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

David Xu
Princeton University
dx8527@princeton.edu

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.

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) [21] [22], text-guided audio editing [32], automatic audio-captioning [25] [10], and text-to-audio retrieval [26] [15] have been proposed as objectives that improve model understanding of audio signal [27].

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 [29] [30]. However, unlike audio-language, the vision-language learning benefits from the existence of large-scale, high quality datasets such as MSCOCO [20]. This makes it possible to pretrain large, powerful models to learn vision-language multimodal embeddings, which can then be applied for downstream tasks [19] [18] [30]. 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 [9], 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 [14] 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 [27].

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) [34] 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.

2. Background and Related Work

2.1. 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 [17], in which audio labels are concatenated together in a random order. [17] 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, [36] randomly inserts $\langle MASK \rangle$ tokens between labels in the hope that the model will learn to substitute in relational words. [12] 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 pretrained CLAP model to embed audio and text to a shared latent space [21] [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 [21], 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 [27], 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.

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 background noise [11]. 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, [11] 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 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

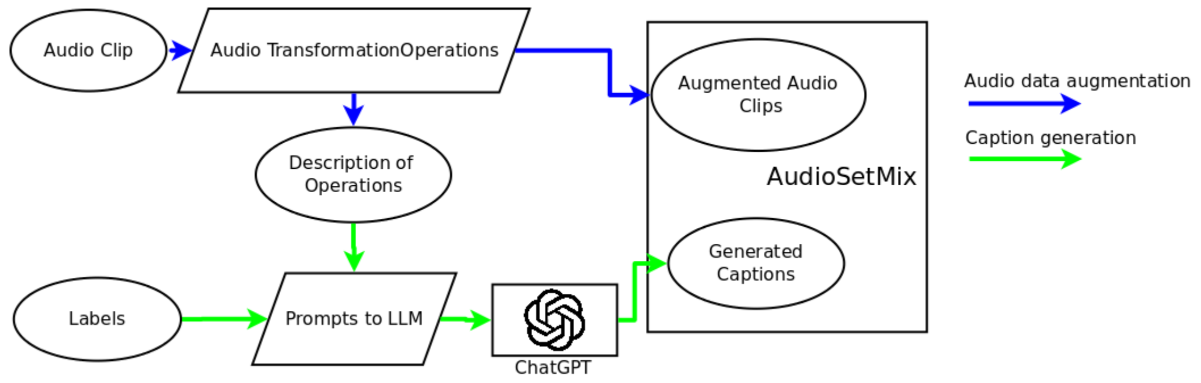

Figure 1. The overall pipeline to generate the AudioSetMix.

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 [27]. 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.

3.2.2 Audio Data Augmentation

Before discussing in details about the audio data enrichment, Table 1 lists the math symbols used in this sub section to guide the reading.

To generate a new audio-caption pair, we first select n audio clips A uniformly at random from TS-AudioSet. n is also selected uniformly from range $[1, 5]$, 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 $[C_0, C_1, ...C_n]$ 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 [17] [36] 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 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 $order$s will incrementally differ by 1. The final query to LLM is a list of the JSON dic-

Table 1. Math symbols used in describing data augmentation

Math symbol	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"

Table 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 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.

tionaries, as well as a prompt instructing LLM to generate a short, realistic description based on the dictionary values.

Listing 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 1, and the detailed construction of the full queries to LLM and sample responses are shown in Table 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 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 [28] to compute the average perplexity of captions. We observe that our captions have lower perplexity comparing to Clotho.

Table 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

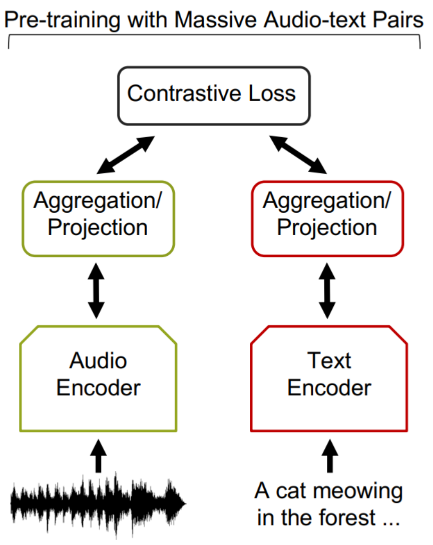

Figure 2. The baseline bi-encoder model architecture (from [33]).

4. Experiments

In this section, we study the effectiveness of AudioSetMix for learning text-to-audio retrieval task. 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.

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 [27].

4.1.1 Baseline Models

Following [33], we select a common dual encoder architecture used in the 2022/2023 DCASE Challange on Language-Based Audio Retrieval [35] (Figure 2). 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:

$$\begin{aligned} a &= \text{Proj}_a(E_a(A)), \\ t &= \text{Proj}_t(E_t(T)). \end{aligned} \tag{1}$$

Next, assume we have a training batch of B audio-caption pairs $(A_1, T_1), (A_2, T_2), ...(A_B, T_B)$. 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 ith audio clip and the jth caption using dot product:

$$s_{ij} = a_i \cdot t_j^T. \tag{2}$$

We use the popular InfoNCE [31] 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 [13], we introduce a $B \times B$ masking term M where B is the batch size:

$$M_{ij} = \begin{cases} 0, \text{ if } i^{th} \text{ clip matches } j^{th} \text{ caption} \\ 1, \text{ otherwise.} \end{cases} \tag{3}$$

We slightly modify the InfoNCE loss function with learnable temperature τ using M, as shown in Eq 6. We note that when M consists of all ones, Eq 6 is identical to the standard InfoNCE loss.

$$L_{TA} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{e^{\mathbf{s}_{ii}/\tau}}{\sum_{j=1}^{B} \mathbf{M}_{ij} e^{\mathbf{s}_{ij}/\tau}}, \tag{4}$$

$$L_{AT} = -\frac{1}{B}\sum_{j=1}^{B} \log \frac{e^{\mathbf{s}_{jj}/\tau}}{\sum_{i=1}^{B} \mathbf{M}_{ij} e^{\mathbf{s}_{ij}}/\tau}, \tag{5}$$

$$L = L_{TA} + L_{AT}. \tag{6}$$

Following [35], we select a *ResNet38* model with pretrained weights from PANNs [16] 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 [23] as our choices for E_t.

Table 4. R@K on Clotho test set

Model	Dataset	R@1	R@5	R@10
ResNet38-BERT-medium	Baseline	11.349	28.784	39.387
ResNet38-BERT-medium	Baseline + AudioSetMix	**13.454**	**34.717**	47.559
ResNet38-BERT-medium	Baseline+AudioSetMix+Hard Negatives	13.416	34.392	**48.306**
ResNet38-BERT-base	Baseline	10.755	27.253	38.794
ResNet38-BERT-base	Baseline+AudioSetMix	13.110	34.928	**48.631**
ResNet38-BERT-base	Baseline+AudioSetMix+Hard Negatives	**13.454**	**35.138**	48.325
ReSNet38-RoBERTa-large	Baseline	11.770	29.971	41.665
ReSNet38-RoBERTa-large	Baseline+AudioSetMix	**12.669**	**34.449**	**47.157**
ReSNet38-RoBERTa-large	Baseline+AudioSetMix+Hard Negatives	12.478	33.071	46.870

4.1.2 Text-to-Audio Retrieval Training

Following [35], we combine training sets from multiple sources to form a single, larger training set. We selected AudioCaps [14], Clotho [8], and MACS [24]. 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 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 2. The Recall@K metric is defined as the probability that the top k most similar audio clips contains the targeted ground-truth clip. We use the test split from Clotho as the evaluation dataset. We report the recall for different k for our baseline models in Table 4.

4.1.3 Evaluating Modifier Understanding for Text-to-Audio Retrieval

As shown in [33], existing audio-language models depend heavily on the keywords in the caption, which are typically nouns and verbs. [33] finds that this over-reliance on keywords causes current audio-language models to not capture the order of the audio event. Motivated by [33], 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 Section 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 [33], 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" in comparison to using the original captions as the query. The results are reported in Table 5. 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 6, the model unsurprisingly performs significantly better than random choice in every modifier category, but is far from perfect.

Table 5. 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

Table 6. 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	35.714	43.181	36.296	37.875
ResNet38-BERT-medium	Baseline+AudioSetMix+Hard Negatives	**33.928**	45.450	**32.592**	**35.871**
ResNet38-BERT-base	Baseline	44.643	31.818	38.519	33.066
ResNet38-BERT-base	Baseline+AudioSetMix	37.50	38.636	33.334	33.266
ResNet38-BERT-base	Baseline+AudioSetMix+Hard Negatives	**35.714**	31.818	**31.111**	**28.857**
ResNet38-RoBERTa-large	Baseline	48.214	**40.909**	39.259	36.272
ResNet38-RoBERTa-large	Baseline+AudioSetMix	46.428	47.727	**35.560**	**31.863**
ResNet38-RoBERTa-large	Baseline+AudioSetMix+Hard Negatives	**37.50**	50.0	38.518	32.865

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 7. We refer to this experiment as Modifier Differentiating Test (MDT).

4.1.4 Training with AudioSetMix

The percentage of sentences in the training data that contain modifiers is extremely small, as shown in Table 8. 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 6, the models show significant gains in understanding duration, speed, and volume modifiers when trained with AudioSetMix. Furthermore, Table 7 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 [27], our data generation pipeline Figure 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 6, 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 Clotho and show the results in Table 4. We note that the models trained using AudioSetMix and hard negatives beat the baseline results consistently on Clotho. In contrast to the previous experiments, the addition of hard negative mining provides only a marginal improve-

Table 7. 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	<u>15.258</u>	47.002	59.809
ResNet38-BERT-medium	Baseline+AudioSetMix	15.208	<u>53.678</u>	<u>67.438</u>
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 8. 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%)

ment 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.

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 demonstrate that AudioSetMix improves model understanding of audio event modifiers. In future we hope to evaluate models trained from AudioSetMix using human feedback.

References

[1] Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. Musiclm: Generating music from text, 2023.

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

[3] Piyush Bagad, Makarand Tapaswi, and Cees G. M. Snoek. Test of time: Instilling video-language models with a sense of time, 2023.

[4] Prajjwal Bhargava, Aleksandr Drozd, and Anna Rogers. Generalization in nli: Ways (not) to go beyond simple heuristics, 2021.

[5] 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, 2020.

[6] Peihao Chen, Yang Zhang, Mingkui Tan, Hongdong Xiao, Deng Huang, and Chuang Gan. Generating visually aligned sound from videos. *IEEE Transactions on Image Processing*, 29:8292–8302, 2020.

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

[8] Konstantinos Drossos, Samuel Lipping, and Tuomas Virtanen. Clotho: An audio captioning dataset, 2019.

[9] 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 *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[10] Felix Gontier, Romain Serizel, and Christophe 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.

[11] 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, 2021.

[12] 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, 2023.

[13] Gabriel Ilharco, Yuan Zhang, and Jason Baldridge. Large-scale representation learning from visually grounded untranscribed speech, 2019.

[14] Chris Dongjoo Kim, Byeongchang Kim, Hyunmin Lee, and Gunhee Kim. Audiocaps: Generating captions for audios in the wild. In *NAACL-HLT*, 2019.

[15] A. Sophia Koepke, Andreea-Maria Oncescu, João F. Henriques, Zeynep Akata, and Samuel Albanie. Audio retrieval with natural language queries: A benchmark study. *IEEE Transactions on Multimedia*, 25:2675–2685, 2023.

[16] 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, 2020.

[17] Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. Audiogen: Textually guided audio generation, 2023.

[18] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation, 2022.

[19] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. Oscar: Object-semantics aligned pre-training for vision-language tasks, 2020.

[20] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[21] 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, 2023.

[22] Haohe Liu, Qiao Tian, Yi Yuan, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Yuping Wang, Wenwu Wang, Yuxuan Wang, and Mark D. Plumbley. Audioldm 2: Learning holistic audio generation with self-supervised pretraining, 2023.

[23] 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, 2019.

[24] Irene Martin-Morato and Annamaria Mesaros. What is the ground truth? reliability of multi-annotator data for audio tagging, 2021.

[25] Xinhao Mei, Xubo Liu, Mark D. Plumbley, and Wenwu Wang. Automated audio captioning: an overview of recent progress and new challenges. *EURASIP Journal on Audio, Speech, and Music Processing*, 2022(1), October 2022.

[26] Xinhao Mei, Xubo Liu, Jianyuan Sun, Mark D. Plumbley, and Wenwu Wang. On metric learning for audio-text cross-modal retrieval, 2022.

[27] Xinhao Mei, Chutong Meng, Haohe Liu, Qiuqiang Kong, Tom Ko, Chengqi Zhao, Mark D. Plumbley, Yuexian Zou,

and Wenwu Wang. Wavcaps: A chatgpt-assisted weakly-labelled audio captioning dataset for audio-language multi-modal research, 2023.

[28] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[29] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022.

[30] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[31] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019.

[32] Yuancheng Wang, Zeqian Ju, Xu Tan, Lei He, Zhizheng Wu, Jiang Bian, and Sheng Zhao. Audit: Audio editing by following instructions with latent diffusion models, 2023.

[33] Ho-Hsiang Wu, Oriol Nieto, Juan Pablo Bello, and Justin Salamon. Audio-text models do not yet leverage natural language, 2023.

[34] Huang Xie, Samuel Lipping, and Tuomas Virtanen. Language-based audio retrieval task in dcase 2022 challenge, 2022.

[35] Xuenan Xu, Zeyu Xie, Mengyue Wu, and Kai Yu. The SJTU system for DCASE2022 challenge task 6: Audio captioning with audio-text retrieval pre-training. Technical report, DCASE2022 Challenge, July 2022.

[36] Dongchao Yang, Jianwei Yu, Helin Wang, Wen Wang, Chao Weng, Yuexian Zou, and Dong Yu. Diffsound: Discrete diffusion model for text-to-sound generation, 2023.