Language-based audio retrieval with textual embeddings of tag names

Thomas Pellegrini

LANGUAGE-BASED AUDIO RETRIEVAL WITH TEXTUAL EMBEDDINGS OF TAG NAMES

Thomas Pellegrini

IRIT, Université de Toulouse, CNRS, Toulouse INP, UT3, Toulouse, France
Artificial and Natural Intelligence Toulouse Institute (ANITI)
thomas.pellegrini@irit.fr

ABSTRACT

Language-based audio retrieval aims to retrieve audio recordings based on a queried caption, formulated as a free-form sentence written in natural language. To perform this task, a system is expected to project both modalities (text and audio) onto the same subspace, where they can be compared in terms of a distance. In this work, we propose a first system based on large scale pretrained models to extract audio and text embeddings. As audio embeddings, we use logits predicted over the set of 527 AudioSet tag categories, instead of the most commonly used 2-d feature maps extracted from earlier layers in a deep neural network. We improved this system by adding information from audio tag text embeddings. Experiments were conducted on Clotho v2. A 0.234 mean average precision at top 10 (mAP@10) was obtained on the development-testing split when using the tags, compared to 0.229 without. We also present experiments to justify our architectural design choices[1].

Index Terms— Language-based audio retrieval, Pre-trained representations, PaSST, MPnet, Transformers, AudioSet tags

1. INTRODUCTION

The ability to retrieve audio recordings from a text query can be very useful when searching for recordings in a large audio database. Language-based audio retrieval systems rank audio samples according to their match with a queried caption.

Recent systems are usually comprised of two neural networks: a text encoder, responsible for encoding a concise and dense representation of a textual query, and an audio encoder that encodes the audio content of the candidate recordings [1, 2]. Both encoders are expected to provide representations embedded in the same subspace, where the textual and audio samples can be compared in terms of a distance.

In this paper, we present a baseline system that follows this architecture, based on two open-sourced pre-trained models to extract text and audio embeddings. Our main innovation is two-fold: i) we use logits as basic audio embeddings [3], instead of the more usual 2-d feature maps extracted from a hidden layer of a pretrained deep neural network [4, 5, 6], ii) we improve our baseline system by incorporating information from tags predicted on the audio recordings. We propose to combine the basic audio logit embeddings with the textual embeddings of the tag names.

Thanks for funding by the French ANR agency (LUDAU, ANR-18-CE23-0005-01) and "Investing for the Future — PIA3" AI Interdisciplinary Institute ANITI (ANR-19-PI3A-0004).

[1]PyTorch code available: `https://github.com/topel/my-audio-retrieval-dcase2022`

After describing our system architectures and experimental setup, we report results obtained on Clotho v2 [7]. We then discuss our architecture design choices in depth.

2. SYSTEM DESCRIPTION

Our system architecture is shown in Fig. 1, with the audio encoder on the right part of the figure, and the caption encoder on the left part. We used two open-sourced pretrained transformers as encoders. Both were "frozen": the embeddings were extracted only once offline and used as input to our systems. The learnable part of the model corresponds to a single linear layer that projects the audio embeddings onto the caption embedding subspace.

2.1. Text encoder: sentence embeddings with MPnet

The textual queries (captions) are encoded as sentence embeddings, obtained by averaging the word embeddings outputted by a sentence transformer. The all-mpnet-base-v2 model [8], further referred to as MPnet, was used to extract these embeddings $\vec{e}_s$, which are 768-dimensional ℓ_2-normalized vectors. MPnet is a transformer of more than 109 M parameters, trained on over a billion pairs of sentences. It was chosen since it was reported as the best model for sentence embedding against a selected list of other models [9]. We apply Layer normalization [10], followed by ℓ_2-normalization is applied to the embeddings to obtain the final ones, named $\vec{e}_q$:

$$\vec{e}_q = \ell_2\text{-norm}\left(\text{LayerNorm}\left(\vec{e}_s\right)\right) \quad (1)$$

Layer normalization (*LayerNorm*, LN) is used to normalize the samples of a minibatch independently, with adaptive per-dimension bias and gain (scale). We found that using those led to slight overfitting, thus, we removed this option. Therefore, our text encoder does not have any learnable parameter.

2.2. Audio encoder: AudioSet logit embeddings with PaSST

In the proposed system, an audio recording is encoded as a single vector, a so-called scene embedding, using the Patchout transformer named PaSST [11], pretrained on AudioSet [12]. We use the "logits" embedding outputted by PaSST, which is a 527-dimensional dense vector, comprised of the logits predicted for the 527 AudioSet event tag classes. We denote this embedding $\vec{e}_l$.

Layer-norm is then used, in this case with adaptive bias and gain, followed by a linear layer and a final ℓ_2-normalization. The final embedding is denoted $\vec{e}_a$. We can summarize these steps in Eq. 2:

$$\vec{e}_a = \ell_2\text{-norm}\left(\text{Linear}\left(\text{LayerNorm}\left(\vec{e}_l\right)\right)\right) \quad (2)$$

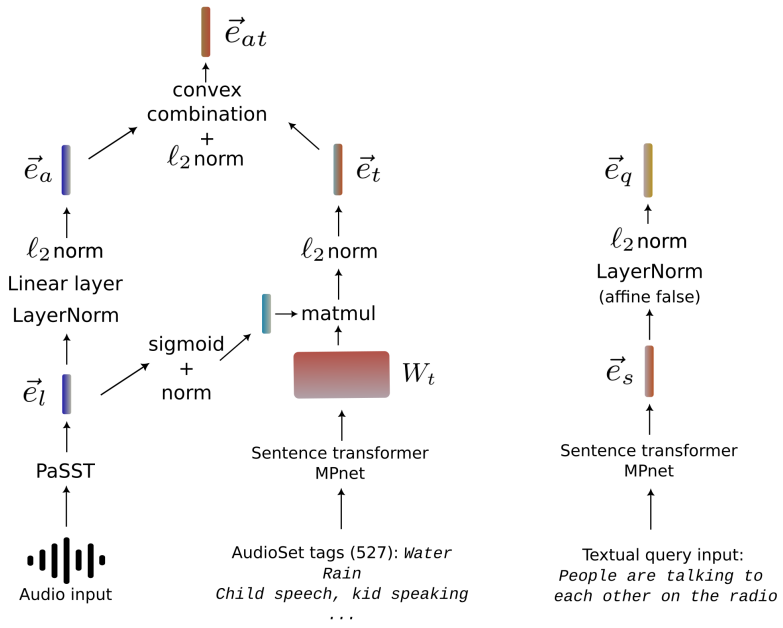

Figure 1: System architecture. Left: audio encoder (with and without tag embeddings), right: text encoder. $\vec{e}_l$: logit embedding, $\vec{e}_a$: audio embedding, $\vec{e}_t$: weighted AudioSet tag embedding, $\vec{e}_{at}$: combined audio and tag embedding, $\vec{e}_s$: sentence embedding, $\vec{e}_q$: textual query embedding, W_t: matrix of the AudioSet tag name embeddings.

We will, here-after, refer to our baseline system using PaSST and MPnet as PaSST-MPnet.

2.3. Augmented audio encoder: tag embeddings with MPnet

Adding information from tags and audio event predictions was shown to help audio captioning systems [13, 14, 15]. It should also help in our retrieval task. In this work, we tried to add such information within the audio encoder, by using MPnet to encode tag names as text embeddings. This tag name encoder is shown in Fig. 1, on the left of the audio encoding branch.

AudioSet includes 527 audio event classes, with names that can be single words, *e.g.*, "Water", "Rain" or a sequence of words, such as "Child speech, kid speaking". We precomputed 527 tag name embeddings, extracted with MPnet.

For a given audio file, we use the PaSST logits $\vec{e}_l$ to weight each of the 527 embeddings, sum them to obtain a single tag embedding $\vec{e}_t$, according the following equations. First, the PaSST logit embedding is passed through a sigmoid function to obtain multilabel probabilities $\vec{p}$, then we compute the weighted sum of the precomputed tag embeddings:

$$\vec{p} = \text{sigmoid}\,(\vec{e}_l) \tag{3}$$

$$\vec{e}_t = \ell_2\text{-norm}\left(\frac{\vec{p}}{\sum_c p_c} W_t\right) \tag{4}$$

where $W_t \in \mathbb{R}^{527\times d}$ is the matrix containing the 527 embeddings of the AudioSet tag names. d is the dimension of the embeddings provided by MPnet ($d = 768$). c is an index over the 527 tag categories.

After an ℓ_2-normalization, $\vec{e}_t$ is combined to $\vec{e}_a$ with a convex linear summation:

$$\vec{e}_{at} = \lambda\,\vec{e}_a + (1-\lambda)\,\vec{e}_t \tag{5}$$

where $\lambda \in [0, 1]$ is a weight tuned on a validation subset. The final audio embedding $\vec{e}_{at}$ is ℓ_2-normalized before being compared to the caption query embedding:

$$\vec{e}_{at} = \ell_2\text{-norm}\,(\vec{e}_{at}) \tag{6}$$

The learnable parameters of the proposed system correspond to the audio encoder ones: in the linear layer and in the element-wise adaptive biases and gains of LN, with a total of 407 k parameters. If we take into account the large-scale transformers, this number increases up to 196M parameters, but as previously mentioned, we did not fine-tune the transformers.

2.4. Training objective

We used the same loss and similarity scoring functions than in the challenge baseline system [2]. Similarity between audio and caption embeddings is estimated by taking the dot product between their embeddings, and all our systems were trained with a sampling-based triplet loss [16]. We would like a model to provide a similarity score S^p high for positive audio-query pairs, and low for either an audio embedding not paired with the caption query (audio impostor score S^n_a), or for a caption not corresponding to the audio embedding (query impostor score S^n_q). The audio and query impostors are randomly selected from the minibatch of samples being processed. The loss function as a function of the network learnable parameters θ is the sum of the contribution of these three scores:

System	# params	Development-testing split mAP@10	R@1	R@5	R@10	Evaluation dataset mAP@10	R@1	R@5	R@10
Challenge baseline	732k	0.068	0.032	0.109	0.188	0.061	0.026	0.102	0.176
CNN-transformer [5]	195M	0.260	0.150	0.400	0.530	0.251	0.153	0.387	0.504
PaSST-MPnet	196M	0.229	0.134	0.355	0.482	0.212	0.124	0.319	0.448
PaSST-MPnet-tags	196M	0.234	0.138	0.364	0.485	0.214	**0.128**	**0.332**	0.445
PaSST-MPnet-tags-AC	196M	0.240	0.145	0.365	**0.500**	0.213	0.125	0.322	0.454
Ensemble (2 models)	196M	**0.243**	**0.148**	**0.369**	0.498	**0.216**	0.127	0.321	**0.463**

Table 1: Results on Clotho v2.

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{\text{batch}} \max(0, S_a^n - S^p + \eta) + \max(0, S_q^n - S^p + \eta) \quad (7)$$

where N is the number of samples in the minibatch, η a margin.

2.5. Evaluation

Once a system is trained, retrieval is performed by scoring the similarity between the caption embedding and the audio embeddings of all the candidate audio files. The same scoring function as during training is used: the dot product. Following the DCASE 2022 challenge requirements, our systems retrieve ten audio files for each queried caption, sorted according to their match with the query.

Standard information retrieval metrics are used to evaluate the audio retrieval models: mean average precision at top-10 (mAP@10) [17] and recall at k, the number of retrieved audio items considered for scoring (R@k with $k \in \{1, 5, 10\}$) [18]. To estimate these metrics, only a correct audio-caption pair is considered as positive, all the other audio recordings retrieved are considered negative. To rank the systems submitted to the DCASE 2022 challenge, mAP@10 was the main metric [2]. It is a rank-aware metric. For a given queried caption, the precision values are averaged at the positions where the relevant items are in the retrieved rank list. The R@k metric is rank-unaware, and corresponds to the proportion of relevant items among the top-k retrieved results. For both mAP@10 and R@k, the higher the value, the better the system.

3. EXPERIMENTAL RESULTS

3.1. Datasets

We conducted our experiments on Clotho v2 [7], with the splits defined in the DCASE 2022 challenge [19]. The development set is comprised of 3839 audio clips with 19195 reference captions for training, and 1045 audio recordings with 5225 captions for testing. We refer to this test subset as development-testing, dev-test in short. We did not use the validation set. A Clotho evaluation set was provided in the challenge, used to rank the submitted systems. It contains 1000 audio files with 1000 associated captions [2]. We also used for pretraining the training subset of AudioCaps [20], containing 46231 audio files with one reference caption per file.

3.2. Experimental setup

All our models were trained on Clotho v2 for 50 epochs with the same setup: minibatches of 128 samples, a 1e-3 initial learning rate, and the Adam optimizer. We used a reduce-on-plateau scheduler based on the Clotho validation split loss, with a 0.5 ratio and a 5-epoch patience. Smaller and larger batch sizes were tested, but 128 was found to be the best one. We used $\eta = 0.4$, and $\lambda = 0.8$ when using the tags, as will be discussed here-after. When pretraining a model on AudioCaps, the number of epochs was 100 and no learning rate scheduler was used.

3.3. Results

Table 1 shows the results of our systems, together with the ones of the baseline proposed by the DCASE 2022 Task 6b challenge organizers (based on a convolutional recurrent neural network and pretrained word embeddings), and of a CNN-transformer [5], which reached the second rank out of ten teams in the challenge, with models comprised of a number of parameters similar to ours.

First, our systems based on pretrained transformers for both audio and text encoding largely outperformed the challenge baseline, which was trained from scratch on the Clotho v2 development dataset. Relying on pretrained models is efficient in this case, where labeled training data is scarce. Second, our results are worse by about 0.02 on the evaluation dataset (eval) than on the development-testing subset (dev-test). This is probably due to the fact that model selection was performed on this subset, so overfitting might have happened. It can also be that eval is more difficult than dev-test.

Compared to our baseline model PaSST-MPnet, adding information from the tag embeddings was beneficial: PaSST-MPnet-tags reached 0.234 mAP@10 compared to 0.229, on dev-test, and 0.214 compared to 0.212 on eval.

Pretraining on AudioCaps (PaSST-MPnet-tags-AC) brought a 0.06 absolute mAP@10 improvement on dev-test, compared to PaSST-MPnet-tags. Interestingly, this gain was not observed on eval, so we cannot conclude that pretraining was beneficial or not. PaSST-MPnet-tags models trained on AudioCaps, before finetuning on Clotho, reached about 0.18 mAP@10 on Clotho dev-test.

Finally, an ensemble of two PaSST-MPnet-tags-AC models trained with different seeds, led to our best results. By ensemble, we mean that the distance scores outputted by the two models were averaged. This setting allowed us to obtain the fourth place of the challenge [19].

4. ARCHITECTURE DESIGN CHOICES

In the following, all reported experiments were conducted with PaSST-MPnet-tags models.

4.1. Influence of *LayerNorm* in the audio encoder

As described in Section 2, in our models, we use *LayerNorm* (LN) with adaptive gain and bias applied to the PaSST embeddings in the

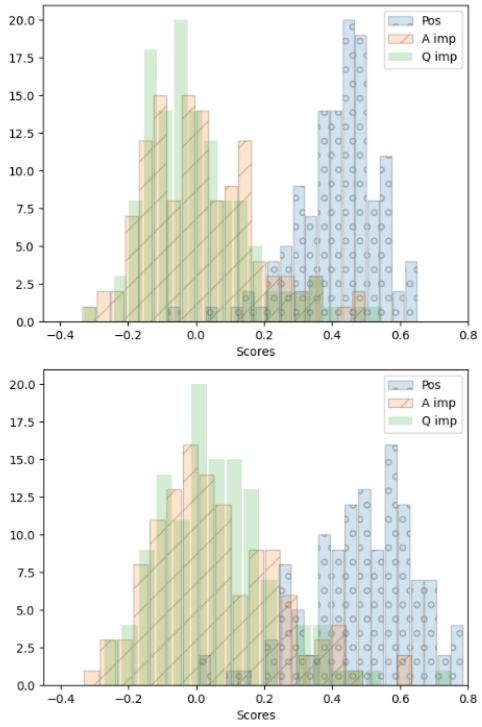

Figure 2: Similarity score distributions on the training subset when using either a 0.4 (top) or a 1.0 (bottom) value for the margin η, when training a PaSST-MPnet-tags model. Pos: positive audio-caption pair, A imp: negative audio-query pair (audio impostor), Q imp: negative query-audio pair (caption impostor).

audio encoder.

First, if we do not use LN in the audio encoder, mAP@10 and the other metrics are much worse: 0.217 mAP@10, compared to 0.234 reported in Table 1.

Second, we tried to use batch normalization (*BatchNorm*, BN) in the audio encoder, instead of LN. Using BN and keeping all the rest of the architecture unchanged (and for a given random seed), led to a 0.229 mAP@10 value, again worse than 0.234 when using LN. By observing the learning curves, BN seems to overfit a little more than LN. When using adaptive bias and scale, the number of learnable parameters is the same for both types of normalization, *i.e.* twice the dimension of the embeddings. The only possible reason why BN overfits more is because of the running means and standard deviations estimated on the training subset of data, that may not generalize so well on the test subset. These are necessary for BN but not for LN, since LN performs a per sample normalization.

4.2. Impact of the margin η

In [16], the authors used a default value of 1.0 for the margin η, used in the loss function, given in Eq. 7. In the DCASE challenge baseline, this default value was also used. In our case, the audio and queried caption embeddings are of unit norm. The results of the dot products, *i.e.,* the similarity scores, are theoretically between -1.0 and 1.0. A margin of 1.0 might be to difficult to obtain when training a model. Indeed, $\eta = 1.0$ led to a 0.225 mAP@10 on dev-test, worse than 0.234 obtained with $\eta = 0.4$. We tried several values lower than 1.0, and $\eta = 0.4$ was the best value.

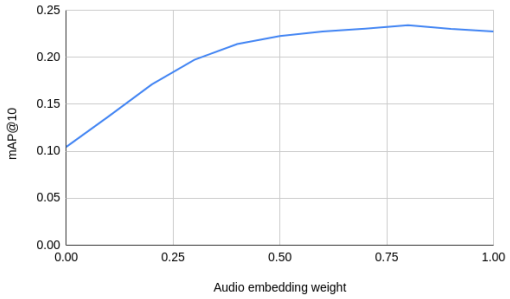

Figure 3: mAP@10 values on dev-test according to the audio embedding weight λ.

In Fig. 2, we plotted the score distributions of the positive pairs (*Pos*, a positive caption-audio pair) and negative pairs (*A imp* and *Q imp* for audio and query impostors, resp.), on the training subset. The distributions on the dev-test subset are very similar. The difference between the top and the bottom plots is the value of the margin: $\eta = 0.4$ (top figure) and $\eta = 1.0$ (bottom figure). As can be seen, the support interval of the scores is smaller with $\eta = 0.4$ than with $\eta = 1.0$: the positive pair scores reach about 0.62 maximum with $\eta = 0.4$, and about 0.8 with $\eta = 1.0$. Nevertheless, we can see more overlap with impostor scores in the case of $\eta = 1.0$, which mean that more mistakes are made by this model.

4.3. Influence of the audio/tag weight λ

Figure 3 shows the mAP@10 values on dev-test, when varying λ from Eq. 5. When $\lambda = 0$, the audio encoder provides as output the tag embeddings $\vec{e}_t$, and the model has no learnable parameters at all. In this setting, a 0.105 mAP@10 is obtained. This shows that using the tag name embeddings weighted with their probabilities already brings some information. When $\lambda = 1$, the audio encoder provides as output the projected PaSST embeddings $\vec{e}_a$, and the model corresponds to PaSST-MPnet, which reached 0.229 mAP@10. In between, the score varies, and the best value was obtained with $\lambda = 0.8$.

5. CONCLUSION

We reported text-based audio retrieval experiments, with systems based on pretrained audio and sentence large scale transformers. In our baseline system, we used logits as audio embeddings, instead of the most commonly used 2-d feature maps extracted from earlier layers in a deep neural network. We improved this system by adding information from AudioSet tags, encoded as sentence embeddings. Finally, we discussed some of our architecture design choices: the use of layer normalization, a value smaller than 1.0 for the margin in the contrastive loss function for learning, and the 0.8 value for weighting the audio embeddings when adding the tag contribution within the audio encoder.

As a short-term perspective, we would like to use logit embeddings together with more standard 2-d feature maps in the audio encoder, to see whether those are complementary [5]. We also would like to investigate the use of external data, either for pretraining models or in a semi-supervised learning setting, where existing algorithms could be adapted to the present task [21].

6. REFERENCES

[1] H. Xie, O. Räsänen, K. Drossos, and T. Virtanen, "Unsupervised audio-caption aligning learns correspondences between individual sound events and textual phrases," in *Proc. ICASSP*. IEEE, 2022, pp. 8867–8871.

[2] H. Xie, S. Lipping, and T. Virtanen, "Dcase 2022 challenge task 6b: Language-based audio retrieval," *arXiv preprint arXiv:2206.06108*, 2022.

[3] E. Tzinis, S. Wisdom, J. R. Hershey, A. Jansen, and D. P. Ellis, "Improving universal sound separation using sound classification," in *Proc. ICASSP*. IEEE, 2020, pp. 96–100.

[4] 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," DCASE2022 Challenge, Tech. Rep., July 2022.

[5] X. Mei, X. Liu, H. Liu, J. Sun, M. D. Plumbley, and W. Wang, "Language-based audio retrieval with pre-trained models," DCASE2022 Challenge, Tech. Rep., July 2022.

[6] T. L. de Gail and D. Kicinski, "Take it easy: Relaxing contrastive ranking loss with CIDEr," DCASE2022 Challenge, Tech. Rep., July 2022.

[7] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: An audio captioning dataset," in *Proc. ICASSP*. IEEE, 2020, pp. 736–740.

[8] https://huggingface.co/sentence-transformers/all-mpnet-base-v2.

[9] https://www.sbert.net/docs/pretrained_models.html.

[10] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[11] K. Koutini, J. Schlüter, H. Eghbal-zadeh, and G. Widmer, "Efficient training of audio transformers with patchout," *arXiv preprint arXiv:2110.05069*, 2021.

[12] 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*, New Orleans, LA, 2017.

[13] Z. Ye, H. Wang, D. Yang, and Y. Zou, "Improving the performance of automated audio captioning via integrating the acoustic and textual information," DCASE2021 Challenge, Tech. Rep., July 2021.

[14] F. Gontier, R. Serizel, and C. Cerisara, "Automated audio captioning by fine-tuning bart with audioset tags," in *Proc. Workshop DCASE*, 2021, pp. 170–174.

[15] Q. Han, W. Yuan, D. Liu, X. Li, and Z. Yang, "Automated audio captioning with weakly supervised pre-training and word selection methods," in *Proc. Workshop DCASE*, Barcelona, Spain, November 2021, pp. 6–10.

[16] D. Harwath, A. Torralba, and J. Glass, "Unsupervised learning of spoken language with visual context," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://Proc..neurips.cc/paper/2016/file/82b8a3434904411a9fdc43ca87cee70c-Paper.pdf

[17] P. Kaur, H. S. Pannu, and A. K. Malhi, "Comparative analysis on cross-modal information retrieval: A review," *Computer Science Review*, vol. 39, p. 100336, 2021.

[18] A. S. Koepke, A.-M. Oncescu, J. Henriques, Z. Akata, and S. Albanie, "Audio retrieval with natural language queries: A benchmark study," *IEEE Trans. on Multimedia*, pp. 1–11, 2022.

[19] http://dcase.community/challenge2022/.

[20] C. D. Kim, B. Kim, H. Lee, and G. Kim, "Audiocaps: Generating captions for audios in the wild," in *Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 119–132.

[21] L. Cances, E. Labbé, and T. Pellegrini, "Comparison of semi-supervised deep learning algorithms for audio classification," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2022, no. 23, Sept. 2022. [Online]. Available: https://doi.org/10.1186/s13636-022-00255-6