

**From the holdings of the Department of Special Collections , Princeton University Library,
Princeton, NJ**

These documents are either in the public domain, or can only be used for educational and research purposes (Fair use) as per U.S. Copyright law (text below). By accessing this file, all users agree that their use falls in the public domain or within fair use as defined by the copyright law. We do not charge any permission or usage fees for the publication of images of material in our collections, including those provided by Princeton University Library, via our website, catalog, or directly from staff. For more information regarding copyright, credit, and citation please review the information on the Copyright, Credit and Citation Guidelines page of our website.

<https://library.princeton.edu/special-collections/policies/copyright-credit-and-citation-guidelines>. If you have any questions, including how to order higher-quality images from the collection, contact Special Collections Public Services via our Ask Us! Form. <https://library.princeton.edu/special-collections/ask-us>

Master's Theses, 2022-2026

Computer Science, 2023

2024

AudioSetMix: Enhancing Audio-Language Datasets with LLM-Assisted Augmentations

Xu, David

Princeton, NJ : Princeton University

<http://arks.princeton.edu/ark:/88435/dsp01d217qs90z>

U.S. Copyright law: The copyright law of the United States (Title 17, §108, United States Code) governs the making of photocopies or other reproductions of copyrighted material. Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be used for any purpose other than private study, scholarship, or research. If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of fair use that user may be liable for copyright infringement. This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve a violation of copyright law. <https://www.copyright.gov/title17/title17.pdf>

AUDIOSETMIX: ENHANCING
AUDIO-LANGUAGE DATASETS WITH
LLM-ASSISTED AUGMENTATIONS

DAVID XU

A MASTER'S THESIS
PRESENTED TO THE FACULTY
OF PRINCETON UNIVERSITY
IN CANDIDACY FOR THE DEGREE
OF MASTER OF SCIENCE

RECOMMENDED FOR ACCEPTANCE BY
THE DEPARTMENT OF COMPUTER SCIENCE

ADVISOR: KARTHIK NARASIMHAN

MAY 2024



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License (<https://creativecommons.org/licenses/by-nc-sa/4.0/>)

Abstract

Multi-modal learning in the audio-language domain has seen significant advancements in recent years. However, audio-language learning faces challenges due to limited and lower-quality data compared to image-language tasks. Existing audio-language datasets are notably smaller, and manual labeling is hindered by the need to listen to entire audio clips for accurate labeling.

Our method systematically generates audio-caption pairs by augmenting audio clips with natural language labels and corresponding audio signal processing operations. Leveraging a Large Language Model, we generate descriptions of augmented audio clips with a prompt template. This scalable method produces AudioSetMix, a high-quality training dataset for text-and-audio related models.

Integration of our dataset improves models performance on benchmarks by providing diversified and better-aligned examples. Notably, our dataset addresses the absence of modifiers (adjectives and adverbs) in existing datasets. By enabling models to learn these concepts, and generating hard negative examples during training, we achieve state-of-the-art performance on multiple benchmarks.

Acknowledgements

Many thanks to Dr. David Harwath for his suggestions and advice that inspired and greatly improved the ideas introduced in this work. I am extremely grateful for his constant guidance and support over the years.

In addition, I would like to thank my thesis advisor Dr. Karthik Narasimhan and my second reader Dr. Adam Finkelstein for their advice on my thesis.

Chapter 1

Introduction

In recent years, there has been a large amount of work in expanding comprehension of audio content by augmenting audio signal with information from another modality such as natural language. Tasks such as text-to-audio generation (TTA) [24] [25], text-guided audio editing [35], automatic audio-captioning [28] [11], and text-to-audio retrieval [29] [17] have been proposed as objectives that improve model understanding of audio signal [30].

A closely related field to audio-language learning is the vision-language learning, which comprises of tasks such as visual question-answering [2] and text-to-image generation [32] [33]. However, unlike audio-language, the vision-language learning benefits from the existence of large-scale, high quality datasets such as MSCOCO [23]. This makes it possible to pretrain large, powerful models to learn vision-language multimodal embeddings, which can then be applied for downstream tasks [22] [21] [33]. On the other hand, a significant challenge of audio-language learning is the lack of a large dataset consisting of high-quality audio-caption pairs. We note the distinction between captioned natural sound and captioned speech, the latter of which is more readily available. A commonly used dataset in audio-language is the AudioSet dataset [10], a collection of 2M YouTube videos of natural sounds with multi-label

annotations. However, it is quite clear that labels alone are not sufficient to replace high-quality audio captions. Furthermore, datasets such as AudioCaps [16] that provide human-generated text captions for audio are generally not large enough to train a deep neural network, and only suitable for fine-tuning [30].

In this work, we introduce AudioSetMix, an audio-caption dataset generated through the application of audio transformations to clips from AudioSet. In addition, we use prompt engineering and large language model (LLM) to ensure that the transformed audio and its caption are aligned. Our dataset supports speed, pitch, volume, and duration augmentations for individual clips, as well as mixing and concatenation augmentations to combine multiple clips into one. Besides having high quality text descriptions for supervised audio-language tasks, our data augmentation scheme also supports a dataset for studying text-guided audio manipulations, as we have access to both the original and edited audio.

To demonstrate the effectiveness of our dataset, we train a state-of-the-art model from the 2022/2023 DCASE Challenge on Language-Based Audio Retrieval (Task 6B) [37] using AudioSetMix. We demonstrate that our model exhibits an improved understanding of common audio event modifiers such as volume or duration, as well as a better retrieval score overall compared to baseline models. Finally, we introduce a hard negative mining technique for the AudioSetMix data which further boosts model performance.

Chapter 2

Background and Related Work

2.1 Audio Signal Processing

In this section, we give a brief overview of the feature extraction processes for audio signals. These techniques from audio signal processing are important to enable machine learning by converting audio signals to neural-networks-friendly input formats.

Because sound waves are continuous and cannot be represented in a digital format, it is necessary to sample from the sound signal.

More formally, let $x(t)$ be a continuous time signal. We define the sample $x[n]$ as:

$$x[n] = x(nT_s), n \in \mathbb{Z},$$

where T_s is the sampling period. For audio processing, T_s is typically set such that the sampling rate is 16000 Hz [40].

Because audio signal are more conveniently processed in the frequency domain, we apply the Discrete Fourier Transform (DFT). For N samples the DFT is defined as:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-i \frac{2\pi k}{N} n}.$$

Audio signals change over time and are only locally periodic. To address this issue, we first apply a window function $w[n]$ to the audio signal, and compute the DFT for each windowed portion of the signal. This is known as the Short Time Fourier Transform (STFT), which is defined as follows:

$$X[m, k] = \sum_{n=m}^{m+N-1} w[n-m]x[n]e^{-i\frac{2\pi k}{N}n},$$

where N is the window length and m is the window stride. A common window configuration is $N=25\text{ms}$ and $m=10\text{ms}$.

Next, we compute the power spectrum:

$$X_{pow}[m, k] = |X[m, k]|^2.$$

Finally, we apply a set of 64 mel-scale filters to the power spectrum and compute the log of the result:

$$X_{logmel} = \log(mel \times X_{pow}),$$

where mel is a matrix containing the mel-scale filters.

This gives us a 64-dimension logmel spectrogram, a commonly used input for models that process audio. For more details about the established work in audio signal processing, please refer to [40].

2.2 Dataset Improvements for Audio-Language Learning

In light of the dataset shortcomings for audio-language tasks, several workarounds for the data shortage have been proposed to train large models for audio-language tasks

such as TTA. These approaches can be broadly classified into three categories.

The first approach is to use predefined text templates to form rough approximations of descriptions. The simplest text templates are proposed by [20], in which audio labels are concatenated together in a random order. [20] also proposes a data augmentation for the AudioSet dataset by mixing multiple audio samples together. The corresponding text caption is simply the concatenated labels of each sample. To allow for more complex relationships to be expressed in the captions, [39] randomly inserts $\langle MASK \rangle$ tokens between labels in the hope that the model will learn to substitute in relational words. [14] improves this approach by applying a set of common audio augmentations to AudioSet data, and associates each augmentation with a caption template.

The second approach trains models in a self-supervised manner by using a pre-trained CLAP model to embed audio and text to a shared latent space [24] [1]. During training, when no captions are available, the CLAP model is used to perform a form of zero-shot audio captioning by substituting the text embedding with the audio embedding. This approach has been applied to both TTA and to music generation with good results. In particular, this approach has been adopted by AudioLDM [24], which achieves state-of-the-art performance on both TTA task and other audio editing tasks such as style transfer and audio inpainting.

The third and most recent approach to overcome the description scarcity issue utilizes recent LLM models such as ChatGPT [5] from OpenAI to generate text descriptions. This approach provides a few benefits. Firstly, text descriptions from LLM are much more varied when compared to text templates in terms of sentence structure and word choice. Our observations also indicate that LLM descriptions are also fairly realistic when compared to human descriptions. Secondly, the quality of captions from LLM can be readily improved using prompt engineering techniques such as few-shot or chain-of-thought prompting. Furthermore, using LLM allows us

to increase the complexity of data augmentations without requiring an intractably large number of human created templates.

The first work to adopt the third approach is WavCaps [30], a large audio-captioned dataset using ChatGPT. WavCaps combines several weakly-labelled datasets by prompting ChatGPT to create a natural language caption, given a list of sound event labels. However, WavCaps does not incorporate any form of data augmentation in its captions, thus reducing the diversity and complexity of captions. The lack of data augmentation also hinders the training of text-guided audio editing models, as editing keywords do not appear often in AudioSet labels. Finally, WavCaps does not explore the idea of generating hard negative examples, such as including two audio-caption pairs that differ by only audio event modifiers. In the next section, we introduce AudioSetMix, an audio-caption dataset which addresses these concerns.

Chapter 3

AudioSetMix Dataset

In this section, we describe the process of creating the AudioSetMix dataset. Firstly, we introduce the data source and its characteristics. Secondly we describe a four-stage pipeline for generating weak captions, including data preprocessing, audio clip augmentation, LLM-based caption generation, and postprocessing. Finally, we provide an analysis of AudioSetMix and compare it with existing audio-language datasets.

3.1 TS-AudioSet

The original AudioSet dataset consists of approximately 2 millions 10 second audio clips with human annotations. However, AudioSet labels are considered weak as an audio clip in its entirety may not correspond to its label due to interference such as

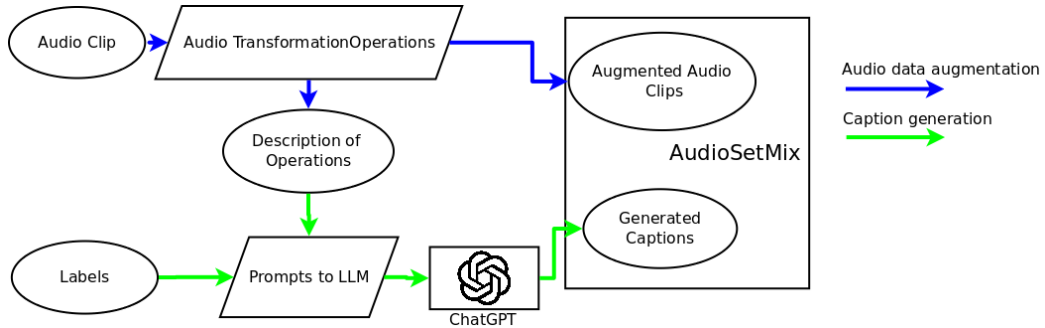


Figure 3.1: The overall pipeline to generate the AudioSetMix.

background noise [12]. This imperfection is undesirable when training text-to-audio models as they may learn to associate labels with silence or white noise that dominate audio clips. For this reason, previous works [6] restrict themselves to particular sound classes such as drumming. To resolve this issue, [12] released Temporally-Strong AudioSet (TS-AudioSet), in which each audio clip has precise start and end timestamps and are labeled by humans. We use the clips and labels from TS-AudioSet as the basis for forming AudioSetMix.

3.2 Data Generation

We propose the following pipeline for generating clip captions using a LLM. We will first discuss our audio clip augmentations, then move into the details of generating audio-aligned captions that are specific to each augmentation method applied to the audio clips. Figure 3.1 illustrates the full data processing pipeline.

3.2.1 Audio Data Preprocessing

We first apply preprocessing to remove noisy or undesirable data. This largely consists of filtering operations to audio clips and captions, referred to as duration-based filtering and class-based filtering respectively. Due to the noisiness of the raw audio-caption pairs, these filtering operations are needed to ensure the cleanliness of the data as the starting point for the subsequent enhancement and augmentation. For duration-based filtering, we remove clips with a duration of less than two seconds. This is because shorter clips tend to lack meaningful content and require long amounts of padding during training [30]. Furthermore, short clips may be even further reduced if duration augmentations are applied in later stages. For class-based filtering, we remove clips with labels such as “background/environment” and “unknown” that lack semantically significant contents.

Table 3.1: Math symbols used in describing data augmentation

Math bol	sym-	Description
A		A set of audio clips sampled from TS-AudioSet
n		the number of clips in A , $n \in [1, 5]$
T		A group of operations to transform the audio clips in A . Operations include Volume, Pitch, Speed, and Duration
p_t		$p_t = 0.3$ is the parameter of Bernoulli distribution that decides if a transformation $T_j \in T$ is applied to an audio clip
C_i		C_i is an augmented audio clip, and $i \in [0, n]$
p_c		$p_c = 0.2$ is the parameter of Bernoulli distribution that decides “mixing” operations for each consecutive pair of augmented clips (C_{i-1}, C_i) . Mixing operation can be either “Concatenation”, or “Mix”

3.2.2 Audio Data Augmentation

Before discussing in details about the audio data enrichment, Table 3.1 lists the math symbols used in this sub section to guide the reading.

To generate a new audio-caption pair, we first select a set of audio clips A uniformly at random from TS-AudioSet. The size of A , n is also selected uniformly from range $[1, 5]$. In our pipeline, we set $\max(n) = 5$, as large n cause the generated audio-caption pair to be unrealistic and overly complex.

Furthermore, we define a set of audio clip transformations T . The four types of transformations are volume, pitch, speed and duration, with implementation details of each transformation given below. Each transformation is also associated with a set of keywords which describe the transformation. For instance, the “volume” transformation is associated with keyword “loud”.

Volume: Given clip C , we randomly apply either amplification or attenuation with uniformly random magnitude in range $[0.5, 1]$ dB to the entire clip.

Pitch: Given clip C , we randomly shift the pitch by a uniformly random number of octaves between $[-0.5, 0.5]$.

Speed: Given clip C , we randomly stretch C in time domain by uniformly random

rate in $[0.8, 1.2]$. For this transformation, we use the *TimeStretch()* function from *torchaudio*, which preserves the pitch rate when modifying speed.

Duration: Given clip C , we randomly reduce the length of C by half.

For each pair of audio clip A_i and transformation T_j , we use a Bernoulli random variable with parameter p_t to determine if T_j should be applied to A_i .

In addition, we define 2 transformations to combine two clips together:

Concatenation: Given clips C_1 and C_2 , we concatenate C_1 and C_2 with 0.5 second of silence separating the clips.

Mix: Given clips C_1 and C_2 , we randomly select a temporal offset for combining C_1 and C_2 . We then draw a signal-to-noise ratio (SNR) between $[-5, 5]$ and mix the two clips together.

Once individual augmentations are applied to each clip, we combine the clips to form the final waveform. Let $[C_0, C_1, \dots, C_n]$ be the augmented clips. For every i , we combine clips (C_{i-1}, C_i) using the mix transformation with probability p_c , or the concatenation transformation with probability $1 - p_c$. In addition, C_{i-1} and C_i are said to occur simultaneously if they are mixed together. Otherwise, C_{i-1} will occur before C_i . We set $p_t = 0.3$ and $p_c = 0.2$ in our pipeline implementation. The final clip is then padded/truncated such that the length of the clip is 10 seconds.

3.2.3 Caption Generation

Instead of using simple concatenation or other techniques in previous works [20] [39] to assemble the captions corresponding to the audio clip, we use LLM to generate natural language description of the new audio clips based on the augmentations applied in the previous step. In this section, we describe how we construct the prompts in order to generate text captions.

To query the LLM, we introduce a JSON-formatted dictionary to describe each clip. The dictionary for clip C contains the original list of labels from TS-AudioSet for

Table 3.2: Example LLM Query and Response. Using the original text labels, a query is constructed based on the operations in audio clips transformations. The basic prompt in the listing 3.1 and query are sent to LLM to generate the response, which is the captions for the augmented audio clip.

Original Labels	Query	Response
1. car driving , firecracker , tree falling	1. [sound: car driving , description: [loud], order: 0 , sound: firecracker , description: [background , high-pitch], order: 0 , sound: tree falling , description: [fast], order: 1]	1. A car drove by loudly as the distant sound of a firecracker echoed in the background; suddenly, a tree fell with a swift crash.
2. train horn , fire engine/ fire truck (siren) , tree falling	2. [sound: train horn , description: [loud], order: 0 , sound: fire engine/fire truck (siren) , description: [background , high-pitch], order: 1 , sound: tree falling , description: [fast], order: 2]	2. The blaring horn of a train resonated loudly, followed by the high-pitched wail of a fire engine’s siren in the background, just before a tree toppled with a rapid crash.

C , the keywords for the transformations applied to C , and an *order* value. We assign the first clip to have an *order* = 0. Furthermore, C_{i-1} and C_i will have the same *order* value if they occur simultaneously, i.e. occurred as the consequence of ‘mixing’, otherwise, their *orders* will incrementally differ by 1. The final query to LLM is a list of the JSON dictionaries, as well as a prompt instructing LLM to generate a short, realistic description based on the dictionary values.

Listing 3.1: Prompt for LLM

I will provide a list of scenarios. For each scenario ,
I want you to provide a short , one sentence story .
Each scenario will be described as a JSON list .
Pay particular attention to the order attribute , which
describes the temporal ordering of the story . Only return
the stories .

The prompt for the LLM is illustrated in the Listing 3.1, and the detailed con-

Table 3.3: Comparison between AudioSetMix and existing audio-language datasets

Dataset	Num. Audio-Caption Pairs	Duration (h)	Avg. Caption Length	Perplexity
AudioCaps	52904	144.94	4.01	1007.25
Clotho	29645	37	16.97	296.76
MACS	17685	9.83	7.01	1174.34
AudioSetMix (Ours)	49971	138.81	20.04	286.46

struction of the full queries to LLM and sample responses are shown in Table 3.2.

We select GPT 3.5 as our LLM of choice for the implementation due to cost and availability of inference resources. However, we note that similar techniques for creating audio-caption pairs can be applied to other LLMs on the market.

3.2.4 Data Postprocessing

We apply additional postprocessing steps to refine the quality of generated captions. We filter out captions by setting a minimum/maximum threshold for word count. This ensures that short captions that lack information are not included, while excluding excessively wordy captions that tend to include unnecessary details or reflect poor grammar or sentence structure. Furthermore, we manually inspected a randomly selected subset of captions to ensure quality.

3.2.5 Dataset Analysis

Table 3.3 shows a comparison of statistics between AudioSetMix and human-annotated audio-language datasets. We note that because up to 5 distinct audio events can occur in a single AudioSetMix clip, a more complex caption is needed to fully describe the entire clip. Thus, the average caption for AudioSetMix is longer than the other datasets. To evaluate the quality of our captions compared to human-generated captions, we use GPT2 [31] to compute the average perplexity of captions. We observe that our captions have lower perplexity comparing to Clotho.

Chapter 4

Experiments

In this section, we study the effectiveness of AudioSetMix for learning text-to-audio retrieval and text-to-audio generation tasks. We additionally study the impact of AudioSetMix for improving model understanding of audio event modifiers, words that describe an attribute of an audio event such as volume or pitch. We provide a description of each task, as well as the experimental settings, results, and analysis. For all experiments performed, we use 16kHz sampling rate and 64-dimensional logmel-spectrogram with 1024-point Hamming window and 160 hop size to compute the audio input.

Table 4.1: R@K on AudioCaps and Clotho

Model	Dataset	AudioCaps			Clotho		
		R@1	R@5	R@10	R@1	R@5	R@10
ResNet38-BERT-medium	Baseline	29.060	65.414	81.546	11.349	28.784	39.387
ResNet38-BERT-medium	Baseline + AudioSetMix	26.850	60.0	78.232	13.454	34.717	<u>47.559</u>
ResNet38-BERT-medium	Baseline+AudioSetMix+Hard Negatives	<u>27.292</u>	<u>63.314</u>	<u>79.116</u>	<u>13.416</u>	<u>34.392</u>	48.306
ResNet38-BERT-base	Baseline	31.270	67.182	80.441	10.755	27.253	38.794
ResNet38-BERT-base	Baseline+AudioSetMix	27.845	<u>64.198</u>	78.121	<u>13.110</u>	<u>34.928</u>	48.631
ResNet38-BERT-base	Baseline+AudioSetMix+Hard Negatives	<u>28.508</u>	64.088	<u>78.453</u>	13.454	35.138	<u>48.325</u>
ReSNet38-RoBERTa-large	Baseline	30.165	66.077	81.436	11.770	29.971	41.665
ReSNet38-RoBERTa-large	Baseline+AudioSetMix	<u>28.066</u>	<u>63.756</u>	<u>77.900</u>	12.669	34.449	47.157
ReSNet38-RoBERTa-large	Baseline+AudioSetMix+Hard Negatives	26.519	60.220	76.685	<u>12.478</u>	<u>33.071</u>	<u>46.870</u>

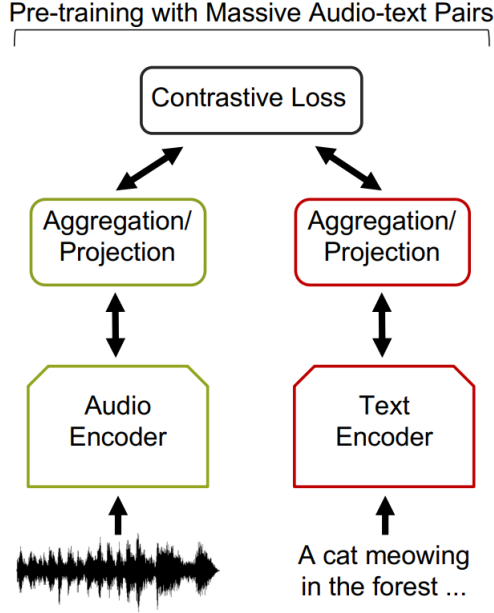


Figure 4.1: The baseline bi-encoder model architecture (from [36]).

4.1 Text-to-Audio Retrieval

Text-to-audio retrieval involves retrieving the audio clip that best matches a given text caption/query from a database of clips. Retrieval is generally done by pushing matching audio-caption pairs closer in an embedding space and keeping non-matching pairs apart [30].

4.1.1 Baseline Models

Following [36], we select a common dual encoder architecture used in the 2022/2023 DCASE Challenge on Language-Based Audio Retrieval [38] (Figure 4.1). This architecture consists of audio encoder E_a and text encoder E_t . For an audio-caption pair (A, T) , the we computes an audio and text embedding, and project them to a shared dimension using a linear layer:

$$\begin{aligned}
a &= \text{Proj}_a(E_a(A)), \\
t &= \text{Proj}_t(E_t(T)).
\end{aligned}
\tag{4.1}$$

Next, assume we have a training batch of B audio-caption pairs $(A_1, T_1), (A_2, T_2), \dots, (A_B, T_B)$. We denote the model's output for the i^{th} pair (A_i, T_i) as a_i and t_i , respectively. We now compute a similarity score s_{ij} between the i th audio clip and the j th caption using dot product:

$$s_{ij} = a_i \cdot t_j^T. \tag{4.2}$$

We use the popular InfoNCE [34] contrastive loss function to train the model. However, because an audio clip can potentially to multiple text captions in the training dataset, we wish to avoid penalizing the model for correctly associating these audio-text pairs when they occur in the same minibatch. Thus, following [15], we introduce a $B \times B$ masking term M where B is the batch size:

$$M_{ij} = \begin{cases} 0, & \text{if } i^{\text{th}} \text{ clip matches } j^{\text{th}} \text{ caption} \\ 1, & \text{otherwise.} \end{cases} \tag{4.3}$$

We slightly modify the InfoNCE loss function with learnable temperature τ using M , as shown in Eq 4.6. We note that when M consists of all ones, Eq 4.6 is identical to the standard InfoNCE loss.

$$L_{TA} = -\frac{1}{B} \sum_{i=1}^B \log \frac{e^{s_{ii}/\tau}}{\sum_{j=1}^B M_{ij} e^{s_{ij}/\tau}}, \tag{4.4}$$

$$L_{AT} = -\frac{1}{B} \sum_{j=1}^B \log \frac{e^{s_{jj}/\tau}}{\sum_{i=1}^B M_{ij} e^{s_{ij}/\tau}}, \tag{4.5}$$

$$L = L_{TA} + L_{AT}. \quad (4.6)$$

Following [38], we select a *ResNet38* model with pretrained weights from PANNs [19] as E_a . To investigate whether more powerful text encoders are better at capturing the presence of audio event modifiers in the captions, we select BERT-medium [4], BERT-base [7], and RoBERTa-large [26] as our choices for E_t .

4.1.2 Text-to-Audio Retrieval Training

Following [38], we combine training sets from multiple sources to form a single, larger training set. We selected AudioCaps [16], Clotho [8], and MACS [27]. Because AudioCaps data consists of YouTube videos may become unavailable over time, we obtain 49k audio-caption pairs out of the original 50k. Combining these datasets gives us a total of 89k training audio-text pairs. We refer to this dataset as the baseline dataset

We trained our models with the loss function defined in Eq 4.6 for 20 epochs using 10^{-5} learning rate, batch size of 64, Adam optimizer, and cosine decay learning rate scheduler.

We evaluate our models using the Recall@K metric (R@K), which is defined as follows: Given a dataset of audio-caption pairs, we first compute embeddings for caption and audio clip using the pretrained models respectively. Next, for each caption, we compute the similarity between its embedding and all audio clips using Eq 4.2. The Recall@K metric is defined as the probability that the top k most similar audio clips contains the targeted ground-truth clip. Following, [36], we use the test splits from AudioCaps and Clotho as the evaluation datasets. We report the recall for different k for our baseline models in Table 4.1.

Table 4.2: R@10 for original and flipped captions in MTe, separated by modifier category

Caption Type	Duration	Pitch	Speed	Volume
Flipped	96.43	90.91	74.08	67.33
Original	98.21	88.63	74.81	69.34

4.1.3 Evaluating Modifier Understanding for Text-to-Audio Retrieval

As shown in [36], existing audio-language models depend heavily on the keywords in the caption, which are typically nouns and verbs. [36] finds that this over-reliance on keywords causes current audio-language models to not capture the order of the audio event. Motivated by [36], we extended, and investigated whether audio-language models also fail to “understand” the modifiers for events, such as *loud* vs. *quiet*. We study the impact from four categories of common modifiers {volume, pitch, speed, and duration} for audio events, to the models. This is aligned with the methods to produce the AudioSetMix in Chapter 3. To do this, we create a subset from the Clotho and AudioCaps evaluation sets called “Modifier Test Set” (MTe). Captions in MTe contain words describing one of these modifiers, such as *loud*, *slow*, *short* and etc.. In total, this gives us 700 pairs of data in MTe.

We first determine if existing audio-language models can capture modifiers in the caption. For this experiment, we select the ResNet38-Bert-base model as our baseline. Following the BAT test introduced in [36], we replace each modifier in MTe with an antonym, forming what we call the flipped caption. For instance, the modifier *loud* would be replaced with *quiet*, and the modifier *quickly* would be replaced with *slowly*. We then use the flipped captions to retrieve the original audio and report the recall as performance metrics. The retrieval set is the set of all other audio-text pairs containing the same class of modifier. If the model is able to distinguish modifiers, we would expect that the recall scores would degrade significantly with “flipped caption”

Table 4.3: Performance on MUT for model trained on the baseline dataset, the augmented dataset, and the augmented dataset with hard negatives. Performance is measured as the percentage of samples for which the flipped text caption is closer to the audio than the original caption (lower is better).

Model	Dataset	Duration	Pitch	Speed	Volume
ResNet38-BERT-medium	Baseline	42.857	40.909	46.667	44.889
ResNet38-BERT-medium	Baseline+AudioSetMix	<u>35.714</u>	<u>43.181</u>	<u>36.296</u>	<u>37.875</u>
ResNet38-BERT-medium	Baseline+AudioSetMix+Hard Negatives	33.928	45.450	32.592	35.871
ResNet38-BERT-base	Baseline	44.643	<u>31.818</u>	38.519	<u>33.066</u>
ResNet38-BERT-base	Baseline+AudioSetMix	<u>37.50</u>	38.636	<u>33.334</u>	33.266
ResNet38-BERT-base	Baseline+AudioSetMix+Hard Negatives	35.714	<u>31.818</u>	31.111	28.857
ResNet38-RoBERTa-large	Baseline	48.214	40.909	39.259	36.272
ResNet38-RoBERTa-large	Baseline+AudioSetMix	<u>46.428</u>	<u>47.727</u>	35.560	31.863
ResNet38-RoBERTa-large	Baseline+AudioSetMix+Hard Negatives	37.50	50.0	<u>38.518</u>	<u>32.865</u>

in comparison to using the original captions as the query. The results are reported in Table 4.2. We see only a marginal change in performance, which suggest that the models failed to learn to distinguish the modifiers well.

To more rigorously study model understanding of modifiers, we design the Modifier Understanding Test (MUT). For each audio-caption pair in MTe, We first compute the embedding distances between the original audio and both the flipped and original captions. We then count the percentage of times that the flipped caption embedding is closer to the audio embedding than the original caption embedding. If the model completely fails to capture the presence of modifiers, we should expect that the flipped caption is closer 50% of the time. On the other hand, a perfect model should have a score of 0%. As shown by the results in Table 4.3, the model unsurprisingly performs significantly better than random choice in every modifier category, but is far from perfect.

Finally, we wish to evaluate the model’s ability to distinguish between different categories of modifiers. We perform retrieval on MTe using the original captions as queries where the retrieval set is the set of every audio-text pairs in MTe, and report the recall scores in Table 4.4. We refer to this experiment as Modifier Differentiating Test (MDT).

Table 4.4: Performance on MDT for model trained on the baseline dataset, the augmented dataset, and the augmented dataset with hard negatives.

Model	Dataset	R@1	R@5	R@10
ResNet38-BERT-medium	Baseline	15.258	47.002	59.809
ResNet38-BERT-medium	Baseline+AudioSetMix	15.208	53.678	67.438
ResNet38-BERT-medium	Baseline+AudioSetMix+Hard Negatives	16.485	56.675	68.801
ResNet38-BERT-base	Baseline	15.122	47.002	61.580
ResNet38-BERT-base	Baseline+AudioSetMix	<u>16.893</u>	<u>55.722</u>	<u>70.163</u>
ResNet38-BERT-base	Baseline+AudioSetMix+Hard Negatives	17.029	55.858	70.980
ResNet38-RoBERTa-large	Baseline	14.032	47.138	61.989
ResNet38-RoBERTa-large	Baseline+AudioSetMix	16.076	55.585	69.891
ResNet38-RoBERTa-large	Baseline+AudioSetMix+Hard Negatives	<u>15.122</u>	<u>53.814</u>	<u>68.528</u>

Table 4.5: Top: Number of sentences in MTe containing each modifier type, with % indicating size relative to the full test set. Bottom: the same distribution for pretraining set, with % indicating size relative to the full pretraining set.

Dataset	Duration	Pitch	Speed	Volume
MTe	56 (0.5%)	44 (0.4%)	135 (1.3%)	499 (4.8%)
Training	452 (0.5%)	347 (0.3%)	778 (0.8%)	4540 (5.1%)

4.1.4 Training with AudioSetMix

The percentage of sentences in the training data that contain modifiers is extremely small, as shown in Table 4.5. As such, we study whether increase the number of modifiers in the training data improves understanding of modifiers. We augment the baseline dataset using AudioSetMix, giving us a total of 132k training audio-text pairs. We train new models using the same training procedure as the baseline. As shown in Table 4.3, the models show significant gains in understanding duration, speed, and volume modifiers when trained with AudioSetMix. Furthermore, Table 4.4 shows that all models improve in their ability to distinguish between the different modifier categories.

4.1.5 Training with Generated Hard Negatives

In contrastive learning, each audio clip A_i is contrasted with other texts T_j , which usually describe completely different clips. As such, models are able to ignore finer

details of modifiers in clips and simply focus on audio events [3]. We hypothesize that hard negative examples that contain the same audio events are needed to encourage models to capture and understand modifiers. In contrast to [30], our data generation pipeline Figure 3.1 provides a natural way to generate hard negatives for each data point in AudioSetMix. Recall that in the data generation process, we sample a set of audio clips A for augmentation. We randomly select a set of augmentation operations T for $A_i \in A$, with each $T_j \in T$ applies to A_i , and produced the augmented clips C_i . We finally assemble the audio clips in $[C_i]$ by mixture of concatenation and mixing. For hard negatives, while keeping the same procedure as outlined above, we “reverse” each operation T_j (i.e. if the original T_j is *loud*, the new transformation T'_j is *quiet*). We argue that this creates hard negative samples since the two final augmented clips contain the same set of audio events, but with opposite augmentation operations applied to each. This makes it more challenging for the model to learn to distinguish them and forces the model to attend to the audio modifiers.

To train with these hard negatives, we randomly select c AudioSetMix inputs in each minibatch and generate their hard negatives. We then append the hard negatives to the minibatch and perform the model update. We empirically find that $c = 16$ works well for our batch size setting of 64. In Table 4.3, we show that model performance on MUT is generally improved using hard negatives on all modifier categories except pitch. We hypothesize that pitch may be difficult to improve upon because pitch is not as broadly applicable of a modifier, meaning that pitch augmentation in AudioSetMix may not be meaningful semantically (i.e. applying a high-pitch transformation to a base sound of “tree falling over”).

Finally, we compare our models trained on AudioSetMix and hard negatives with the baseline models on the original text-to-audio retrieval task. We evaluate each model using the evaluation set from AudioCaps and Clotho and show the results in Table 4.1. We note that the models trained using AudioSetMix and hard negatives

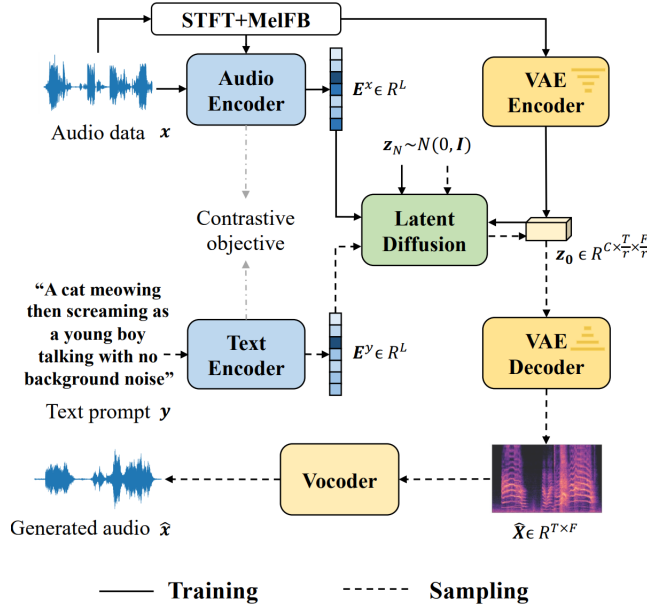


Figure 4.2: The AudioLDM architecture for Text-to-Audio Generation (from [24]).

Table 4.6: Results on AudioCaps

Model	Dataset	FD ↓	IS ↑	KL ↓
AudioLDM-S	AudioCaps	31.881	8.090	1.605
AudioLDM-S	AudioSetMix	33.684	7.598	1.583

beat the baseline results consistently on Clotho. However, they are not performing on par to baseline results on AudioCaps. We conjecture that this is due to the non-naturalistic captions in AudioCaps compared to Clotho and AudioSetMix (refer to Table 3.3), which result in poorer performance for models trained by “more” natural language captions rather than simple keyword-like captions in AudioCaps. Furthermore, in contrast to the previous experiments, the addition of hard negative mining provides only a marginal improvement to the recall scores. Because a) the hard negatives are only concerned with modifiers and b) the lack of modifiers in the eval sets, we hypothesize that the benefits of hard negative mining are not observable in this experiment.

4.2 Text-to-Audio Generation

We now study if AudioSetMix improves performance on the text-to-audio generation task. Text-to-audio generation involves generating natural sounds, conditioned on a textual description of the sound.

4.2.1 Baseline Model

We select AudioLDM-S [24] as our baseline model. AudioLDM-S is a latent diffusion model that uses CLAP [9] model to substitute audio embeddings in the place of text. To reduce the dimensionality of data and speed up inference and sampling, AudioLDM uses a variational autoencoder (VAE) to embed the audio features to a lower dimension latent space. The diffusion probabilistic model is then trained on this latent space. At sample time, the output of the diffusion model is upsampled using the VAE decoder. Finally, the HiFi-GAN [18] vocoder model is used to convert the mel-spectrogram to a waveform.

4.2.2 Text-to-Audio Generation Training

Due to computational limitations, we finetune the pretrained AudioLDM-S model from the implementation of AudioLDM¹ rather than train from scratch. We additionally use the provided CLAP Model, VAE, and vocoder checkpoints. We finetune AudioLDM-S on both AudioCaps and AudioSetMix using the reweighted noise estimation training objective introduced in [13]:

$$L_n(\theta) = \mathbb{E}_{z_0, \epsilon, n} \|\epsilon - \epsilon_\theta(z_n, n, E^x)\|_2^2 \quad (4.7)$$

where $\epsilon \sim N(0, I)$ is standard Gaussian, E^x is the text condition embedding, and z_n is the n th latent in the forward diffusion process. We use a batch size of 8, a

¹<https://github.com/haoheliu/AudioLDM-training-finetuning>

learning rate of 3×10^{-5} , Adam optimizer, and finetune AudioLDM-S for 200k steps.

Following [24], we use inception score (IS), Frechet Distance (FD), and KL divergence (KL) computed on the AudioCaps evaluation set as metrics for our text-to-audio generation model. FD measures the similarity between two audio data distributions. IS measures the quality and diversity of generated samples. KL measures the correlation between output samples and target samples. Each metric is based on the PANNs [19] audio classification model. We use the evaluation pipeline provided by [24] for fairer comparison.

4.2.3 Training With AudioSetMix

We compare the IS, FD, and KL of AudioLDM-S finetuned on AudioCaps and AudioSetMix in Table 4.6. We find that finetuning with AudioSetMix does not lead to a significant improvement. We hypothesize that this is due to a) the lack of modifiers in the evaluation set means that the benefits of AudioSetMix are largely hidden; b) The relative size of AudioSetMix with respect to the AudioLDM-S pretraining dataset causes its impact to be lessened. For comparison, WavCaps dataset is an order of magnitude larger than AudioSetMix.

Chapter 5

Conclusion

In this work, we introduce AudioSetMix, a weakly-labelled audio-caption pair dataset created by applying audio transformations to existing datasets. We propose a pipeline to augment/combine audio clips and generate a corresponding caption using LLM. We evaluate AudioSetMix on text-to-audio retrieval and text-to-audio generation and demonstrate that AudioSetMix improves model understanding of audio event modifiers. In future we hope to evaluate models trained from AudioSetMix using human feedback.

Bibliography

- [1] A. Agostinelli, T. I. Denk, Z. Borsos, J. Engel, M. Verzett, A. Caillon, Q. Huang, A. Jansen, A. Roberts, M. Tagliasacchi, M. Sharifi, N. Zeghidour, and C. Frank. Musiclm: Generating music from text, 2023.
- [2] A. Agrawal, J. Lu, S. Antol, M. Mitchell, C. L. Zitnick, D. Batra, and D. Parikh. Vqa: Visual question answering, 2016.
- [3] P. Bagad, M. Tapaswi, and C. G. M. Snoek. Test of time: Instilling video-language models with a sense of time, 2023.
- [4] P. Bhargava, A. Drozd, and A. Rogers. Generalization in nli: Ways (not) to go beyond simple heuristics, 2021.
- [5] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020.
- [6] P. Chen, Y. Zhang, M. Tan, H. Xiao, D. Huang, and C. Gan. Generating visually aligned sound from videos. *IEEE Transactions on Image Processing*, 29:8292–8302, 2020.

- [7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.
- [8] K. Drossos, S. Lipping, and T. Virtanen. Clotho: An audio captioning dataset, 2019.
- [9] B. Elizalde, S. Deshmukh, M. A. Ismail, and H. Wang. Clap: Learning audio concepts from natural language supervision, 2022.
- [10] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.
- [11] F. Gontier, R. Serizel, and C. Cerisara. Automated audio captioning by fine-tuning bart with audioset tags. In *Proceedings of the 6th Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE2021)*, pages 170–174, Barcelona, Spain, November 2021.
- [12] S. Hershey, D. P. W. Ellis, E. Fonseca, A. Jansen, C. Liu, R. C. Moore, and M. Plakal. The benefit of temporally-strong labels in audio event classification, 2021.
- [13] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models, 2020.
- [14] R. Huang, J. Huang, D. Yang, Y. Ren, L. Liu, M. Li, Z. Ye, J. Liu, X. Yin, and Z. Zhao. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models, 2023.
- [15] G. Ilharco, Y. Zhang, and J. Baldridge. Large-scale representation learning from visually grounded untranscribed speech, 2019.
- [16] C. D. Kim, B. Kim, H. Lee, and G. Kim. Audiocaps: Generating captions for audios in the wild. In *NAACL-HLT*, 2019.

- [17] A. S. Koepke, A.-M. Oncescu, J. F. Henriques, Z. Akata, and S. Albanie. Audio retrieval with natural language queries: A benchmark study. *IEEE Transactions on Multimedia*, 25:2675–2685, 2023.
- [18] J. Kong, J. Kim, and J. Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis, 2020.
- [19] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley. Panns: Large-scale pretrained audio neural networks for audio pattern recognition, 2020.
- [20] F. Kreuk, G. Synnaeve, A. Polyak, U. Singer, A. Défossez, J. Copet, D. Parikh, Y. Taigman, and Y. Adi. Audiogen: Textually guided audio generation, 2023.
- [21] J. Li, D. Li, C. Xiong, and S. Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation, 2022.
- [22] X. Li, X. Yin, C. Li, P. Zhang, X. Hu, L. Zhang, L. Wang, H. Hu, L. Dong, F. Wei, Y. Choi, and J. Gao. Oscar: Object-semantics aligned pre-training for vision-language tasks, 2020.
- [23] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft coco: Common objects in context, 2015.
- [24] H. Liu, Z. Chen, Y. Yuan, X. Mei, X. Liu, D. Mandic, W. Wang, and M. D. Plumbley. Audioldm: Text-to-audio generation with latent diffusion models, 2023.
- [25] H. Liu, Q. Tian, Y. Yuan, X. Liu, X. Mei, Q. Kong, Y. Wang, W. Wang, Y. Wang, and M. D. Plumbley. Audioldm 2: Learning holistic audio generation with self-supervised pretraining, 2023.

- [26] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.
- [27] I. Martin-Morato and A. Mesaros. What is the ground truth? reliability of multi-annotator data for audio tagging, 2021.
- [28] X. Mei, X. Liu, M. D. Plumbley, and W. Wang. Automated audio captioning: an overview of recent progress and new challenges. *EURASIP Journal on Audio, Speech, and Music Processing*, 2022(1), Oct. 2022.
- [29] X. Mei, X. Liu, J. Sun, M. D. Plumbley, and W. Wang. On metric learning for audio-text cross-modal retrieval, 2022.
- [30] X. Mei, C. Meng, H. Liu, Q. Kong, T. Ko, C. Zhao, M. D. Plumbley, Y. Zou, and W. Wang. Wavcaps: A chatgpt-assisted weakly-labelled audio captioning dataset for audio-language multimodal research, 2023.
- [31] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.
- [32] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents, 2022.
- [33] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models, 2022.
- [34] A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding, 2019.
- [35] Y. Wang, Z. Ju, X. Tan, L. He, Z. Wu, J. Bian, and S. Zhao. Audit: Audio editing by following instructions with latent diffusion models, 2023.

- [36] H.-H. Wu, O. Nieto, J. P. Bello, and J. Salamon. Audio-text models do not yet leverage natural language, 2023.
- [37] H. Xie, S. Lipping, and T. Virtanen. Language-based audio retrieval task in dcase 2022 challenge, 2022.
- [38] X. Xu, Z. Xie, M. Wu, and K. Yu. The SJTU system for DCASE2022 challenge task 6: Audio captioning with audio-text retrieval pre-training. Technical report, DCASE2022 Challenge, July 2022.
- [39] D. Yang, J. Yu, H. Wang, W. Wang, C. Weng, Y. Zou, and D. Yu. Diffsound: Discrete diffusion model for text-to-sound generation, 2023.
- [40] U. Zölzer. *Digital Audio Signal Processing*. John Wiley & Sons, 2008.