BEATS-BASED AUDIO CAPTIONING MODEL WITH INSTRUCTOR EMBEDDING SUPERVISION AND CHATGPT MIX-UP

Technical Report

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
{shihlunw, xuankaic, jeeweonj, swatanab}@andrew.cmu.edu

[2] Speech & Audio Team, Mitsubishi Electric Research Labs, Cambridge, MA, USA
{wichern, germain, leroux}@merl.com

ABSTRACT

DCASE 2023 Task 6A, automated audio captioning (AAC), aims at generating informative descriptions for various sounds from nature and/or human activities. Our AAC system follows the sequence-to-sequence (seq2seq) architecture. The audio encoder stack is comprised of a frozen BEATs Transformer followed by a 2-layer Conformer. The BEATs module, which has been pretrained on both masked audio token prediction and audio event classification, extracts fine-grained (i.e., $\approx$ 50 Hz) audio features, while the Conformer downsamples and summarizes the audio features before they are cross-attended by the BART text decoder. Besides the autoregressive negative log-likelihood (NLL) loss computed on decoder outputs, we simultaneously apply an audio-text contrastive loss on our encoder output to infuse language modality knowledge into it. Specifically, we feed ground-truth captions into INSTRUCTOR Transformer, a state-of-the-art text embedding model, and teach our audio encoder to predict the INSTRUCTOR text embeddings through InfoNCE loss. In addition, we leverage ChatGPT to produce caption mix-ups (i.e., grammatical and compact combinations of two captions) which, together with the corresponding audio mixtures, increases not only the amount but also the complexity and diversity of our training data. During inference, we employ nucleus sampling and a hybrid reranking algorithm that considers both likelihood and audio-caption representation similarity. Combining our efforts, our best single model and ensemble system achieve 0.326 and 0.336 SPIDEr-FL scores, respectively, on the Clotho (V2) evaluation split.

Index Terms— BEATs, Conformer, INSTRUCTOR, ChatGPT, InfoNCE

1. INTRODUCTION

Automated audio captioning (AAC) is a multimodal task that describes an input audio clip using text. The description does not use a fixed set of class labels or tags, but instead uses a free text natural language description [1]. Research progress on AAC has accelerated in recent years thanks to the yearly DCASE challenges, the impressive performance of Transformer-based auto-regressive language models, and the release of the open audio captioning datasets Clotho [2, 3] and AudioCaps [4].

Our submission to the DCASE 2023 AAC task is in line with the macro-trend in machine learning where models pretrained on large datasets are actively utilized. Specifically, we use pretrained models in multiple ways to make the quality of audio representations better match the captioning task. We start with audio features extracted from a Bidirectional Encoder representation from Audio Transformers (BEATs) [5] model, a state-of-the-art audio classification model which is pretrained on AudioSet. We also use the pretrained INSTRUCTOR Transformer [6] to obtain embeddings for the ground-truth captions, and train our audio encoder outputs to predict these INSTRUCTOR embeddings as an additional task. Following the pretrained BEATs module, we use a 2-layer Conformer [7, 8] to downsample the audio features. Then, following the typical seq2seq architecture, a BART text decoder attends to the downsampled audio features using cross-attention.

We use ChatGPT [9] to combine the captions from two mixed audio clips in a much more natural way compared to simple concatenation. This idea is inspired by previous work on AAC and text-to-audio synthesis, which performs data augmentation by mixing multiple audio signal inputs and their corresponding text labels. The text label mixing is done by either combining the corresponding caption embeddings [10], or by simply concatenating the captions (or concatenating the keywords from the captions) [11]. In our proposed approach with ChatGPT, we take that concept one step further. Finally, during inference, we observed that using nucleus sampling to generate captions in the decoder would sometimes produce captions that significantly outperformed beam search in terms of SPIDEr-FL. Therefore, we use a hybrid reranking algorithm that combines likelihood and audio-caption representation similarity to select the nucleus sampling output.

2. METHOD

2.1. Network Architecture and Main Loss Function

We utilize BEATs [5] as our main audio encoder. The BEATs module takes a 16 kHz audio waveform as input, converts the waveform into a mel spectrogram with 10-millisecond hop size, splits the spectrogram into 2D patches, and finally transforms the patches into a sequence of latent representations through 12 self-attention (i.e., Transformer) layers. Compared to the PANN [13] audio encoder used in the official baseline and past winning systems [14, 15, 16], BEATs comes with the following modifications:

- **Network:** BEATs features a Transformer backbone, while PANN

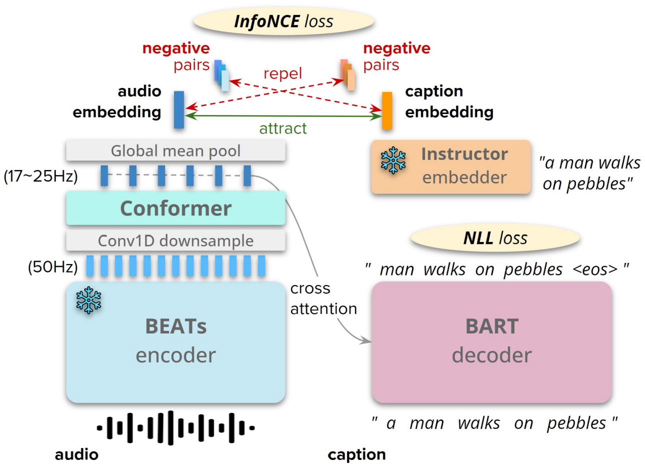

Figure 1: Overview of our Transformer-based captioning system. We utilize a frozen BEATs [5] Transformer (pretrained on masked audio language modeling and multilabel audio classification) to extract audio features from the mel spectrogram. On top of BEATs, we attach a randomly initialized Conformer [7] to further contextualize and downsample the audio features. Then, a BART [12] text decoder cross-attends to the contextualized audio features and generates the caption autoregressively. To give the BEATs-Conformer encoder stack better text-modality guidance, we extract the captions' sentence embeddings from a state-of-the-art embedding model: INSTRUCTOR [6]. An InfoNCE auxiliary loss is applied on the Conformer's output representation (mean-pooled along the time dimension) to train it to mimic the corresponding caption's INSTRUCTOR sentence embedding.

is based on convolutional neural network (CNN).

- **Pretraining objectives:** While both BEATs and PANN are pretrained on AudioSet [17], a general-domain, large-scale audio dataset, BEATs is first trained on masked audio language modeling, and then on multilabel audio classification, and PANN is only trained on the latter.
- **Resolution:** BEATs outputs audio features at about 50 Hz. This is up from around 1 Hz for PANN.

Due to these differences, and the better performance of BEATs on AudioSet multilabel classification (50.6% vs. 43.9% mean average precision), we believe that BEATs could provide higher-quality, more fine-grained features for the audio captioning task. In our pilot experiments, we try either to finetune the BEATs module or to keep it frozen. Both options lead to similar SPIDEr-FL score, so we simply freeze BEATs to reduce computation and memory footprint.

Given that the BEATs module is frozen, to enable further training on the audio representations (more details to come in Section 2.2), we attach a convolutional downsampling layer, followed by a 2-layer[1] Conformer [7] on top of the BEATs module. These additional layers further contextualize the audio features, and reduce the text decoder's workload on summarizing the audio features.

Following the DCASE 2023 official baseline, we adopt a 6-layer BART Transformer decoder [12] 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, which is applied on the BART's output distributions, is the negative log-likelihood (NLL) of audio captions, i.e.,

$$\mathcal{L}_{\text{NLL}} = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}_{\text{train}}}\Big[-\log p(\boldsymbol{y}\mid\boldsymbol{x})\Big] \tag{1}$$

$$= \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}_{\text{train}}}\Big[\sum_{n=1}^{|\boldsymbol{y}|} -\log p(y_n|\boldsymbol{y}_{1:n-1};\boldsymbol{x})\Big]\,, \tag{2}$$

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

Past top-performing submissions in the DCASE captioning challenge have proposed a number of ways to infuse text-related knowledge into the audio (i.e., encoder) representations. For example, [14] pretrained the PANN audio encoder with an audio-caption InfoNCE [18] contrastive loss, and [15] added a feed-forward network on top of PANN to predict keywords extracted from the captions, and trained the captioning model in a multitask fashion.

In our submissions, we strive to combine the benefits of both representation learning and multitask training. In particular, we leverage the INSTRUCTOR-XL[2] Transformer [6] to fetch the text embedding for the audio captions and supervise our encoder stack with them. INSTRUCTOR is based on a pretrained T5 [19] text encoder, that is then finetuned using InfoNCE loss [18] 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 prompts are prepended to the input text as conditions, e.g., "*Represent the Medicine statement for retrieval:*" is used for medical text, hence the name INSTRUCTOR. In the Massive Text Embedding Benchmark (MTEB) [20], INSTRUCTOR-XL is currently the state of the art on summarization and reranking tasks,[3] which are closely related to audio captioning.

In our use case, we place "*Represent the audio caption:*" as the prompt 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:

$$\text{sim}(\boldsymbol{a},\boldsymbol{c}) = \exp\Big(\frac{\boldsymbol{a}^\top\boldsymbol{c}}{||\boldsymbol{a}||\,||\boldsymbol{c}||}\cdot\frac{1}{\tau}\Big), \tag{3}$$

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

$$\mathcal{L}_{\text{InfoNCE\_c}} = \mathbb{E}_{\mathcal{B}\subset\mathcal{D}_{\text{train}}}\Big[\sum_{i=1}^{|\mathcal{B}|} -\log\frac{\text{sim}(\boldsymbol{a}_i,\boldsymbol{c}_i)}{\sum_j^{|\mathcal{B}|}\text{sim}(\boldsymbol{a}_i,\boldsymbol{c}_j)}\Big], \tag{5}$$

$$\mathcal{L}_{\text{InfoNCE}} = \frac{1}{2}\big(\mathcal{L}_{\text{InfoNCE\_a}} + \mathcal{L}_{\text{InfoNCE\_c}}\big)\,, \tag{6}$$

[1] The number of layers is determined by hyperparameter search.

[2] 'XL' describes the size of the network—it has 1.5B parameters.

[3] huggingface.co/spaces/mteb/leaderboard

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 dog is growling rather furiously at something	someone is using a washing machine to do their laundry	a persons laundry spins in a washing machine as a dog growls ferociously somewhere nearby
a man is talking over a radio device	water cascades into a sink while a person cleans their face	a mans voice crackles over a radio as water splashes into a sink during someones routine facial cleanse
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

where $\mathrm{sim}(\cdot,\cdot)$ is the exponentiated temperature-scaled cosine similarity, τ is the temperature hyperparameter,[4] $\mathcal{B}$ denotes a sampled mini-batch, and i, j index samples in the mini-batch. The total multitask loss $\mathcal{L}$ used to train our model can hence be written as:

$$\mathcal{L} = \mathcal{L}_{\text{NLL}} + \alpha\mathcal{L}_{\text{InfoNCE}}\,, \tag{7}$$

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

2.3. ChatGPT Mix-up Augmentation

Since the Clotho [3] dataset has a rather limited size, historically, challenge participants [14, 15, 16, 21] have utilized SpecAugment [22] and external captioning dataset like AudioCaps [4] to pretrain the captioning model. In addition to the two aforementioned data augmentation measures, we leverage a large language model (LLM), ChatGPT [9], to 'mix-up' [23, 24] 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 out of the 5 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.[5] 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. As the maximum caption length in Clotho is 20 words, we instruct ChatGPT to keep the output length less than 25 words. We use the FENSE disfluency detector [25] in the SPIDEr-FL metric to filter out poor examples[6]. The remaining ChatGPT mix-ups are used together with AudioCaps dataset to pretrain our captioning model. Mix-up of audio waveforms is more straightforward, we simply follow the algorithm used in WavLM [26] to scale the two waveforms to ensure their relative root-mean-square energy is within ±5 dB before adding them together.[7]

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 higher SPIDEr-FL score. The cost for generating 50K mix-ups is roughly \$8.50.

2.4. Sampling and Reranking

In past DCASE challenges, the most commonly used decoding algorithm for the audio captioning task has been beam search [14, 15, 16]. However, in our pilot experiments, we find that around 1/3 of generations using *nucleus sampling* [27] (with temperature 0.5 and cumulative distribution truncation point i.e., top-p of 0.95) scores higher in terms of SPIDEr-FL than that using beam search, revealing 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 $\log p(\hat{\boldsymbol{y}} \mid \boldsymbol{x}) = \sum_{n}^{|\hat{\boldsymbol{y}}|} \log p(\hat{y}_n \mid \hat{\boldsymbol{y}}_{1:n-1}; \boldsymbol{x})$.
- **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 aforementioned text and audio embeddings, i.e., $(\boldsymbol{a}^\top\hat{\boldsymbol{c}})\;/\;(||\boldsymbol{a}||\;||\hat{\boldsymbol{c}}||)$.

We tune the weights of the two reranking metrics on the Clotho validation split, and find $\{0.3,\ 0.7\}$ for log-likelihood and representation similarity, respectively, performs well across all models. To strike a balance between diversity and compute-efficiency, we generate 50 independent captions for each test case. Additionally, we leverage the FENSE evaluator [25] to filter out generations that would be punished on the SPIDEr-FL metric. Generally, using nucleus sampling and our hybrid reranking method leads to a 0.01∼0.02 SPIDEr-FL gain over beam search.

3. SUBMISSIONS AND RESULTS

In total, we submit 4 captioning systems, with submission #1 being a single model, and submissions #2, #3 and #4 being ensemble models. As the architecture and techniques proposed in Sections 2.1 through 2.4 are orthogonal to each other, we combine all of them in our submissions. Important characteristics of our submissions and results on the Clotho evaluation split can be found in Table 2.

For our submission #1, we first pretrain the model on the combined dataset of AudioCaps [4] and 50K ChatGPT mix-ups from

[4] Generally speaking, a higher temperature makes the contrastive objective more challenging, as the distribution is made less peaky. We perform a search in τ = {0.03, 0.07, 0.2, 0.5, 1.0} and find τ = 0.5 works the best. Note that the DCASE official baseline uses τ = 0.07.

[5] ChatGPT API guide: `platform.openai.com/docs/guides/chat/introduction`

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

[7] We also attempt loudness normalization via `pyloudnorm` to scale the waveforms to a peak energy of –15 to –5 dBFS (randomly chosen from uniform distribution), but no SPIDEr-FL improvement is seen with this strategy.

Table 2: Characteristics and performance (on Clotho evaluation split) of our submissions. All metrics are the higher the better. All systems use BART as the text decoder. Our systems leverage AudioCaps as extra data while the DCASE baseline does not. 'Aux. loss' is applied to train the audio encoder to predict the sentence embedding of corresponding captions.

		Characteristics					**Performance metrics**				
		# models	Audio encoder	ChatGPT mix-up	Aux. loss	Weighted ensemble	METEOR	CIDEr	SPICE	SPIDEr	SPIDEr-FL
Single	DCASE baseline	1	PANN	✗	InfoNCE	n.a.	.177	.420	.119	.270	.261
	Submission #1	1	BEATs + Conformer	✓	InfoNCE	n.a.	.193	.506	.146	.326	.326
Ensemble	Submission #2	7	BEATs + Conformer	✓/✗	InfoNCE / cos sim	✗	**.198**	.510	**.147**	.329	.329
	Submission #3	20	BEATs + Conformer	✓	InfoNCE	✓	.197	**.525**	**.147**	**.336**	**.336**
	Submission #4	20	BEATs + Conformer	✓	InfoNCE	✗	.197	.505	.145	.325	.325

Clotho [3] for 10 epochs (about 12K gradient steps), and then finetune it on the Clotho development split for 40 epochs (or 1.2K steps). The optimizer adopted is AdamW [28], with a 2e–4 learning rate for the 'AudioCaps + ChatGPT mix-up' pretraining stage, and 2e–5 for the Clotho finetuning stage. As the Conformer attention is the primary memory bottleneck,[8] there is a tradeoff between the largest batch size we could use 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 the smaller batch size allowed would make the gradient less reliable. Through experiments, we eventually settle on 3x downsampling rate and a batch size of 32 for submission #1. We train the model on two NVIDIA A100 (40G) GPUs, and the two training stages take around 6 and 3 hours respectively. Early stopping on validation accuracy (with 5 epochs of patience) is employed.

For our ensemble system submission #2, we try to prioritize diversity across the 7 component models in the following aspects:

- **Attention architecture:** In Conformer, we use either the conventional quadratic-complexity attention, or the Performer attention [29], which provably approximates softmax attention in linear time and memory. Performer attention enables us to use batch size 48 instead of 32.
- **Downsampling rate:** Either 2x or 3x downsampling rate is used in the Conv1D layer between BEATs and Conformer.
- **Training data:** In the pretraining stage, some of the component models do not use mix-up augmentations from ChatGPT, but rather only the AudioCaps dataset.
- **Auxiliary loss:** Some component models simply learn to predict INSTRUCTOR text embeddings through cosine similarity maximization, instead of using InfoNCE. Cosine similarity maximization can be thought of as InfoNCE minimization without the repelling force from negative samples.

We ensemble the models by simply averaging the BART text decoder output distributions.

For submissions #3 and #4, we include 4 top-performing model configurations from submission #2, and attempt to leverage data from Clotho validation split using cross validation. Specifically, we first combine the Clotho development and validation splits (which contain 3.9K and 1.0K samples respectively), and repartition the combined data into 5 equal-sized folds. Then, we can train a model using folds 1∼4, validating it on the 5th fold, another model on folds 2∼5, validating it on the 1st fold, and so on. This way, we will have 5 models which have collectively seen all data in the Clotho development and validation splits, expanding the dataset size by roughly 25%. Since we consider 4 model configurations, the total number of models is $4 \times 5 = 20$. The only difference between submissions #3 and #4 is that for submission #3, we additionally train the combination weights of BART output distributions rather than simply averaging them.

The performance metrics in Table 2 show that our submissions outperform the DCASE baseline by a considerable margin, and perform on par with or slightly better than the DCASE 2022 winning system [14], which achieved a 0.325 SPIDEr score, without using reinforcement learning [30] to directly optimize the CIDEr metric.

4. CONCLUSION AND FUTURE WORK

In the DCASE 2023 challenge, we strived to improve audio captioning models from multiple aspects. We employed the BEATs Transformer to extract more fine-grained audio features. We then utilized the INSTRUCTOR embedding model and multitask learning to provide rich language-modality guidance to our audio encoder stack. ChatGPT was also leveraged to generate faithful and fluent caption mix-ups which, when paired with mix-ups in the audio domain, increased the size, diversity, and complexity of our training data. Finally, nucleus sampling and a hybrid reranking method once again utilized our model's capabilities and led to a healthy additional performance gain.

Future endeavors may explore feature extractors that are pretrained with larger amounts of data [31] or multimodal supervision [32]. Reinforcement learning can also be included into our pipeline to optimize captioning metrics that correlates well with human judgment [33] without introducing fluency issues [34].

5. ACKNOWLEDGEMENTS

Shih-Lun Wu, Xuankai Chang, Jee-weon Jung, and Shinji Watanabe used the Bridges2 system at PSC and Delta system at NCSA through allocation CIS210014 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

6. REFERENCES

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

[2] S. Lipping, K. Drossos, and T. Virtanen, "Crowdsourcing a dataset of audio captions," in *Proc. DCASE*, Nov. 2019.

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

[8] due to the long sequence length of audio features

[4] C. D. Kim, B. Kim, H. Lee, and G. Kim, "AudioCaps: Generating captions for audios in the wild," in *Proc. NAACL-HLT*, 2019, pp. 119–132.

[5] S. Chen, Y. Wu, C. Wang, S. Liu, D. Tompkins, Z. Chen, and F. Wei, "BEATs: Audio pre-training with acoustic tokenizers," *arXiv preprint arXiv:2212.09058*, 2022.

[6] H. Su, J. Kasai, Y. Wang, Y. Hu, M. Ostendorf, W.-t. Yih, N. A. Smith, L. Zettlemoyer, T. Yu, *et al.*, "One embedder, any task: Instruction-finetuned text embeddings," *arXiv preprint arXiv:2212.09741*, 2022.

[7] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. Interspeech*, 2020.

[8] K. Miyazaki, T. Komatsu, T. Hayashi, S. Watanabe, T. Toda, and K. Takeda, "Convolution-augmented transformer for semi-supervised sound event detection," in *Proc. DCASE*, 2020.

[9] J. Schulman, B. Zoph, C. Kim, J. Hilton, J. Menick, J. Weng, *et al.*, "Introducing ChatGPT," *OpenAI Blog*, 2022.

[10] T. Kouzelis, G. Bastas, A. Katsamanis, and A. Potamianos, "Efficient audio captioning transformer with patchout and text guidance," DCASE2022 Challenge, Tech. Rep., July 2022.

[11] F. Kreuk, G. Synnaeve, A. Polyak, U. Singer, A. Défossez, J. Copet, D. Parikh, Y. Taigman, and Y. Adi, "AudioGen: Textually guided audio generation," in *Proc. ICLR*, 2023.

[12] 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," in *Proc. ACL*, 2020.

[13] 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*, 2020.

[14] 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., 2022.

[15] Z. Ye, Y. Zou, F. Cui, and Y. Wang, "Automated audio captioning with multi-task learning," DCASE2022 Challenge, Tech. Rep., 2022.

[16] W. Yuan, Q. Han, D. Liu, X. Li, and Z. Yang, "The DCASE 2021 challenge task 6 system: Automated audio captioning with weakly supervised pre-training and word selection methods," DCASE2021 Challenge, Tech. Rep., 2021.

[17] J. F. Gemmeke, D. P. 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. ICASSP*, 2017.

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

[19] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, 2020.

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

[21] C. P. Narisetty, T. Hayashi, R. Ishizaki, S. Watanabe, and K. Takeda, "Leveraging state-of-the-art asr techniques to audio captioning." in *Proc. DCASE*, 2021.

[22] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "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] 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.

[25] Z. Zhou, Z. Zhang, X. Xu, Z. Xie, M. Wu, and K. Q. Zhu, "Can audio captions be evaluated with image caption metrics?" in *Proc. ICASSP*, 2022.

[26] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao, *et al.*, "WavLM: Large-scale self-supervised pre-training for full stack speech processing," *IEEE Journal of Selected Topics in Signal Processing*, 2022.

[27] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *Proc. ICLR*, 2020.

[28] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. ICLR*, 2019.

[29] K. M. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Q. Davis, A. Mohiuddin, L. Kaiser, *et al.*, "Rethinking attention with Performers," in *Proc. ICLR*, 2021.

[30] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," in *Proc. CVPR*, 2017.

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

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

[33] J. Hessel, A. Holtzman, M. Forbes, R. Le Bras, and Y. Choi, "CLIPScore: A reference-free evaluation metric for image captioning," in *Proc. EMNLP*, 2021.

[34] 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.