TAKE IT SERIOUSLY: IMPROVING ON LAST YEAR WITH ALL SORTS OF TRICKS

Technical Report

Theodore Lamort de Gail, Bartłomiej Zgórzyński, Anna Płęs, Kamil Górzyński

Samsung R&D Institute Poland, Warsaw, Poland
{t.lamort, b.zgorzynski, a.ples, k.gorzynski2}@samsung.com

ABSTRACT

In this report, we present our solution to DCASE 2023, task 6B: Language-Based Audio Retrieval. We employ a bi-encoder architecture trained using contrastive ranking loss. The audio encoder is an ensemble of three pre-trained models (BEATs, VGGish, CLAP) with added self-attention heads, while the text encoder is a pre-trained MPNet. To address the small dataset size, we gather 1.7M caption-audio pairs from YouTube videos. We use MixGen and paraphrasing, as well as traditional audio augmentation, and Low-Rank Adaptation (LoRA) for fine-tuning on Clotho.

We achieve 29.66% mAP@10 on the development-testing split of Clotho using an ensemble solution, and 26.93% mAP@10 with a single model. We also submit an ensemble of our solution and CLAP.

1. INTRODUCTION

Language-Based Audio Retrieval is the task of retrieving audio recordings from a database given a natural language description. Our system uses the standard approach of a bi-encoder producing text and audio embeddings, which are subsequently compared with cosine similarity.

We encounter two significant challenges. The first revolves around enhancing the system architecture, while the second and more pressing pertains to the scarcity of data. To address them, we have made notable advancements in our previous year's solution [1].

As described in Section 1, we gather and filter a diverse range of data, including our newly collected **YouTube** audio-caption pairs as well as samples obtained from various additional sources. We employ **MixGen**[2] to combine multiple samples from distinct origins to strengthen the data robustness further.

We employ an ensemble of pretrained audio expert models. The specific details regarding architectural improvements can be found in Section 3. In Section 4, we provide a description of the pre-training and fine-tuning process. One of the critical differences is incorporating Low-Rank Adaptation[3] to reduce overfitting while fine-tuning our model on the **Clotho 2.1**[4] dataset.

We develop three distinct ensembles, as explained in Section 5. The final results are detailed in Section 6.

2. DATA

As the ground development dataset, we take **Clotho v2.1**[4]. We support it with three more datasets that contain captioned audio recordings: **AudioCaps**[5], **SoundDescs**[6] and **MACS**[7].

Furthermore, we include the **Freesound**[8] dataset that we used in our submission last year[1]. The set consists of 15K text-audio pairs scraped from Freesound descriptions in a semi-supervised way using a text classifier.

In order to scale up the training dataset, we add 1.7M caption-audio pairs extracted from **YouTube** videos. We query for creative-commons licensed clips that include subtitles. We search for closed captions by matching text enclosed in square brackets, parentheses and asterisks. We apply filtering heuristics such as discarding captions containing pronouns, numeric and punctuation signs, or any of the words from a handmade blacklist ("subscribe", "inaudible", "let's", and so on). We use Spacy for Named Entity Recognition[9] and name/gender statistics to replace person names with "man", "woman" or "person". We download the audio of the corresponding clips with an addditional leading and trailing 3 seconds.

We notice that the captions are very thematically unbalanced: a large portion of the data consists of a small set of common captions such as "music" and "applause". To even out the distribution, training-time sampling is performed with custom weights. In order to compute weights, first we embed each caption using *all-mpnet-base-v2* [10]. Next, we compute pairwise cosine similarity among embeddings, with a batch size of $n = 5000$. The weight of the i-th sample in the batch is then

$$w_i = 1/\sum_{j=1}^{n} \text{sim}(v_i, v_j),$$

where sim is the cosine similarity, and v_k is the embedding of the k-th caption in the batch.

Augmentation. We use standard audio augmentation including gain, time masking, random cropping, and noise mixing. Furthermore, we use Pegasus[11] for caption paraphrasing and MixGen[2] to create more pairs by concatenating captions and adding audio.

3. ARCHITECTURE

To establish cross-modal embeddings, we train an audio encoder and a caption encoder with matching output dimensions. During inference, the audio segments are selected based on cosine similarity. For caption embeddings, we utilize the SentenceTransformers[10] library, specifically the *all-mpnet-base-v2* model. We do not freeze the weights. We use a single linear layer to map this to a common dimension of 256. In order to encode audio, we initially compute per-time-frame embeddings using three pre-trained expert models.

1. **CLAP** [12] is a deep learning model that utilizes natural language descriptions to learn and associate audio concepts. By training on paired audio samples and textual descriptions, CLAP can effectively understand and recognize specific audio

characteristics described in natural language. We only use the audio encoder, with an inference window of 5 seconds and a hop length of 2.5 seconds. Code can be found here[1].

2. **BEATS** [13] We obtain a time-contextualized sequence of embeddings of dimension 768, to which we apply one-dimensional average pooling with a kernel of 60 and stride of 30, to obtain approximately 2 frames per second.
3. **VGGish** [14]. We use this[2] PyTorch implementation, with a hop size modified to 0.5 seconds. The inference window is 1 second.

The weights of the audio experts are kept frozen. Each sequence of embeddings is mapped through its own transformer model, and then averaged. Finally, we take the average per-expert. For more details, please refer to our report from last year [1].

4. TRAINING

Sampling. During each training step, we adopt a random sampling approach instead of iterating over the entire dataset. First, we randomly select a dataset based on predefined weights, which are hyperparameters. Then, we randomly choose an audio sample from the selected dataset. In the case of our YouTube-collected data, the selection is also weighted. If multiple captions are associated with the chosen audio sample, we randomly select one of them. We construct a batch of such positive pairs $(\text{audio}_i, \text{caption}_i)$, and consider all other pairs $(\text{audio}_i, \text{caption}_j)$ where $i \neq j$ to be negative.

Loss. The system is trained using a contrastive ranking loss with a batch size of 512. For a given batch, if we let $\alpha_1, \alpha_2, ..., \alpha_B$ be the audio embeddings and $\beta_1, \beta_2, ..., \beta_B$ be the corresponding caption embeddings, and we denote the cosine similarity between α_i and β_j by s_{ij} then the loss is expressed as

$$\mathcal{L} = \frac{1}{B} \sum_{1 \leq i,j \leq B,\ i \neq j} [m + s_{ij} - s_{ii}]_+ + [m + s_{ji} - s_{ii}]_+$$

where $[\cdot]_+$ is the hinge function $\max(\cdot, 0)$ and m is the *margin*, set to 0.4.

Optimizer. We utilize AdamW as our optimizer with a learning rate of 1e-4. Additionally, we incorporate the CosineAnnealingWarmRestarts scheduler with the following parameters: $T_0 = 5, T_{mult} = 2, \eta_{min} = 0$.

Monitored metrics. We monitor validation loss, as well as batch-wise validation text-to-audio R@1, expressed as

$$\frac{1}{B} ||\{s_{ii} = \max_j(s_{ji})\}||$$

where B is the batch size, s_{ij} is as defined in the loss subsection, and $||\cdot||$ is set cardinality.

Model pre-training. During the pre-training phase, all listed datasets were utilized and split into validation and training sets. We selected the model with the highest mAP@10 value on the Clotho evaluation set for further training.

Model fine-tuning. During the fine-tuning phase, we employed only the AudioCaps and Clotho training and validation splits for training and Clotho evaluation split for validation purposes. We freeze most layers of the text encoder and of the self-attention heads of the audio encoders. We apply Low-Rank Adaptation [3] to the remaining layers, so that the fine-tuning process has a small number of trainable parameters (around 500k - 1.5M for different runs, depending on the number of frozen layers and LoRA dimension).

5. SUBMISSION DESCRIPTION

Each of our submissions was an ensemble, constructed by taking a weighted average of individual model predictions (i.e. matrices of cosine similarities). For submission 1, we fit the weights automatically across around 100 output similarity matrices, using gradient descent and the InfoNCE loss calculated globally at each step, while constraining the weights to be nonnegative. In the end 22 models make a nonzero contribution. We assume that fitting these weights directly to the validation data will not result in too much overfitting because of the small number of parameters. Submission 4 was generated similarly to submission 1, but using a smaller set of models.

Submissions 2 and 3 were then created by further ensembling the predictions from submission 1 with outputs from CLAP [12], using weights of $\frac{5}{6}$ for ours and $\frac{1}{6}$ for CLAP in submission 2, and $\frac{2}{3}$ for ours and $\frac{1}{3}$ for CLAP in submission 3. However, the model and weights that we obtained for CLAP achieved almost 60% mAP@10 on the Clotho evaluation split, much higher than reported in the paper [12], strongly suggesting that the version we received was trained on our evaluation data. Therefore, we deem the evaluation results unreliable and we do not report them.

6. RESULTS

The results achieved by our best solution are presented in Table 1. Please note that for competition purposes, this split was also used for validation and checkpointing.

Table 1: Evaluation scores on Clotho v2.1

	R@1	R@5	R@10	mAP@10
1.	0.1908	0.4362	0.5793	0.2966

[1] github.com/microsoft/CLAP
[2] github.com/harritaylor/torchvggish

7. REFERENCES

[1] T. Lamort and D. Kicinski, "Take it easy: relaxing contrastive ranking loss with cider," in *Detection and Classification of Acoustic Scenes and Events*, 2022. [Online]. Available: https://dcase.community/documents/challenge2022/technical_reports/DCASE2022_Lamort_72_t6b.pdf

[2] X. Hao, Y. Zhu, S. Appalaraju, A. Zhang, W. Zhang, B. Li, and M. Li, "Mixgen: A new multi-modal data augmentation," 2023. [Online]. Available: https://arxiv.org/abs/2206.08358

[3] E. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021.

[4] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: An audio captioning dataset," 2019. [Online]. Available: https://arxiv.org/abs/1910.09387

[5] C. D. Kim, B. Kim, H. Lee, and G. Kim, "Audiocaps: Generating captions for audios in the wild," in *NAACL-HLT*, 2019.

[6] A. S. Koepke, A.-M. Oncescu, J. Henriques, Z. Akata, and S. Albanie, "Audio retrieval with natural language queries: A benchmark study," in *IEEE Transactions on Multimedia*, 2022.

[7] I. M. Morato and A. Mesaros, "Macs - multi-annotator captioned soundscapes," July 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5114771

[8] "Freesound." [Online]. Available: https://freesound.org/

[9] M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," 2017. [Online]. Available: https://spacy.io/

[10] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," 2019. [Online]. Available: https://arxiv.org/abs/1908.10084

[11] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu, "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization," 2019.

[12] B. Elizalde, S. Deshmukh, M. A. Ismail, and H. Wang, "Clap: Learning audio concepts from natural language supervision," in *International Conference on Acoustics, Speech and Signal Processing*. IEEE, June 2022. [Online]. Available: https://www.microsoft.com/en-us/research/publication/clap-learning-audio-concepts-from-natural-language-supervision/

[13] S. Chen, Y. Wu, C. Wang, S. Liu, D. Tompkins, Z. Chen, and F. Wei, "Beats: Audio pre-training with acoustic tokenizers," 2022. [Online]. Available: https://arxiv.org/abs/2212.09058

[14] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "Cnn architectures for large-scale audio classification," 2016. [Online]. Available: https://arxiv.org/abs/1609.09430