Improving Audio Captioning Models with Fine-grained Audio Features, Text Embedding Supervision, and LLM Mix-up Augmentation

Wu, Shih-Lun; Chang, Xuankai; Wichern, Gordon; Jung, Jee-weon; Germain, François G; Le Roux, Jonathan; Watanabe, Shinji

TR2024-028 March 19, 2024

Abstract

Automated audio captioning (AAC) aims to generate informative descriptions for various sounds from nature and/or human activities. In recent years, AAC has quickly attracted research interest, with state- of-the-art systems now relying on a sequence-to-sequence (seq2seq) backbone powered by strong models such as Transformers. Fol- lowing the macro-trend of applied machine learning research, in this work, we strive to improve the performance of seq2seq AAC models by extensively leveraging pretrained models and large language models (LLMs). Specifically, we utilize BEATS to extract fine-grained audio features. Then, we employ INSTRUCTOR LLM to fetch text embeddings of captions, and infuse their language-modality knowledge into BEATs audio features via an auxiliary InfoNCE loss function. Moreover, we propose a novel data augmentation method that uses ChatGPT to produce caption mix-ups (i.e., grammatical and compact combinations of two captions) which, together with the corresponding audio mixtures, increase not only the amount but also the complexity and diversity of training data. During inference, we propose to employ nucleus sampling and a hybrid reranking algorithm, which has not been explored in AAC research. Combining our efforts, our model achieves a new state- of-the-art 32.6 SPIDEr-FL score on the Clotho evaluation split, and wins the 2023 DCASE AAC challenge.

IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2024

IMPROVING AUDIO CAPTIONING MODELS WITH FINE-GRAINED AUDIO FEATURES, TEXT EMBEDDING SUPERVISION, AND LLM MIX-UP AUGMENTATION

Shih-Lun Wu[1]*, Xuankai Chang*[1]*, Gordon Wichern*[2]*, Jee-weon Jung*[1]*,*
François Germain[2]*, Jonathan Le Roux*[2]*, Shinji Watanabe*[1]

[1] Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA
[2] Mitsubishi Electric Research Labs (MERL), Cambridge, MA, USA

ABSTRACT

Automated audio captioning (AAC) aims to generate informative descriptions for various sounds from nature and/or human activities. In recent years, AAC has quickly attracted research interest, with state-of-the-art systems now relying on a sequence-to-sequence (seq2seq) backbone powered by strong models such as Transformers. Following the macro-trend of applied machine learning research, in this work, we strive to improve the performance of seq2seq AAC models by extensively leveraging pretrained models and large language models (LLMs). Specifically, we utilize BEATs to extract fine-grained audio features. Then, we employ INSTRUCTOR LLM to fetch text embeddings of captions, and infuse their language-modality knowledge into BEATs audio features via an auxiliary InfoNCE loss function. Moreover, we propose a novel data augmentation method that uses ChatGPT to produce caption mix-ups (i.e., grammatical and compact combinations of two captions) which, together with the corresponding audio mixtures, increase not only the amount but also the complexity and diversity of training data. During inference, we propose to employ nucleus sampling and a hybrid reranking algorithm, which has not been explored in AAC research. Combining our efforts, our model achieves a new state-of-the-art 32.6 SPIDEr-FL score on the Clotho evaluation split, and wins the 2023 DCASE AAC challenge.

Index Terms— AAC, BEATs, LLM, mix-up, InfoNCE

1. INTRODUCTION

Automated audio captioning (AAC) is a multimodal task whose goal is to describe an input audio clip using text. The descriptions are not restricted to a fixed set of class labels or tags, but are free-form sentences [1], which allow better flexibility and expressivity. Research progress on AAC has accelerated in recent years thanks to the yearly DCASE challenges, the impressive performance of Transformer-based language models, and the release of the open audio captioning datasets Clotho [2] and AudioCaps [3]. Recent leading works in AAC [4–9] all used the sequence-to-sequence (seq2seq) modeling framework, where an audio encoder is used to extract features from the input audio, and a text decoder learns to generate the caption autoregressively based on the extracted audio features.

While our work is also seq2seq-based, we extensively leverage machine learning models that are pretrained on large-scale datasets to improve AAC performance from multiple aspects, namely, audio feature extraction, auxiliary training objective, and data augmentation. We begin with revamping the audio encoder (Section 2.1). While PANN [10], a convolution-based audio feature extractor, has long been the tried-and-true choice for AAC research [4–6], many recently proposed audio encoders [11–13] have the potential to further improve AAC performance due to more advanced architectures, pretraining objectives, and finer-grained output features. Specifically, we choose Bidirectional Encoder representation from Audio Transformers (BEATs) [13], a state-of-the-art multi-label audio tagging model pretrained on AudioSet [14], as our audio encoder.

Next, witnessing the tremendous success of large language models (LLMs) in representing and generating text [15, 16], we use text embeddings from the INSTRUCTOR Transformer [17] to provide additional supervision (Section 2.2), and employ ChatGPT [18] to perform a novel mix-up data augmentation (Section 2.3). In previous AAC studies, to help linking audio features with concepts in text, which is the output space of AAC tasks, [5] pretrained the PANN encoder with an audio-caption InfoNCE [19] contrastive loss, while [6] used multitask learning to predict keywords in the caption. In our work, we combine the benefits of both representation learning and multitask training, and also leverage the LLM's rich knowledge of text. Particularly, we use INSTRUCTOR to obtain text embeddings for ground-truth captions, and apply an auxiliary InfoNCE loss on a Conformer [20, 21] postencoder to refine/summarize the BEATs audio features and align them with INSTRUCTOR text embeddings.

For data augmentation, other than SpecAugment [22] used commonly in audio-related tasks, researchers have leveraged the original mix-up [23] to linearly combine audio/text embeddings from two unrelated samples [8,24], synonym substitution on ground-truth captions [7], and caption concatenation [7]. As for LLM-based efforts, ChatGPT has been used to compile the large-scale WavCaps [7,8,25] AAC dataset by rewriting tags or fragmented descriptions, which are often associated with audio files on the web, to coherent sentences. Integrating the ideas of mix-up and LLM augmentation methods, we prompt ChatGPT to mix-up the captions of two audio clips, which produces more natural combined captions than simple text concatenation [7]. The text mix-ups, when paired with audio mix-ups (i.e., summations of waveforms), increase the amount, complexity, and diversity of our training data. With all the techniques above, we also discover that nucleus sampling decoding [26] followed by hybrid reranking (Section 2.4) leads to a further performance boost.

Our AAC model attains a state-of-the-art SPIDEr-FL score of 32.6 (on Clotho V2 [2] evaluation set) and is the winner of the 2023 DCASE AAC Challenge.[1] Despite the numerous components introduced to our model, we show in our ablation study (Section 3.4)

This work used *Bridges2-PSC* and *Delta-NCSA* through allocation CIS210014 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, supported by NSF grants #2138259, #2138286, #2138307, #2137603, and #2138296.

[1]DCASE Challenge 2023 technical report: https://dcase.community/documents/challenge2023/technical_reports/DCASE2023_Wu_31_t6a.pdf.

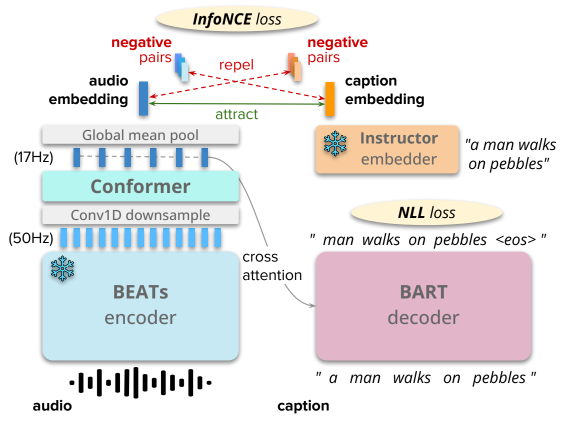

Fig. 1: Overview of our Transformer-based captioning system. We utilize a frozen BEATs [13] to extract audio features from the mel spectrogram. On top of BEATs, we attach a Conformer [20] postencoder to further contextualize the audio features. Then, a BART [27] text decoder cross-attends to the contextualized audio features and generates the caption autoregressively. To provide text-modality guidance to our encoder stack, we extract the captions' sentence embeddings from an instruction-tuned large language model (LLM), INSTRUCTOR-XL [17], and apply an InfoNCE [19] auxiliary loss to train Conformer's output audio representation to mimic the corresponding caption's INSTRUCTOR sentence embedding.

that every component is indispensable to its great performance. We open-source our implementation,[2] pretrained model weights,[3] and ChatGPT-generated caption mix-ups.[4]

2. METHOD

2.1. Network Architecture and Main Loss Function

We utilize BEATs [13] as our main audio encoder. The BEATs module takes a 16 kHz audio waveform as input, converts the waveform into a mel spectrogram with a 10-millisecond hop size, splits the spectrogram into 2D patches, and transforms the patches into a sequence of representations through 12 self-attention (i.e., Transformer) layers. Compared to PANN [10], which has been popular in the AAC literature, BEATs comes with some key modifications:

- **Architecture:** BEATs features a Transformer backbone, while PANN is based on a convolutional neural network (CNN).
- **Pretraining objectives:** While both BEATs and PANN are pretrained on AudioSet [14], a general-domain, large-scale audio dataset, BEATs is first trained on masked language modeling [28] of tokenized audio, and then on multilabel audio classification. PANN is only trained on the latter.
- **Resolution:** BEATs provides more fine-grained outputs at 50 Hz, compared to PANN, which has 1-Hz outputs.

Due to these differences, and the better performance of BEATs on AudioSet multilabel classification (50.6% vs. 43.9% mean average precision), BEATs would likely be a more suitable selection for the AAC task. In our pilot experiments, we tried either to finetune the BEATs module or to keep it frozen. Both options led to similar SPIDEr-FL score, so we simply freeze BEATs to reduce computation and memory footprint.

[2]Code: https://github.com/slSeanWU/beats-conformer-bart-audio-captioner.
[3]Model: https://huggingface.co/slseanwu/beats-conformer-bart-audio-captioner.
[4]Mix-ups: https://huggingface.co/datasets/slseanwu/clotho-chatgpt-mixup-50K.

Given that the BEATs module is frozen, to enable further training on the audio features (more details in Section 2.2), we attach a convolutional downsampling layer, followed by a 2-layer Conformer [20] postencoder on top of the BEATs module. These additional layers further contextualize the audio features, and reduce the text decoder's workload for summarizing the audio features.

Following the recent trend of AAC models [4, 5], we adopt a 6-layer BART Transformer decoder [27] to generate captions. We use the default BART text tokenizer with a 50K vocabulary size, and train the BART's weights from scratch. The BART decoder cross-attends to the Conformer's output representations and self-attends to the historical caption tokens to generate the next caption token autoregressively. The main loss function to our BEATs-BART captioning model, applied on the BART's output distributions, is the negative log-likelihood (NLL) of audio captions, i.e.,

$$\mathcal{L}_{\text{NLL}} = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}_{\text{train}}}\left[\sum_{n=1}^{|\boldsymbol{y}|} -\log p(y_n|\boldsymbol{y}_{1:n-1};\boldsymbol{x})\right], \tag{1}$$

where $\mathcal{D}_{\text{train}}$ is the training dataset, $\boldsymbol{x}$ is the input audio waveform, $\boldsymbol{y}$ is an audio caption, and y_n is the n^{th} token in the caption. A schematic overview of our captioning model is depicted in Fig. 1.

2.2. INSTRUCTOR Embedding Supervision

To infuse text-related knowledge into our audio (i.e., encoder) features, we leverage an LLM, INSTRUCTOR-XL Transformer [17], to fetch the text embeddings for ground-truth captions and supervise our encoder stack with them. INSTRUCTOR is based on a pretrained T5 [29] text encoder, that is then finetuned using InfoNCE loss [19] on a variety of natural language processing (NLP) tasks, such as classification, reranking, summarization, and text quality evaluation, to learn sentence-level text embeddings. Task- and domain-specific instructions are prepended to the input text as conditions, e.g., "*Represent the Medicine statement for retrieval:*", hence the name INSTRUCTOR. In the Massive Text Embedding Benchmark (MTEB) [30], INSTRUCTOR-XL is the state of the art on text summarization and reranking tasks,[5] which are closely related to audio captioning.

In our use case, we place "*Represent the audio caption:*" as the instruction to the (frozen) INSTRUCTOR to fetch sentence embeddings from ground-truth captions. On our BEATs-Conformer encoder stack, we perform mean-pooling along the timestep dimension to obtain a single audio embedding for the input waveform. We denote the audio embedding and the INSTRUCTOR caption embedding by $\boldsymbol{a}$ and $\boldsymbol{c}$ respectively. An auxiliary InfoNCE loss is computed using in-batch negative samples:

$$\text{sim}(\boldsymbol{a},\boldsymbol{c}) = \exp\left(\frac{\boldsymbol{a}^\top\boldsymbol{c}}{||\boldsymbol{a}||\,||\boldsymbol{c}||}\cdot\frac{1}{\tau}\right), \tag{2}$$

$$\mathcal{L}_{\text{InfoNCE\_a}} = \mathbb{E}_{\mathcal{B}\subset\mathcal{D}_{\text{train}}}\left[\sum_{i=1}^{|\mathcal{B}|} -\log\frac{\text{sim}(\boldsymbol{a}_i,\boldsymbol{c}_i)}{\sum_{j=1}^{|\mathcal{B}|}\text{sim}(\boldsymbol{a}_j,\boldsymbol{c}_i)}\right], \tag{3}$$

$$\mathcal{L}_{\text{InfoNCE\_c}} = \mathbb{E}_{\mathcal{B}\subset\mathcal{D}_{\text{train}}}\left[\sum_{i=1}^{|\mathcal{B}|} -\log\frac{\text{sim}(\boldsymbol{a}_i,\boldsymbol{c}_i)}{\sum_{j=1}^{|\mathcal{B}|}\text{sim}(\boldsymbol{a}_i,\boldsymbol{c}_j)}\right], \tag{4}$$

$$\mathcal{L}_{\text{InfoNCE}} = 0.5\cdot(\mathcal{L}_{\text{InfoNCE\_a}} + \mathcal{L}_{\text{InfoNCE\_c}})\,, \tag{5}$$

where $\text{sim}(\cdot,\cdot)$ is the exponentiated temperature-scaled cosine sim-

[5]As of May 2023, when our project was being carried out.

Table 1: Randomly chosen samples of ChatGPT mix-up augmentations. In general, ChatGPT is able to faithfully reflect all content in the two input captions grammatically, and sometimes exhibits creativity in sentence structuring and vocabulary choice.

Clotho caption #1	Clotho caption #2	ChatGPT mix-up
water flowing over some rocks throughout a creek	in the distance fireworks pop and crackle constantly as they are set off	a serene creek babbles over rocks as distant fireworks pop and crackle in celebration
a muffled object is dragged along a surface in a room that echoes	several dogs barking with many birds making noise in the background	dogs bark in a room that echoes while a muffled object is dragged as birds chirp faintly in the background
a gate squeals as it sways while birds chirp in the background	a machine is whirring loudly at first and then slowly shuts off	as the gate sways and creaks a nearby machine loudly whirs before slowly powering down amidst chirping birds

ilarity, τ is the temperature hyperparameter,[6] $\mathcal{B}$ denotes a sampled mini-batch, and i, j index samples in the mini-batch. The multitask loss $\mathcal{L}$ used to train our model can hence be written as:

$$\mathcal{L} = \mathcal{L}_{\text{NLL}} + \alpha\mathcal{L}_{\text{InfoNCE}}\,, \tag{6}$$

where α is a hyperparameter and we find $\alpha = 1$ works well.

2.3. ChatGPT Mix-up Augmentation

As a novel data augmentation measure, we employ another LLM, ChatGPT [18], to 'mix-up' [23, 31] pairs of captions in the Clotho dataset, and create more complex and diverse in-domain training data. Specifically, we mix-up captions with different corresponding audio clips, rather than two ground-truth captions for the same audio. The corresponding audio waveforms are also mixed up to ensure consistency between audio and mixed-up captions.

We collect such mix-up augmentations using the public ChatGPT API. In the prompt, we ask ChatGPT to "*Generate a mix of the following two audio captions, and keep the generation under 25 words:*", and then provide it with two randomly sampled captions from Clotho [2]. We explicitly limit the number of words to force ChatGPT to be more concise. We use the FENSE disfluency detector [32] to filter out poor examples.[7] Mix-up of audio waveforms is more straightforward: we follow the algorithm used in WavLM [33], scaling the two waveforms to ensure their relative root-mean-square energy is within ± 5 dB before adding them together.

Table 1 displays a few examples of ChatGPT-generated mix-ups. We try including either 50K or 100K ChatGPT mix-ups, and using 50K yields a better performance. The API cost for generating 50K mix-ups is roughly \$8.50.

2.4. Sampling and Reranking

In past AAC research works, the most commonly used decoding algorithm has been beam search [4–6]. However, we find that, after introducing all the techniques in Section 2.1∼2.3, around 1/3 of generations using *nucleus sampling* [26], which is known to produce more diverse and informative generations than beam search, score higher in terms of SPIDEr-FL than those using beam search. This reveals the potential advantage of a sampling-then-reranking approach.

To 'pick the right sample' with nucleus sampling, we propose a hybrid reranking algorithm that utilizes again the knowledge of both our learned audio encoder stack and our text decoder. The two reranking metrics we consider are:

- **Caption log-likelihood:** We feed the input waveform $\boldsymbol{x}$ and the generated caption $\hat{\boldsymbol{y}}$ into our captioning model to directly compute $\log p(\hat{\boldsymbol{y}} \mid \boldsymbol{x}) = \sum_{n=1}^{|\hat{\boldsymbol{y}}|} \log p(\hat{y}_n \mid \hat{\boldsymbol{y}}_{1:n-1}; \boldsymbol{x})$ (cf. Eq. (1)). As the log-likelihood is computed on decoder outputs, we call this **decoder reranking**.
- **Audio-caption representation similarity:** We feed the generated caption $\hat{\boldsymbol{y}}$ into the INSTRUCTOR model to get its text embedding $\hat{\boldsymbol{c}}$, and fetch the audio embedding $\boldsymbol{a}$ of the input waveform $\boldsymbol{x}$ from our trained audio encoder stack. Then, we compute the cosine similarity between the text and audio embeddings, i.e., $(\boldsymbol{a}^\top \hat{\boldsymbol{c}}) \,/\, (||\boldsymbol{a}||\,||\hat{\boldsymbol{c}}||)$ (cf. Eq. (2)). As the representation from the audio encoder is used here, we refer to this as **encoder reranking**.

Candidate captions are ranked by the weighted sum of the two metrics above (with weights tuned on some held-out validation data), and we return the highest-scoring one as the final predicted caption. Our inference process is about three times slower than simple beam search with a beam size of 4.

3. EXPERIMENTS AND RESULTS

3.1. Training and Inference

We first pretrain the model on the combined dataset of AudioCaps[8] [3] and 50K ChatGPT mix-ups of samples from the better-curated but smaller Clotho [2] dataset for 10 epochs (about 13K gradient steps), and then finetune it on Clotho (development split, ∼4K samples) for 40 epochs (or 1.2K steps). Teacher-forcing is applied on the BART decoder inputs. We adopt the AdamW optimizer with a 2×10^{-4} learning rate for the 'AudioCaps + ChatGPT mix-up' pretraining stage, and 2×10^{-5} for the Clotho finetuning stage.

As the Conformer attention (see Section 2.1) is the primary memory bottleneck due to the long sequence length of audio features, there is a tradeoff between the batch size that can be used and the downsampling rate for the Conv1D layer between our BEATs and Conformer modules—using less downsampling gives the model finer-grained audio features, but a smaller batch size would lead to less reliable gradients and hamper contrastive learning [35]. Through experiments, we settle on the 3x downsampling rate, which allows a batch size of 32 and achieves the best performance.

We train the model on two NVIDIA A100 (40GB) GPUs, and the two training stages take around 6 and 3 hours respectively. Next-token prediction accuracy on the Clotho validation split is used as the checkpoint selection criterion.

At inference time, we experiment with generating $\{20, 50, 100\}$ candidate captions per test case with nucleus sampling,[9] and find that generating 50 strikes the best balance between performance gain and compute efficiency. Additionally, we leverage the FENSE evaluator [32] to filter out generations with fluency issues. We tune the weights of the reranking metrics (see Section 2.4) on the Clotho validation split and eventually pick $\{0.3, 0.7\}$ respectively for decoder and encoder reranking metrics.[10]

[6] Generally speaking, a higher temperature makes the contrastive objective more challenging, as the distribution is made less peaky. We perform a search in $\tau = \{0.03, 0.07, 0.2, 0.5, 1.0\}$ and find $\tau = 0.5$ works the best.

[7] Less than 1% of ChatGPT mix-ups are detected as disfluent.

[8] We filter out captions with <6 words, leading to 35K remaining samples.

[9] Nucleus sampling hyperparameters: temperature 0.5; cumulative distribution truncation point, i.e., top-p, 0.95.

[10] Weights for ablated models (see Table 2) are separately tuned to be fair.

Table 2: Ablation study on Clotho [2] evaluation split. The results demonstrate that all components in our AAC model, i.e., BEATs [13] audio encoder (Section 2.1), INSTRUCTOR [17] sentence embedding supervision (Section 2.2), ChatGPT [18] mix-up augmentation (Section 2.3), and nucleus sampling [26] + reranking (Section 2.4), are beneficial to the performance. We highlight the ablated components in brown.

	Model components					**Performance metrics** (in %)				
	Audio encoder	INSTRUCTOR emb.	ChatGPT mix-up	Decoding	Reranking	METEOR	CIDEr	SPICE	SPIDEr	SPIDEr-FL
Full model	BEATs	✓	✓	Sampling	Hybrid	**19.3**	**50.6**	**14.6**	**32.6**	**32.6**
w/o hybrid rerank and/or **sampling**	BEATs	✓	✓	Sampling	Decoder only	18.6	45.1	13.7	29.4	29.4 (−3.2)
	BEATs	✓	✓	Sampling	Encoder only	18.0	43.7	13.4	28.5	28.5 (−4.1)
	BEATs	✓	✓	Beam search	n.a.	18.7	47.4	13.4	30.4	30.3 (−2.3)
w/o ChatGPT mixup	BEATs	✓	✗	Sampling	Hybrid	19.1	47.6	13.8	30.7	30.7 (−1.9)
	BEATs	✓	✗	Beam search	n.a.	18.6	47.7	13.1	30.4	30.2 (−2.4)
w/o INSTRUCTOR	BEATs	✗	✓	Beam search	n.a.	18.6	45.8	13.4	29.6	29.4 (−3.2)
w/o BEATs	PANN	✓	✓	Sampling	Hybrid	17.5	39.7	12.3	26.0	26.0 (−6.6)
	PANN	✓	✓	Beam search	n.a.	17.4	41.6	12.2	26.9	26.7 (−5.9)

Table 3: Performance comparison (on Clotho [2] evaluation split) with top-ranking methods in recent DCASE challenges. All models presented here are single models, not ensembles. 'RL' indicates the use of reinforcement learning [34] to directly optimize CIDEr score. For fair comparison, best results in each group (i.e., *w/* or *w/o* RL) are **bold**-faced. Notice that RL can lead to a heavy punishment on SPIDEr-FL, the new DCASE official metric, due to fluency flaws.

		Performance metrics (in %)				
	RL	METEOR	CIDEr	SPICE	SPIDEr	SPIDEr-FL
Ours	✗	**19.3**	**50.6**	**14.6**	**32.6**	**32.6**
Labbé et al., '23 [8]	✗	19.2	48.5	13.9	31.2	31.0
Cho et al., '23 [7]	✗	18.8	48.3	13.7	31.0	30.7
Ye et al., '22 [6]	✗	17.8	44.5	12.7	28.6	n.a. (≤28.6)
Xu et al., '22 [5]	✗	17.9	42.1	12.7	27.4	n.a. (≤27.4)
Cho et al., '23 [7]	✓	**19.5**	**52.6**	**14.3**	**33.5**	**22.5**
Ye et al., '22 [6]	✓	18.5	50.3	13.2	31.7	n.a. (≤31.7)
Xu et al., '22 [5]	✓	18.6	50.9	12.0	31.5	n.a. (≤31.5)

3.2. Evaluation

The metrics used to evaluate the quality of generated captions are: METEOR, CIDEr, SPICE, SPIDEr, and SPIDEr-FL. METEOR and CIDEr are both based on n-gram overlap, with the former penalizing fragmentation between the ground-truth and generated captions, and the latter promoting generating informative words by weighting n-grams by their TF-IDF scores. SPICE focuses on the overlap computed on semantic graphs constructed by objects, object attributes, and relations. SPIDEr is the simple mean of CIDEr and SPICE, and it had been the official evaluation metric in DCASE AAC challenges until 2022. In 2023, the official metric was changed to SPIDEr-FL, which uses FENSE [32], i.e., a BERT-based binary classifier, to penalize the SPIDEr score of disfluent generations by 90%.

Our evaluation is done on the public 'evaluation' split[11] of the Clotho [2] dataset, which consists of 1,045 samples. Evaluation results of our full model can be found in the 1st row of Table 2.

3.3. Comparison with Past and Concurrent Works

We compare our model (i.e., winner of the DCASE 2023 AAC challenge) to other top performers in the 2022 and 2023 challenges [5–8]. We note that while all the best-scoring systems for each participant were ensemble models, we present the metrics for single models for fairness and practicality reasons.[12] The comparison in Table 3 shows that our model is state-of-the-art in terms of the new official metric, SPIDEr-FL, and performs competitively on other metrics. Moreover, while optimizing CIDEr with reinforcement learning has been popular among challenge submissions, the resulting disfluency issues [36] get severely punished on SPIDEr-FL (see 6th row in Table 3).

3.4. Ablation Study

To show that every component in our AAC model is indispensable to achieve the best performance, we conduct a comprehensive ablation study that tries to remove components *one at a time*. Table 2 presents the results that corroborate the necessity of all of our model components—Each one of them gives at least a 2-point improvement on SPIDEr-FL,[13] with BEATs audio encoder (replacing the popular PANN) being the most crucial one causing a 6-point difference.

Some intriguing additional findings are: (i) hybrid reranking is required to outperform beam search (see rows 1∼4 in Table 2), suggesting that decoder and encoder reranking methods are strongly complementary and hence should be used together when possible; (ii) the 'sampling + reranking' decoding approach improves performance the most when both BEATs encoder and ChatGPT mix-ups are used (see rows '1 vs. 4', '5 vs. 6', and '8 vs. 9' in Table 2).

4. CONCLUSION AND FUTURE WORK

In this work, we improved audio captioning models from multiple aspects with an extensive use of pretrained models. We employed the BEATs Transformer to extract more fine-grained audio features. We then utilized the INSTRUCTOR text embeddings for multitask learning to provide rich language-modality guidance. ChatGPT was also leveraged to generate faithful and fluent caption mix-ups which, when paired with the corresponding audio mix-ups, increased the size, diversity, and complexity of our training data. Finally, nucleus sampling and hybrid reranking were used to exploit our model's capabilities to the fullest extent. We accomplished a state-of-the-art 32.6 SPIDEr-FL score and demonstrated via a thorough ablation study that all components are crucial to our model's success.

Future endeavors may explore audio feature extractors that are pretrained with larger amounts of data [37] or multimodal supervision [38]. More advanced reinforcement learning methods [39] can also be applied to optimize captioning metrics that correlate well with human judgment [40] without introducing disfluency issues.

[11] DCASE challenge uses the blind 'test' split to rank the submissions.

[12] Ensembles can contain a wildly different # of models (e.g., 3∼20), and the performance gain can seldom justify the extra compute required.

[13] We cannot perform hybrid reranking with the '**w/o INSTRUCTOR**' setting as the audio encoder is not trained to match the caption text embedding. Thus, we simply use beam search as it outperforms decoder-only reranking.

5. REFERENCES

[1] K. Drossos, S. Adavanne and T. Virtanen, "Automated audio captioning with recurrent neural networks," in *Proc. WASPAA*, 2017.

[2] K. Drossos, S. Lipping and T. Virtanen, "Clotho: an audio captioning dataset," in *Proc. ICASSP*, 2020.

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

[4] W. Yuan, Q. Han, D. Liu et al., "The DCASE 2021 challenge task 6 system: Automated audio captioning with weakly supervised pre-training and word selection methods," Tech. Rep., DCASE Challenge, 2021.

[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," Tech. Rep., DCASE Challenge, 2022.

[6] Z. Ye, Y. Zou, F. Cui and Y. Wang, "Automated audio captioning with multi-task learning," Tech. Rep., DCASE Challenge, 2022.

[7] J.-H. Cho, Y.-A. Park, J. Kim and J.-H. Chang, "HYU submission for the DCASE 2023 task 6a: Automated audio captioning model using AL-MixGen and synonyms substitution," Tech. Rep., DCASE Challenge, 2023.

[8] E. Labbé, T. Pellegrini and J. Pinquier, "IRIT-UPS DCASE 2023 audio captioning and retrieval system," Tech. Rep., DCASE Challenge, 2023.

[9] C. P. Narisetty, T. Hayashi, R. Ishizaki et al., "Leveraging state-of-the-art ASR techniques to audio captioning.," in *Proc. DCASE*, 2021.

[10] Q. Kong, Y. Cao, T. Iqbal et al., "PANNs: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2020.

[11] K. Chen, X. Du, B. Zhu et al., "HTS-AT: A hierarchical token-semantic audio transformer for sound classification and detection," in *Proc. ICASSP*, 2022.

[12] P.-Y. Huang, H. Xu, J. Li et al., "Masked autoencoders that listen," in *Proc. NeurIPS*, 2022.

[13] S. Chen, Y. Wu, C. Wang et al., "BEATs: Audio pre-training with acoustic tokenizers," *arXiv preprint arXiv:2212.09058*, 2022.

[14] J. F. Gemmeke, D. P. Ellis, D. Freedman et al., "Audio Set: An ontology and human-labeled dataset for audio events," in *Proc. ICASSP*, 2017.

[15] L. Ouyang, J. Wu, X. Jiang et al., "Training language models to follow instructions with human feedback," in *Proc. NeurIPS*, 2022.

[16] H. Touvron, T. Lavril, G. Izacard et al., "LLaMA: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[17] H. Su, J. Kasai, Y. Wang et al., "One embedder, any task: Instruction-finetuned text embeddings," *Findings of ACL*, 2023.

[18] J. Schulman, B. Zoph, C. Kim et al., "Introducing ChatGPT," *OpenAI Blog*, 2022.

[19] A. van den Oord, Y. Li and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[20] A. Gulati, J. Qin, C.-C. Chiu et al., "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. Interspeech*, 2020.

[21] K. Miyazaki, T. Komatsu, T. Hayashi et al., "Convolution-augmented transformer for semi-supervised sound event detection," in *Proc. DCASE*, 2020.

[22] D. S. Park, W. Chan, Y. Zhang et al., "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Proc. Interspeech*, 2019.

[23] H. Zhang, M. Cisse, Y. N. Dauphin and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *Proc. ICLR*, 2018.

[24] T. Kouzelis, G. Bastas, A. Katsamanis and A. Potamianos, "Efficient audio captioning transformer with patchout and text guidance," Tech. Rep., DCASE Challenge, 2022.

[25] X. Mei, C. Meng, H. Liu et al., "WavCaps: A chatgpt-assisted weakly-labelled audio captioning dataset for audio-language multimodal research," *arXiv preprint arXiv:2303.17395*, 2023.

[26] A. Holtzman, J. Buys, L. Du et al., "The curious case of neural text degeneration," in *Proc. ICLR*, 2020.

[27] M. Lewis, Y. Liu, N. Goyal et al., "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proc. ACL*, 2020.

[28] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of deep bidirectional Transformers for language understanding," in *Proc. NAACL-HLT*, 2019.

[29] C. Raffel, N. Shazeer, A. Roberts et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, 2020.

[30] N. Muennighoff, N. Tazi, L. Magne and N. Reimers, "MTEB: Massive text embedding benchmark," *arXiv preprint arXiv:2210.07316*, 2022.

[31] Y. Gong, Y.-A. Chung and J. Glass, "PSLA: Improving audio tagging with pretraining, sampling, labeling, and aggregation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2021.

[32] Z. Zhou, Z. Zhang, X. Xu et al., "Can audio captions be evaluated with image caption metrics?," in *Proc. ICASSP*, 2022.

[33] S. Chen, C. Wang, Z. Chen et al., "WavLM: Large-scale self-supervised pre-training for full stack speech processing," *IEEE Journal of Selected Topics in Signal Processing*, 2022.

[34] S. J. Rennie, E. Marcheret, Y. Mroueh et al., "Self-critical sequence training for image captioning," in *Proc. CVPR*, 2017.

[35] T. Chen, S. Kornblith, M. Norouzi and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. ICML*, 2020.

[36] X. Mei, Q. Huang, X. Liu et al., "An encoder-decoder based audio captioning system with transfer and reinforcement learning," in *Proc. DCASE*, 2021.

[37] B. Elizalde, S. Deshmukh, M. Al Ismail and H. Wang, "CLAP: learning audio concepts from natural language supervision," in *Proc. ICASSP*, 2023.

[38] J. Schulman, B. Zoph, C. Kim et al., "ImageBind: holistic AI learning across six modalities," *Meta AI Blog*, 2023.

[39] G. O. dos Santos, E. L. Colombini and S. Avila, "CIDEr-R: Robust consensus-based image description evaluation," in *Proc. Workshop on Noisy User-generated Text*, 2021.

[40] J. Hessel, A. Holtzman, M. Forbes et al., "CLIPScore: A reference-free evaluation metric for image captioning," in *Proc. EMNLP*, 2021.