DCASE 2023 TASK 6 AUTOMATED AUDIO CAPTIONING AND LANGUAGE-BASED RETRIEVAL

Technical Report

Greeshma Karanth, Ninaad Rao, Srikumar Subramanian, Ankit Shah

Carnegie Mellon University
Language Technologies Institute
Pittsburgh, PA, USA,
grk, ninaadrr, srikumas, aps1@cs.cmu.edu

ABSTRACT

The objective of this project is to examine audio signals utilizing natural language to capture their complex characteristics. This initiative is part of Task 6 in the DCASE 2023 Competition and consists of two subtasks. The first subtask is Automated Audio Captioning, which generates text descriptions of audio content. This task involves the intermodal processing of an audio signal as input and a text description as output. Our best-performing model for this uses the PANN architecture [1] with the CNN-14 feature extractor and BART [2] encoder and decoder. The second subtask is called Language-Based Audio Retrieval, where the system retrieves audio signals by searching for their sound content descriptions. The queries for this subtask are human-generated audio captions. In this task, our best-performing model uses CLAP [3] audio embeddings and Roberta text embeddings [4]. This document presents a summary of our work done for this challenge.

Index Terms— CLAP embeddings, RoBERTa text embeddings, audio captioning, PANNs

1. INTRODUCTION

1.1. Automated Audio Captioning

Automated audio captioning, or AAC, is a technology that generates textual descriptions of audio signals using natural language. Unlike speech-to-text transcription, AAC is an inter-modal translation task that captures not only the physical characteristics of sounds but also abstract concepts and high-level knowledge. This allows for a wide range of applications, from automatic content descriptions to sophisticated machine-to-machine interactions focused on content.

Task 6 of the DCASE 2023 Challenge builds on the previous year's challenge and aims to explore how machines can understand higher-level and human-perceived information from general sounds. The challenge requires participants to develop AAC models that accurately generate natural language descriptions of audio signals and understand the semantic meaning of sounds in various scenarios. The submissions will be evaluated based on a combination of SPIDEr and a fluency error detection model, along with traditional metrics such as METEOR, CIDEr, and SPICE. The ultimate goal of this challenge is to create AAC models that can improve accessibility for people with hearing disabilities and transform the way we interact with audio content.

1.2. Language Based Audio Retrieval

The subtask on language-based audio retrieval at the DCASE 2023 Challenge is an extension of the previous year's task, aimed at addressing the limitations of binary relevance assessment for audio files. The objective of this subtask is to retrieve relevant audio files from a dataset based on their sound content textual descriptions, or audio captions, which are provided by humans. The goal is to develop methods that can rank audio signals in a fixed dataset based on their match to the given description, allowing for more sophisticated machine-to-human interactions with a focus on content. This subtask is of great importance as it helps to bridge the gap between the human perception of sounds and the ability of machines to understand and retrieve relevant audio files based on that perception.

The 2023 Challenge introduces graded relevance scores, which are crowdsourced to allow for a more nuanced assessment of the relevance of audio files. Mean Average Precision (mAP)@10 is used as a primary metric and recall@k (including recall@1, recall@5, and recall@10) serve as secondary metrics. These metrics are crucial in measuring the effectiveness of the language-based audio retrieval methods developed by participants in the challenge.

By emphasizing the importance of language-based audio retrieval, this subtask encourages further exploration and innovation in the field of sound processing and analysis. It provides a platform for researchers and practitioners to develop and refine methods that can bridge the gap between human perception and machine understanding of sound content, with potential applications in a wide range of fields, including multimedia content creation, search and retrieval, and human-machine interaction.

2. DATASET

Clotho v2 is an audio dataset that extends the original Clotho dataset. It comprises 6972 audio samples, each with five captions, and is built with a focus on audio content and caption diversity. The dataset is divided into four splits: development, validation, evaluation, and testing, with audio samples publicly available for all four splits, but captions available only for the development, validation, and evaluation splits. The audio samples have durations ranging from 10s to 30s, with no spelling errors and good quality. The content of the audio samples is diverse, including ambiance, animal sounds, crowd noises, machines and engines, and devices. The splits are created using multi-label stratification and ensure that there is no overlap in audio samples or words between the different

splits. The dataset is suitable for training and evaluating methods for audio captioning and other related tasks. In total, there are 6972 audio clips and 34,860 captions.

VGGish [1]	CNN6	CNN10	CNN14
Log-mel spectrogram 96 frames × 64 mel bins	Log-mel spectrogram 1000 frames × 64 mel bins		
3×3 @ 64 ReLU	5×5 @ 64 BN, ReLU	$\begin{pmatrix} 3 \times 3 \text{ @ } 64 \\ \text{BN, ReLU} \end{pmatrix} \times 2$	$\begin{pmatrix} 3 \times 3 \text{ @ } 64 \\ \text{BN, ReLU} \end{pmatrix} \times 2$
MP 2×2	Pooling 2×2		
3×3 @ 128 ReLU	5×5 @ 128 BN, ReLU	$\begin{pmatrix} 3 \times 3 \text{ @ } 128 \\ \text{BN, ReLU} \end{pmatrix} \times 2$	$\begin{pmatrix} 3 \times 3 \text{ @ } 128 \\ \text{BN, ReLU} \end{pmatrix} \times 2$
MP 2×2	Pooling 2×2		
$\begin{pmatrix} 3 \times 3 \text{ @ } 256 \\ \text{ReLU} \end{pmatrix} \times 2$	5×5 @ 256 BN, ReLU	$\begin{pmatrix} 3 \times 3 \text{ @ } 256 \\ \text{BN, ReLU} \end{pmatrix} \times 2$	$\begin{pmatrix} 3 \times 3 \text{ @ } 256 \\ \text{BN, ReLU} \end{pmatrix} \times 2$
MP 2×2	Pooling 2×2		
$\begin{pmatrix} 3 \times 3 \text{ @ } 512 \\ \text{ReLU} \end{pmatrix} \times 2$	5×5 @ 512 BN, ReLU	$\begin{pmatrix} 3 \times 3 \text{ @ } 512 \\ \text{BN, ReLU} \end{pmatrix} \times 2$	$\begin{pmatrix} 3 \times 3 \text{ @ } 512 \\ \text{BN, ReLU} \end{pmatrix} \times 2$
MP 2×2 Flatten	Global pooling		Pooling 2×2
FC 4096 ReLU $\times 2$	FC 512, ReLU		$\begin{pmatrix} 3 \times 3 \text{ @ } 1024 \\ \text{BN, ReLU} \end{pmatrix} \times 2$
FC 527, Sigmoid	FC 527, Sigmoid		Pooling 2×2
			$\begin{pmatrix} 3 \times 3 \text{ @ } 2048 \\ \text{BN, ReLU} \end{pmatrix} \times 2$
			Global pooling
			FC 2048, ReLU
			FC 527, Sigmoid

Figure 1: PANN architecture

3. SYSTEM DESCRIPTION

Both of the tasks in this project have a state-of-the-art baseline system available for those participating in the challenge, provided as a starting point to build better solutions.

The baseline system for the first task of Automated Audio Captioning is a sequence-to-sequence model implemented using PyTorch. The system is divided into four parts:

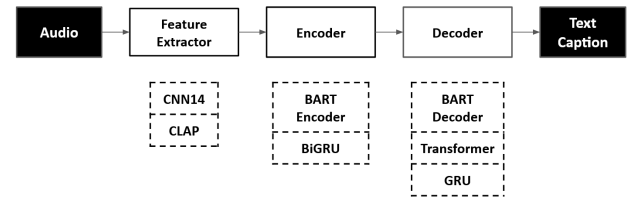

Figure 2: System overview of Task 6a

- Caption Evaluation
- Dataset Pre-Processing/feature extraction
- Encoder that takes the pre-trained audio embeddings as input.
- Decoder that returns the audio captions

Caption evaluation is performed using a modified version of the caption evaluation tools from the MS COCO challenge.

For dataset pre-processing, the Clotho dataset examples are available as WAV and CSV files, and features have to be extracted from the audio clips, and captions in CSV files have to be pre-processed, such as removing punctuation. The extracted features and processed words are then matched as input-output pairs.

In addition to the baseline system provided by the challenge organizers, we also use the top submission of the previous iteration as a baseline to compare against. The top submission for the task of Automated Audio Captioning is an audio encoder taken after training the bi-encoder with text and audio encoders for the task of audio retrieval. This feature extractor is used with a sequence to sequence model trained on Clotho. In our tests, we used a CNN 14 audio encoder for this baseline.

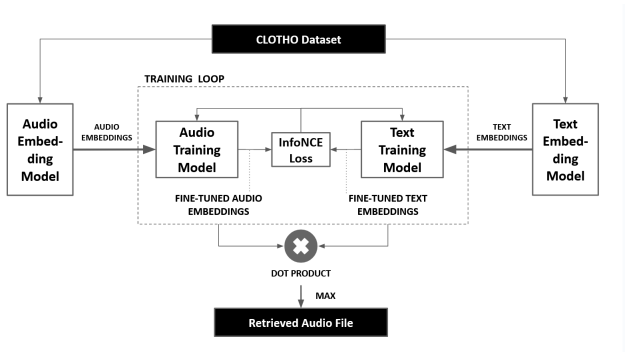

Figure 3: System overview of Task 6b

The baseline system for the Audio Retrieval System uses a Pre-trained Audio Neural Network [1] with 10-layer CNN architecture that is pre-trained on AudioSet dataset and fine-tuned on Clotho Dataset. The text encoder used in the baseline is the all-mpnet-base-v2(Sentence-BERT) model. The text encoder is frozen during training and only the PANN architecture is trained.

3.1. Model Description

For the first task, the input can be denoted as $X \in R^{T \times N_{features}}$ where T is the number of frames and $N_{features}$ is the number of features. The output can be denoted as $Y \in R^{I \times L_{words}}$ where I is the number of captions and L_{words} is the length of words.

The baseline system uses a BART encoder and BART decoder as the text caption module. The encoder is a fully-convolutional network (PANNs CNN14) as described in Fig. 2 trained as part of the baseline of task 6b on audio retrieval, explained further in this report. The final temporal average and dense layers are omitted to obtain sequences of audio embeddings with a dimension of 2048. These embeddings are then transformed frame-wise by an affine layer, reducing the dimension to 768, and fed to the decoder.

The decoder uses encoder outputs to generate the caption in an autoregressive manner. Previously generated words are tokenized and transformed into embeddings as inputs to the decoder. In the baseline model, the tokenizer is pre-trained with a byte-pair-encoding process, where each token corresponds to a sub-word from the model vocabulary instead of a full English word. Each token in the past sequence is associated with a feature vector through an embedding map and input to the decoder. Each layer of the decoder attends on the previously generated tokens with self-attention, as well as to the entire encoder output sequence with cross-attention. A classifier head consisting of one linear layer with softmax activation outputs a probability for each token of the vocabulary.

The system overview for the audio captioning pipeline is presented in Figure 3. Raw audio files are first processed by a feature extractor, and then the resulting features are passed to an encoder. The encoder generates a representation of the audio, which is then passed to a decoder to produce the corresponding text caption for the audio. This process enables the system to automatically gener-

ate captions for audio files, making it useful for tasks such as video indexing, content retrieval, and accessibility for people with hearing impairments.

The audio retrieval task matches audio files with relevant text captions using a dual encoder architecture that encodes both modalities. This cross-modal architecture is trained using joint loss metrics to improve system performance. Figure 3 provides an overview of the system for audio retrieval. In training, the pre-trained audio and text embeddings are fine-tuned using the InfoNCE loss function, which is used to learn the relative order of audio-text pairs. During retrieval, the dot product is used to calculate the similarity between audio and text pairs to identify the top matches for the caption. By using these methods, the accuracy of the audio retrieval task is enhanced, and the model can effectively match audio with relevant text captions.

3.2. Metrics

3.2.1. Automated Audio Captioning

For task 6a, there are five different metrics used as defined below.

- BLEU1, BLEU2, BLEU3: measures the similarity between the generated text and the reference texts based on the weighted geometric mean of n-grams of different lengths
- ROUGEl: measures the similarity between the generated text and the reference texts based on F-measures computed by counting the longest common subsequence
- METEOR: measures the similarity between the generated text and the reference texts based on the harmonic mean of precision and recall using explicit word-to-word matches
- CIDEr: measures the similarity between the generated text and the reference texts based on the cosine similarity between term frequency and inverse document frequency
- SPICE: measures the similarity between the generated text and the reference texts based on F-score calculated using tuples in scene graphs created for the captions
- SPIDEr: a metric that combines the semantic stability of SPICE and the fluency of CIDEr and uses Monte Carlo rollouts for optimization

3.2.2. Language-Based Audio Retrieval

For task 6b, there are four different metrics and they are defined below.

- R@1: This metric finds recall score among the top-1 retrieved result, averaged across all caption queries
- R@5: This metric finds recall score among the top-5 retrieved result, averaged across all caption queries
- R@10: This metric finds recall score among the top-5 retrieved result, averaged across all caption queries
- mAP@10:This metric computes the average precision among the top-10 retrieved results, averaged across all caption queries

3.3. Loss functions

3.3.1. Cross Entropy Loss

For the first task, a cross-entropy loss is used to train the model, where y_t is the ground truth word at time step t, as shown in equation 1.

$$L_{CE} = -\frac{1}{T}\sum_{t=1}^{T}\log(p(y_t|y_{1:t-1},\theta)) \tag{1}$$

3.3.2. InfoNCE loss

For the second task i.e Task 6b audio, the baseline system uses the InfoNCE loss function. NCE stands for Noise-Contrastive Estimation, and this is a type of contrastive loss function that is generally used for self-supervised learning. This loss is used to learn a meaningful representation of the data. Given a set X of n random samples,

$$X = \{x_1, \ldots, x_N\} \tag{2}$$

containing one positive sample from $p(x_{t+k}|c_t)$ and N-1 negative samples from the 'proposal' distribution $p(x_{t+k})$, the loss function can be written as

$$\mathcal{L}_{\mathrm{N}} = -E\left[\log\frac{f_k\left(x_{t+k}, c_t\right)}{\sum_{x_j\in X} f_k\left(x_j, c_t\right)}\right] \tag{3}$$

Optimizing this loss results in $f_k(x_{t+k}, c_t)$ which estimates the density ratio as

$$f_k\left(x_{t+k}, c_t\right) \propto \frac{p\left(x_{t+k} \mid c_t\right)}{p\left(x_{t+k}\right)} \tag{4}$$

3.4. Experimental Settings

The baseline system uses specific hyperparameters for feature extraction and caption processing, including a log-mel spectrogram with 64 mel bands, removal of punctuation, lowercase letters, tokenization, pre-pending of a start-of-sentence token, and appending of an end-of-sentence token. The deep neural network used in the baseline system has specific hyperparameters, including the CNN14 model trained on audio retrieval, an affine layer, six transformer decoder layers, and an affine layer with softmax activation. Parameter optimization is performed using AdamW on the cross-entropy loss for 20 epochs with a batch size of 4 examples and gradient accumulation of 2 steps, and the learning rate is 10-5.

All input audio features and captions in a batch are padded to a fixed length for convenient batching, with attention masks provided to the model to ignore padded elements. The baseline system also provides options for beam search and greedy decoding during evaluation. The entire training process takes approximately 2 hours on a single GPU (GTX 1080ti). The evaluation and inference process on the given datasets takes 5 minutes on a T4 GPU.

Another baseline implementation corresponding to the top submission of the previous iteration of the challenge was run. Here, the audio encoder used was CNN14 model trained on audio retrieval. The whole captioning model except the feature extractor is trained for 25 epochs with a batch size of 64 using the Adam optimizer.

4. EXPERIMENTS

4.1. Automated Audio Captioning

Three components of the pipeline which are the feature extractor, encoder, and decoder were altered for experimentation. For the feature extractor, CNN14 and CLAP were used. For the encoder, the

networks experimented with were BART and a Bi-directional GRU [5].

For this task, the baseline system uses a pre-trained CNN14 audio feature extractor taken from training on task 6B. The system also uses BART for the encoder and decoder.

The audio feature extractor for our experiments was frozen during training as it was taken from task 6 B and only the encoder and decoder are trained.

4.2. Language-Based Audio Retrieval

For Task 6b, The baseline system uses a dot product to find the similarity between an audio signal and a textual description. To optimize the system, InfoNCE loss is used. Each epoch takes 6 minutes and evaluation since the retrieval system needs to build the score for every pair and that's a $O(n^2)$ operation, resulting in the entire scoring taking about 5 hours.

Moreover, several Audio and Text embeddings were tried.

4.2.1. Audio Embeddings

Pretrained Audio Neural Network (PANN) - PANN Embeddings [1] is a fully-convolutional network based on 14 layered CNN networks. This is the one given in the baseline

OpenL3 Embeddings - OpenL3 embeddings [6] is a self-supervised learning approach that is trained on audio-visual correspondence in videos. This embedding can work for many downstream tasks such as audio classification.

Audio Spectrogram Transformer (AST) - AST [7] is convolution-free approach mainly used for audio classification. This is an attention-based model for audio classification.

4.2.2. Text Embeddings

SBERT Embeddings - SBERT embeddings [8] utilizes siamese and triplet network structures to get similar and meaningful sentence embeddings and compare them using cosine similarity.

RoBERTa Embeddings - RoBERTa embeddings [4] is a BERT-based model that is trained on a large corpus of data. The main difference of RoBERTa is removing the next-sentence pretraining objective and training with a larger mini-batch and learning rate.

DistilBERT Embeddings - DistilBERT embeddings [9] is a compressed and lighter version of the BERT language model, which retains most of its performance while requiring fewer computational resources.

XLNet Embeddings - XLNet embeddings [10] is a language model that leverages an autoregressive formulation in combination with permutation-based training to achieve state-of-the-art results on a range of natural language processing tasks.

MPNet-MultiQA Embeddings - MPNet-MultiQA embeddings [10] utilize a multitask learning approach to jointly train a transformer-based language model on multiple question-answering tasks, resulting in improved performance across tasks.

5. RESULTS

5.1. Automated Audio Captioning

The results of this task are shown in Table 4.1. From here, it can be seen that the model that performed the best was the baseline which had a CNN14 feature extractor combined with a BART encoder and a BART decoder which is a transformer encoder and decoder model.

Using CLAP as a feature extractor resulted in very poor performance across all metrics. Using a simple 2 layer transformer in the decoder did not perform as well as the best model and even using an ensemble of techniques where the decoder was different (transformers and GRU)[5] in the decoder did not improve performance significantly.

In all these cases, the audio encoder was trained on the Language-Based Audio Retrieval task and was frozen for the audio captioning task.

5.2. Language-Based Audio Retrieval

For this task, the model creates a database of audio and matches it with the text description. This also helps generate all the metrics such as mAP, R1,R5, and R10. Table 4.2.1 shows the results for the different experiments. From the table, it can be seen that CLAP embeddings along with RoBERTa performed the best compared to all the other models. This can be mainly attributed to the way CLAP embeddings were trained. CLAP embeddings learn acoustic concepts from natural language supervision which is relevant to our task. For the text embeddings, since RoBERTa is trained on huge data, it is able to better represent the text description.

6. CONCLUSION

This project has familiarized us with the domain of automated audio captioning and language-based audio retrieval. We were able to gain a deeper understanding of each task and understand how audio features are correlated with text features to either generate relevant captions for given audio clips or retrieve relevant audio clips given a text query. We observed that the best-performing model for language-based audio retrieval uses CLAP audio embeddings with RoBERTa text embeddings, while a PANN CNN-14 model with BART based encoder and decoder works best to generate captions for audio clips.

Going forward, more extensive exploration and evaluations can help converge to an improved solution. We can modify the architecture of the encoder and decoder in the first task, or the feature architecture model, to generate better captions.

7. ACKNOWLEDGMENT

We would like to thank our mentors Ankit Shah and Professor Bhiksha Ramakrishnan for encouraging us to participate in this challenge and motivating us to make this submission. We learned a lot in this project due to their constant guidance and support.

8. REFERENCES

[1] 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," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[2] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

Table 1: Results for Audio Captioning Task (6a)

Feature extractor	Encoder	Decoder	CIDEr	SPICE	SPIDEr	$BLEU_4$	METEOR	ROUGE
CNN14	BART	BART	0.419	0.121	0.270	0.171	0.178	0.386
CNN14	2 layer BiGRU (256)	2 layer Transformer (256)	0.386	0.120	0.253	0.155	0.171	0.370
CNN14	2 layer BiGRU (256)	2 layer Transformer (256) + 4 layer Transformer (256) + GRU (Ensemble)	0.398	0.122	0.260	0.158	0.175	0.376
CLAP	BART	BART	0.66	0.045	0.056	0.045	0.097	0.249

Table 2: Results for Audio Retrieval Task (6b)

Audio Embedding	Text Embedding	mAP	R1	R5	R10
PANN	SBERT	0.222	0.130	0.343	0.480
PANN	RoBERTa	0.219	0.125	0.341	0.470
PANN	DistilBERT	0.208	0.121	0.320	0.454
PANN	multi-qa-mpnet	0.201	0.110	0.316	0.452
PANN	XLNet	0.133	0.065	0.221	0.338
AST	RoBERTa	0.099	0.042	0.169	0.271
OpenL3	SBERT	0.101	0.044	0.171	0.269
CLAP	SBERT	0.228	0.129	0.357	0.503
CLAP	**RoBERTa**	**0.244**	**0.145**	**0.378**	**0.513**

[3] B. Elizalde, S. Deshmukh, M. A. Ismail, and H. Wang, "Clap: Learning audio concepts from natural language supervision," *arXiv preprint arXiv:2206.04769*, 2022.

[4] 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," *arXiv preprint arXiv:1907.11692*, 2019.

[5] 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.

[6] J. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, "Look, listen, and learn more: Design choices for deep audio embeddings," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3852–3856.

[7] Y. Gong, Y.-A. Chung, and J. Glass, "Ast: Audio spectrogram transformer," *arXiv preprint arXiv:2104.01778*, 2021.

[8] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[9] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," *CoRR*, vol. abs/1910.01108, 2019. [Online]. Available: http://arxiv.org/abs/1910.01108v4

[10] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, pp. 5753–5763, 2019.