighbors. In the Appendix D, we show pairs of examples with both high and low similarity scores to give further intuition for this metric.

As shown in the left and middle column of Table 5.3, we present the average text-audio similarity and nearest-neighbor audio similarity ratios for two thresholds on the 1000 tracks in the Audiostock test set and the generated music from MuBERT and different variants of MusicLDM. We also provide two reference points for text-audio similarity: "Test Set" and "Retrieval Max". Specifically, "Test Set" refers to computing the cosine similarity between the groudtruth text embedding and the groudtruth audio embedding. And "Retrieval Max" refers to first computing the cosine similarities between each text embedding from the test set to all audio embeddings from the training set, then picking the highest score as the score of this text, and taking the average over all text scores.

We can observe that the original MusicLDM without mixup achieves the highest text-audio relevance with an average score of 0.281, but also the highest (worst) nearest-neighbor audio similarity ratio. MusicLDM with the simple mixup strategy achieves the lowest SIM_{AA}@90 ratio while sacrificing a lot in the relevance of the generation. The MusicLDM with BAM and BLM achieve a balance between the audio similarity ratios and the text-to-audio similarity. In combination with the quality evaluation results in Table 5.2, we can conclude that all mixup strategies are effective as a data augmentation techniques to improve generalization of the model to generate more novel music. However simple mixup degrades the generation quality, which affects the relevance score between audio and text, and also thus makes it less similar to the tracks in the training set. BAM and BLM apply the tempo and downbeat filtering on the music pairs to mix, allowing the model to maintain superior generation quality (Table 5.2) and text-audio

relevance (Table 5.3), while still utilizing the benefit brought by the mixup technique to make the generation more novel (less plagiarism). Among the objective metrics, BLM is the best mixup strategy in terms of quality, relevance and novelty of the generated audio. This indicates mixing in the latent space is more efficient than mixing directly in audio space, perhaps because the latent embedding approach implicitly projects the mixture to the learned manifold of well-formed music.

We show the detailed distribution of these metrics over 1000 generated tracks in Figure 5.3, where, for example, audio-audio similarity denotes the individual scores used to calculate the average SIM_{AA}. We find that the original MusicLDM without mixup has more samples with high training similarity than other models, which further reflects that it is more prone to copying.

5.5.5 Subjective Listening Test

As shown in the right of Table 5.3, we conduct the subjective listening test on four models, namely MuBERT, the original MusicLDM, and that with BAM or BLM strategy, to further evaluate the actual hearing experience of the generated music. We do not include the simple mixup MusicLDM because its generation is at a low quality while we avoid confusing subjects with too many models in the same time.

The subjective listening test was conducted in an online survey format to gather feedback and insights on the text-to-music generation using MusicLDMs and MuBERT. The generation of Riffusion was not included due to its lower quality and relevance compared to the standard. The test had an estimated duration of approximately 10 minutes.

At the beginning of the test, participants were asked to provide their age range and music background as metadata. Subsequently, participants were randomly assigned six groups of generated songs. Each group consisted of four songs generated from MusicLDM, MusicLDM with BAM, MusicLDM with BLM, and MuBERT, all based on the same textual description. The order of the songs within each group was shuffled to eliminate positional bias during rating. Participants were required to rate each song based on three metrics:

- Relevance: Determine how well the song matches the given music description. Rate the song based on how closely it aligns with the provided description.
- Quality: Assess the overall quality of the music. Consider factors such as clarity, absence of noise, and general audio quality. Rate the song based on these criteria.
- Musicality: Evaluate the musical attributes of the song, including rhythm, melodies, and textures. Rate the song based on its overall musical appeal.

Each song in the subjective listening test had a duration of approximately 10 seconds and included a fade-in and fade-out to mitigate bias from the song's beginning and ending sections. The rating scale used for evaluating the songs was designed such that a higher score indicates better quality. Participants were asked to rate each song based on the provided metrics, taking into account the song's overall quality, relevance to the given text, and personal preference on its musicality.

We invite 15 subjects to listen to 6 groups of the generations randomly selected from the test set. Each of group contains four generations from four models and the corresponding text descriptions.

From Table 5.3, we observe that the samples of MusicLDM with BAM or BLM mixup strategy achieve a better relevance and quality than those of MuBERT and the original MusicLDM, this strengths our above analysis. The MuBERT samples achieve the best Musicality, because its generation is combined from the real music samples. The generation samples of MusicLDM with BAM or BLM mixup strategies also achieve better relevance and quality than those of the original MusicLDM, with an inferior on musicality but still maintaining above an acceptable threshold. Combined with the objective metrics, beat-synchronous latent mixup stands to be the most effectiveness method for enhancing the text-to-music generation in terms of quality, text-music relevance and novelty (i.e., reducing the risk of plagiarism).

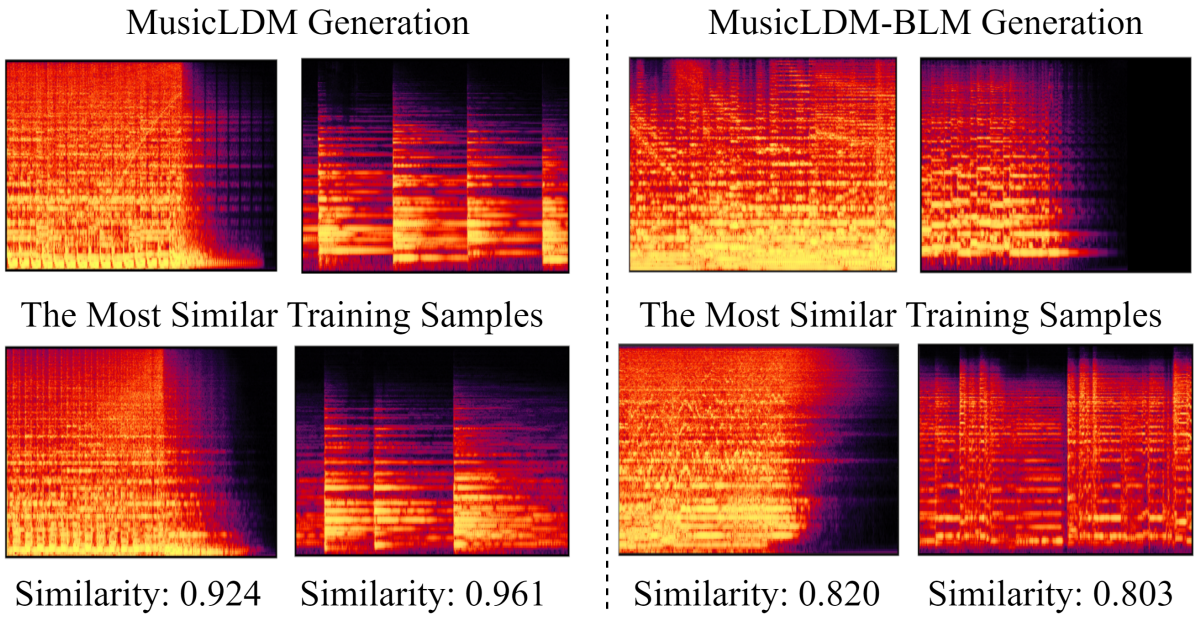


Figure 5.4. The generation examples by two MusicLDM models and their most similar tracks in the Audiostock training set.

Plagiarism Verification

As mentioned in section 5.5.4, we introduced the computation of the nearest-neighbor audio similarity ratio by comparing the cosine similarity between generated music and music tracks in the training set of Audiostock.

In this section, we provide visualizations of the similarity between the generated music and the training music using spectrograms, showcasing if the proposed BLM-version of MusicLDM can mitigate the plagirism from the trainning data.

As shown in Figure 5.4, we present two groups of music pairs from MusicLDM and MusicLDM-BLM models. To achieve this, we divided the training music tracks into 10-second segments and determined the most similar segment to the generated music track.

For instances with high similarity, the cosine similarity of CLAP audio embeddings reveals highly similar structural patterns, indicating a close resemblance in the music arrangements. Conversely, low CLAP cosine similarity indicates significant differences between the spectrograms of the generated music and the training music. This demonstrates the effectiveness

of CLAP embeddings in assessing the similarity between music tracks and serving as a means to detect novelty and potential instances of plagiarism in the generated samples. Add we also observe how BLM helps MusicLDM prevent the plagiarism risk and generate novel music samples with a lower audio similarity to the groundtruth tracks than the original model.

5.6 Limitations and Impacts

As the generation task, there exists several limitations and impacts of our study.

Firstly, MusicLDM is trained on the music data in a sampling rate of 16 kHz, while most standard music productions are 44.1 kHz. This constraint, tied to the Hifi-GAN vocoder's subpar performance at high sampling rates, impedes practical text-to-music application and necessitates further improvements.

Secondly, resource constraints such as limited real text-music data and GPU processing power prevent us from scaling up MusicLDM's training. We are unable to determine if mix-up strategies could yield similar trends as observed with the Audiostock dataset. This issue exists in the image generation task as well.

Lastly, while we recognize beat information as crucial for music alignment, there is scope for exploring other synchronization techniques like key signature and instrument alignment. We also intend to investigate the application of different audio space filters to select suitable music pairs for mixing.

The development and implementation of MusicLDM, or generally a text-to-music generation model offers potential benefits and also raises concerns that must be addressed responsibly.

Positive Impacts

- Promoting Creativity: This model can serve as a tool to augment human creativity. Artists, composers, and music amateurs can use it to transfer their textual ideas into music, broadening the realm of artistic exploration and making music creation more accessible and convenient.

- Cultural Preservation and Evolution: The model provides a unique platform to archive, digitize, and even evolve cultural musical expressions. Textual descriptions of traditional and folk music can be transformed into the actual music, thereby helping to preserve heritage while simultaneously allowing for creative adaptations. Literature, such as poetry, can be interpreted as music to explore more relations between different types of cultural expression forms.

- Education and Research: In academia, this model can be used as a pedagogical tool in music education. It can aid in understanding the complex relationship between music and linguistic structures, enriching interdisciplinary research in musicology, linguistics, and artificial intelligence.

- Entertainment Industry Innovation: The entertainment industry could use this model to generate soundtracks for movies, games, and other media based on scripts. This could potentially revolutionize the way music is produced for media, reducing time and costs.

Negative Impacts

- Artistic Job Displacement: While this model can augment human creativity, it may also lead to job losses in the music industry if widely adopted for composing and production. The model could potentially replace human composers in certain contexts, particularly in industries such as film and gaming that require a significant amount of background music.

- Copyright Issues: In this dissertation, one of the targets is to mitigate the copyright issues and plagiarism. The generated music could unintentionally resemble existing works, raising complex copyright infringement issues. It is crucial to implement measures to ensure that the model does not violate intellectual property rights.

- Ethical Misuse: The model could be misused to create music promoting hate speech, misinformation, or other harmful content if the input text has such characteristics. Thus, it is essential to develop safeguards to mitigate the risk of misuse.

- Cultural Appropriation and Homogenization: While the model can help preserve music, there is a risk of homogenizing unique cultural musical styles or misappropriating them without proper credit or context.

The design and application of this model should be carried out responsibly, considering the potential ethical, social, and economic consequences. Balancing its many benefits with its potential downsides will require the collective effort of developers, users, policy makers, and society at large.

5.7 Conclusion

In this chapter, we introduce MusicLDM, a text-to-music generation model that incorporates CLAP, VAE, Hifi-GAN, and latent diffusion models. We enhance MusicLDM by proposing two efficient mixup strategies: beat-synchronous audio mixup (BAM) and beat-synchronous latent mixup (BLM), integrated into its training process. We conduct comprehensive evaluations on different variants of MusicLDM using objective and subjective metrics, assessing quality, text-music relevance, and novelty. The experimental results demonstrate the effectiveness of BLM as a standout mixup strategy for text-to-music generation. And we prove that the proposed audio representation model (CLAP along with HTS-AT) is able to contribute to this novel format of music generation task by transferring inspiration and understanding from the language modality.

This chapter contains some materials (texts, tables, and figures) from a published conference paper: Ke Chen*, Yusong Wu*, Haohe Liu*, Marianna Nezhurina, Taylor Berg-Kirkpatrick, Shlomo Dubnov, *MusicLDM: Enhancing Novelty in Text-To-Music Generation using Beat-Synchronous Mixup Strategies*, in proceedings of International Conference on Acoustics, Speech and Signal Processing, ICASSP 2024. The dissertation author was the first author of this publication.

Chapter 6

Conclusion and Future Work

In this dissertation,we trace the evolution of an audio representation model from its inception to its adaptation in different audio downstream tasks. We begin from an initial design of an innovative audio transformer as the cornerstone, HTS-AT, that employs imperative designs to capture semantic and acoustic information of audio data. We demonstrate its efficacy in capturing both semantic and acoustic information from audio data. We systematically showcase the versatility of HTS-AT across advanced audio applications, ranging from audio event classification to audio source separation. Moreover, to facilitate multimodal learning, we introduce CLAP, a contrastive language-audio pretraining model that integrates HTS-AT with language understanding. Leveraging these advancements, we present MusicLDM, a latent diffusion model for text-to-music generation, thereby achieving the goal of content creation.

Through extensive experiments and application studies, we validate the adaptability and superior performance of HTS-AT across diverse audio tasks, including semantic extraction, content extraction, multimodal learning, and content generation. Our findings underscore the potential of building a robust audio representation model as a cornerstone for various audio applications, offering promising prospects in the field of artificial intelligence.

In the future, we wish to explore more applications of HTS-AT in the field of audio processing. The applications shown in this dissertation are still in the slots of the general audio tasks, while more fine-grained applications of music and general sound are awaiting:

- Music Recommendation: by leveraging HTS-AT as a Siamese network in capturing both audio content and user preference in the music recommendation scenario, whose method is partially verified in [15].

- Singing Melody Extraction and Music Transcription: by leveraging HTS-AT to extract the fundamental frequencies of singing melodies, or the onset, duration, and velocity information from the polyphonic music tracks, whose method is partially verified in [17].

- Audio Captioning: by combining CLAP with language models to generate the descriptions of audio tracks as a reversed target of text-to-audio, whose method is partially verified in [102].

- Fine-grained Control on Music Generation: by incorporating more controls, such as chord, melody, lyric as the music control signals, and genre, impression, usage as the general control signals. Realizing such target requires the developments in both contrastive learning and generative model.

In conclusion, the field of audio and music processing holds vast untapped potential, and we are committed to addressing its challenges and maximizing its possibilities through our ongoing and future endeavors. With HTS-AT, we are merely at the dawn of a new era.

Bibliography

[1] Introducing chatgpt. https://openai.com/blog/chatgpt.

[2] Dcase challenge task 4: Sound event detection and separation in domestic environments. http://dcase.community/challenge2021, 2021.

[3] Abien Fred Agarap. Deep learning using rectified linear units (relu). *CoRR*, abs/1803.08375, 2018.

[4] Andrea Agostinelli, Timo I Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, et al. MusicLM: Generating music from text. *CoRR*, abs/2301.11325, 2023.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *Proceedings of International Conference on Learning Representations*, 2015.

[6] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[7] Axel Berg, Mark O'Connor, and Miguel Tairum Cruz. Keyword transformer: A self-attention model for keyword spotting. In *Proceedings of Interspeech*, 2021.

[8] Rachel M. Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. Medleydb: A multitrack dataset for annotation-intensive MIR research. In *Proceedings of International Society for Music Information Retrieval Conference*, 2014.

[9] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ B. Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar

Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, and Rohith Kuditipudi. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021.

[10] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. *Deep learning techniques for music generation*. Springer, 2020.

[11] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Proceedings Conference on Neural Information Processing Systems*, 2020.

[12] Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *Proceedings of USENIX Security Symposium*, 2023.

[13] Pritish Chandna, Marius Miron, Jordi Janer, and Emilia Gómez. Monoaural audio source separation using deep convolutional neural networks. In *Proceedings of International Conference on Latent Variable Analysis and Signal Separation*, 2017.

[14] Honglie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman. Vggsound: A large-scale audio-visual dataset. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2020.

[15] Ke Chen, Beici Liang, Xiaoshuan Ma, and Minwei Gu. Learning audio embeddings with user listening data for content-based music recommendation. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2021.

[16] Ke Chen, Cheng-i Wang, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Music sketchnet: Controllable music generation via factorized representations of pitch and rhythm. In *Proceedings of International Society for Music Information Retrieval Conference*, 2020.

[17] Ke Chen, Shuai Yu, Cheng-i Wang, Wei Li, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Tonet: Tone-octave network for singing melody extraction from polyphonic music. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2022.

[18] Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation. In *Proceedings Conference on Neural Information Processing Systems*, 2023.

[19] Yimian Dai, Fabian Gieseke, Stefan Oehmcke, Yiquan Wu, and Kobus Barnard. Attentional feature fusion. In *Proceedings of IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021.

[20] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[21] Soham Deshmukh, Benjamin Elizalde, and Huaming Wang. Audio retrieval with wavtext5k and CLAP training. In *Proceedings of Interspeech*, 2023.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

[23] Hao-Wen Dong, Ke Chen, Julian J. McAuley, and Taylor Berg-Kirkpatrick. Muspy: A toolkit for symbolic music generation. In *Proceedings of International Society for Music Information Retrieval Conference*, 2020.

[24] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of International Conference on Learning Representations*, 2021.

[25] Konstantinos Drossos, Samuel Lipping, and Tuomas Virtanen. Clotho: an audio captioning dataset. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2020.

[26] Benjamin Elizalde, Soham Deshmukh, Mahmoud Al Ismail, and Huaming Wang. CLAP: learning audio concepts from natural language supervision. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2023.

[27] Rebecca Fiebrink and Baptiste Caramiaux. The machine learning algorithm as creative musical tool. *CoRR*, abs/1611.00379, 2016.

[28] Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font, and Xavier Serra. FSD50K: an open dataset of human-labeled sound events. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2022.

[29] Eduardo Fonseca, Manoj Plakal, Frederic Font, Daniel P. W. Ellis, Xavier Favory, Jordi Pons, and Xavier Serra. General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline. *CoRR*, abs/1807.09902, 2018.

[30] Eduardo Fonseca, Manoj Plakal, Frederic Font, Daniel PW Ellis, Xavier Favory, Jordi Pons, and Xavier Serra. General-purpose tagging of Freesound audio with Audioset labels: Task description, dataset, and baseline. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2018.

[31] Frederic Font Corbera, Gerard Roma Trepat, and Xavier Serra. Freesound technical demo. In *Proceedings of Multimedia*, 2013.

[32] Logan Ford, Hao Tang, François Grondin, and James R. Glass. A deep residual network for large-scale acoustic scene analysis. In *Proceedings of Interspeech*, 2019.

[33] Seth Forsgren and Hayk Martiros. Riffusion - Stable diffusion for real-time music generation. 2022.

[34] Wei Gao, Fang Wan, Xingjia Pan, Zhiliang Peng, Qi Tian, Zhenjun Han, Bolei Zhou, and Qixiang Ye. Ts-cam: Token semantic coupled attention map for weakly supervised object localization. In *Proceedings of International Conference on Computer Vision*, 2021.

[35] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2017.

[36] Karan Goel, Albert Gu, Chris Donahue, and Christopher Ré. It's raw! audio generation with state-space models. In *Proceedings of International Conference on Machine Learning*, 2022.

[37] Yuan Gong, Yu-An Chung, and James Glass. Ast: Audio spectrogram transformer. In *Proceedings of Interspeech*, 2021.

[38] Yuan Gong, Yu-An Chung, and James Glass. Psla: Improving audio tagging with pretraining, sampling, labeling, and aggregation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2021.

[39] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *CoRR*, abs/2312.00752, 2023.

[40] Andrey Guzhov, Federico Raue, Jörn Hees, and Andreas Dengel. Audioclip: Extending clip to image, text and audio. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2022.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[42] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: decoding-enhanced bert with disentangled attention. In *Proceedings of International Conference on Learning Representations*, 2021.

[43] Romain Hennequin, Anis Khlif, Felix Voituret, and Manuel Moussallam. Spleeter: a fast and efficient music source separation tool with pre-trained models. *Journal of Open Source Software*, 2020.

[44] Perfecto Herrera, Geoffroy Peeters, and Shlomo Dubnov. Automatic classification of musical instrument sounds. *Journal of New Music Research*, 2010.

[45] Shawn Hershey, Daniel P. W. Ellis, Eduardo Fonseca, Aren Jansen, Caroline Liu, R. Channing Moore, and Manoj Plakal. The benefit of temporally-strong labels in audio event classification. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2021.

[46] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings Conference on Neural Information Processing Systems*, 2020.

[47] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *CoRR*, abs/2207.12598, 2022.

[48] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[49] Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. Joint optimization of masks and deep recurrent neural networks for monaural source separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2015.

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

[51] Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. *CoRR*, abs/2301.12661, 2023.

[52] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of International Conference on Machine Learning*, 2015.

[53] Andreas Jansson, Eric Humphrey, Nicola Montecchio, Rachel Bittner, Aparna Kumar, and Tillman Weyde. Singing voice separation with deep U-Net convolutional networks. In *Proceedings of International Society for Music Information Retrieval Conference*, 2017.

[54] Ilya Kavalerov, Scott Wisdom, Hakan Erdogan, Brian Patton, Kevin W. Wilson, Jonathan Le Roux, and John R. Hershey. Universal sound separation. In *Proceedings of Workshop on Applications of Signal Processing to Audio and Acoustics*, 2019.

[55] Chris Dongjoo Kim, Byeongchang Kim, Hyunmin Lee, and Gunhee Kim. Audiocaps: Generating captions for audios in the wild. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

[56] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations*, 2015.

[57] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *Proceedings of International Conference on Learning Representations*, 2013.

[58] A. Sophia Koepke, Andreea-Maria Oncescu, João F. Henriques, Zeynep Akata, and Samuel Albanie. Audio retrieval with natural language queries: A benchmark study. *CoRR*, abs/2112.09418, 2021.

[59] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. In *Proceedings Conference on Neural Information Processing Systems*, 2020.

[60] Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D. Plumbley. Panns: Large-scale pretrained audio neural networks for audio pattern recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2020.

[61] Qiuqiang Kong, Yuxuan Wang, Xuchen Song, Yin Cao, Wenwu Wang, and Mark D. Plumbley. Source separation with weakly labelled data: an approach to computational auditory scene analysis. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2020.

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

[63] Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. AudioGen: Textually guided audio generation. In *Proceedings of International Conference on Learning Representations*, 2022.

[64] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Handwritten digit recognition with a back-propagation network. In *Proceedings Conference on Neural Information Processing Systems*, 1989.

[65] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Proceedings Conference on Neural Information Processing Systems*, 2000.

[66] Jie Hwan Lee, Hyeong-Seok Choi, and Kyogu Lee. Audio query-based music source separation. In *Proceedings of International Society for Music Information Retrieval Conference*, 2019.

[67] Liwei Lin, Gus Xia, Qiuqiang Kong, and Junyan Jiang. A unified model for zero-shot music source separation, transcription and synthesis. In *Proceedings of International Society for Music Information Retrieval Conference*, 2021.

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

[69] Jen-Yu Liu and Yi-Hsuan Yang. Denoising auto-encoder with recurrent skip connections and residual regression for music source separation. In *Proceedings of International Conference on Machine Learning and Applications*, 2018.

[70] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[71] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of International Conference on Computer Vision*, 2021.

[72] Antoine Liutkus, Fabian-Robert Stöter, Zafar Rafii, Daichi Kitamura, Bertrand Rivet, Nobutaka Ito, Nobutaka Ono, and Julie Fontecave. The 2016 signal separation evaluation campaign. In *Proceedings of International Conference on Latent Variable Analysis and Signal Separation*, 2017.

[73] Yi Luo and Nima Mesgarani. Tasnet: Time-domain audio separation network for real-time, single-channel speech separation. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2018.

[74] Yi Luo and Nima Mesgarani. Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2019.

[75] Yi Luo and Jianwei Yu. Music source separation with band-split rnn. *CoRR*, abs/2209.15174, 2022.

[76] Xinhao Mei, Xubo Liu, Jianyuan Sun, Mark D. Plumbley, and Wenwu Wang. On metric learning for audio-text cross-modal retrieval. In *Proceedings of Interspeech*, 2022.

[77] Annamaria Mesaros, Toni Heittola, Tuomas Virtanen, and Mark D. Plumbley. Sound event detection: A tutorial. *IEEE Signal Processing Letters*, 2021.

[78] MubertAI. Mubert: A simple notebook demonstrating prompt-based music generation.

[79] Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. Attention bottlenecks for multimodal fusion. In *Proceedings Conference on Neural Information Processing Systems*, 2021.

[80] Arun Narayanan and DeLiang Wang. Ideal ratio mask estimation using deep neural networks for robust speech recognition. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2013.

[81] Andreea-Maria Oncescu, A. Sophia Koepke, João F. Henriques, Zeynep Akata, and Samuel Albanie. Audio retrieval with natural language queries. In *Proceedings of Interspeech*, 2021.

[82] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. Specaugment: A simple data augmentation method for automatic speech recognition. In *Proceedings of Interspeech*, 2019.

[83] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2018.

[84] Karol J. Piczak. ESC: dataset for environmental sound classification. In *Proceedings of Multimedia*, 2015.

[85] 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. Learning transferable visual models from natural language supervision. In *Proceedings of International Conference on Machine Learning*, 2021.

[86] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2020.

[87] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. The MUSDB18 corpus for music separation. https://sigsep.github.io/datasets/musdb.html, 2017.

[88] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *CoRR*, abs/2204.06125, 2022.

[89] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech: Fast, robust and controllable text to speech. In *Proceedings Conference on Neural Information Processing Systems*, 2019.

[90] Gerard Roma, Owen Green, and Pierre Alexandre Tremblay. Improving single-network single-channel separation of musical audio with convolutional layers. In *Proceedings of International COnference on Latent Variable Analysis and Signal Separation*, 2018.

[91] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

[92] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of Medical Image Computing and Computer-Assisted Intervention*, 2015.

[93] Simon Rouard, Francisco Massa, and Alexandre Défossez. Hybrid transformers for music source separation. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2023.

[94] David E. Rumelhart and James L. McClelland. Learning internal representations by error propagation. In *Proceedings of Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, 1987.

[95] Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. A dataset and taxonomy for urban sound research. In *Proceedings of Multimedia*, 2014.

[96] David Samuel, Aditya Ganeshan, and Jason Naradowsky. Meta-learning extractors for music source separation. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2020.

[97] Flavio Schneider, Zhijing Jin, and Bernhard Schölkopf. Moûsai: Text-to-music generation with long-context latent diffusion. *CoRR*, abs/2301.11757, 2023.

[98] Romain Serizel, Nicolas Turpault, Ankit Parag Shah, and Justin Salamon. Sound event detection in synthetic domestic environments. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2020.

[99] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of International Conference on Learning Representations*, 2015.

[100] Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Diffusion art or digital forgery? investigating data replication in diffusion models. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2023.

[101] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *Proceedings of International Conference on Learning Representations*, 2021.

[102] Nikita Srivatsan, Ke Chen, Shlomo Dubnov, and Taylor Berg-Kirkpatrick. Retrieval guided music captioning via multimodal prefixes. In *Proceedings of International Joint Conferences on Artificial Intelligence*, 2024.

[103] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation. In *Proceedings of International Society for Music Information Retrieval Conference*, 2018.

[104] Fabian-Robert Stöter, Stefan Uhlich, Antoine Liutkus, and Yuki Mitsufuji. Open-unmix - A reference implementation for music source separation. *Journal of Open Source Software*, 2019.

[105] Naoya Takahashi, Nabarun Goswami, and Yuki Mitsufuji. MMDenseLSTM: An efficient combination of convolutional and recurrent neural networks for audio source separation. In *Proceedings of IEEE International Workshop on Acoustic Signal Enhancement (IWAENC)*, 2018.

[106] Naoya Takahashi and Yuki Mitsufuji. D3net: Densely connected multidilated densenet for music source separation. *CoRR*, abs/2010.01733, 2020.

[107] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of International Conference on Machine Learning*, 2019.

[108] Xu Tan, Jiawei Chen, Haohe Liu, Jian Cong, Chen Zhang, Yanqing Liu, Xi Wang, Yichong Leng, Yuanhao Yi, Lei He, Sheng Zhao, Tao Qin, Frank K. Soong, and Tie-Yan Liu. *CoRR*, abs/2205.04421, 2022.

[109] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proceedings of International Conference on Machine Learning*, 2021.

[110] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2002.

[111] Efthymios Tzinis, Zhepei Wang, and Paris Smaragdis. Sudo RM -rf: Efficient networks for universal audio source separation. In *Proceedings of International Workshop on Machine Learning for Signal Processing*, 2020.

[112] Stefan Uhlich, Marcello Porcu, Franck Giron, Michael Enenkl, Thomas Kemp, Naoya Takahashi, and Yuki Mitsufuji. Improving music source separation based on deep neural networks through data augmentation and network blending. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2017.

[113] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings Conference on Neural Information Processing Systems*, 2017.

[114] C. Veaux, J. Yamagishi, and S. King. The Voice Bank Corpus: Design, collection and data analysis of a large regional accent speech database. In *Proceedings of International Conference Oriental COCOSDA with Conference on Asian Spoken Language Research and Evaluation*, 2013.

[115] Sergey Verbitskiy and Viacheslav Vyshegorodtsev. Eranns: Efficient residual audio neural networks for audio pattern recognition. *CoRR*, abs/2106.01621, 2021.

[116] Roman Vygon and Nikolay Mikhaylovskiy. Learning efficient representations for keyword spotting with triplet loss. In *Proceedings of International Conference on Speech and Computer*, 2021.

[117] Wei Wang, Vincent W. Zheng, Han Yu, and Chunyan Miao. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology*, 2019.

[118] Yun Wang, Juncheng Li, and Florian Metze. A comparison of five multiple instance learning pooling functions for sound event detection with weak labeling. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2019.

[119] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *CoRR*, abs/1804.03209, 2018.

[120] Felix Weninger, John R. Hershey, Jonathan Le Roux, and Björn W. Schuller. Discriminatively trained recurrent neural networks for single-channel speech separation. In *Proceedings of Global Conference on Signal and Information Processing*, 2014.

[121] Donald S Williamson, Yuxuan Wang, and DeLiang Wang. Complex ratio masking for monaural speech separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2015.

[122] Scott Wisdom, Hakan Erdogan, Daniel P. W. Ellis, Romain Serizel, Nicolas Turpault, Eduardo Fonseca, Justin Salamon, Prem Seetharaman, and John R. Hershey. What's all the fuss about free universal sound separation data? In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2021.

[123] Ho-Hsiang Wu, Prem Seetharaman, Kundan Kumar, and Juan Pablo Bello. Wav2clip: Learning robust audio representations from clip. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2022.

[124] Yusong Wu, Tianyu Zhang, and Ke Chen. Text-to-audio retrieval via large-scale contrastive learning. *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, 2022.

[125] Yong Xu, Jun Du, Li-Rong Dai, and Chin-Hui Lee. A regression approach to speech enhancement based on deep neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2014.

[126] Dongchao Yang, Jianwei Yu, Helin Wang, Wen Wang, Chao Weng, Yuexian Zou, and Dong Yu. DiffSound: Discrete diffusion model for text-to-sound generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[127] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proceedings of International Conference on Learning Representations*, 2018.

[128] Jingwei Zhao, Gus Xia, and Ye Wang. Beat transformer: Demixed beat and downbeat tracking with dilated self-attention. In *Proceedings of International Society for Music Information Retrieval Conference*, 2022.