CoNeTTE: An efficient Audio Captioning system leveraging multiple datasets with Task Embedding

Étienne Labbé, Thomas Pellegrini, Julien Pinquier
IRIT, Université de Toulouse, CNRS, Toulouse INP, UT3, Toulouse, France

***Abstract*—Automated Audio Captioning (AAC) involves generating natural language descriptions of audio content, using encoder-decoder architectures. An audio encoder produces audio embeddings fed to a decoder, usually a Transformer decoder, for caption generation. In this work, we describe our model, which novelty, compared to existing models, lies in the use of a ConvNeXt architecture as audio encoder, adapted from the vision domain to audio classification. This model, called CNext-trans, achieved state-of-the-art scores on the AudioCaps (AC) dataset and performed competitively on Clotho (CL), while using four to forty times fewer parameters than existing models. We examine potential biases in the AC dataset due to its origin from AudioSet by investigating unbiased encoder's impact on performance. Using the well-known PANN's CNN14, for instance, as an unbiased encoder, we observed a 1.7% absolute reduction in SPIDEr score (where higher scores indicate better performance). To improve cross-dataset performance, we conducted experiments by combining multiple AAC datasets (AC, CL, MACS, WavCaps) for training. Although this strategy enhanced overall model performance across datasets, it still fell short compared to models trained specifically on a single target dataset, indicating the absence of a one-size-fits-all model. To mitigate performance gaps between datasets, we introduced a Task Embedding (TE) token, allowing the model to identify the source dataset for each input sample. We provide insights into the impact of these TEs on both the form (words) and content (sound event types) of the generated captions. The resulting model, named CoNeTTE, an unbiased CNext-trans model enriched with dataset-specific Task Embeddings, achieved SPIDEr scores of 44.1% and 30.5% on AC and CL, respectively. For the sake of reproducibility, we have made our code publicly available[1].**

***Index Terms*—Audio-language task, automated audio captioning, dataset biases, task embedding, deep learning**

I. Introduction

In recent years, natural language processing has gained significant popularity and emerged as a universal interface facilitating human-machine interactions, owing to remarkable advancements in machine learning systems. Unlike predefined class sets, human-generated free text typically contains more comprehensive information, encompassing relationships between entities, complex scene descriptions, and object attributes. Initially introduced for image content, the image captioning task aims to create models capable of describing visual content using natural language (e.g., [1]). This concept was extended to audio processing, giving rise to Automated Audio Captioning (AAC) (e.g., [2]), with the goal of generating text-based descriptions for audio content.

AAC systems use encoder-decoder architectures, where an audio encoder provides a sequence of embeddings to a decoder, either a recurrent neural network or a Transformer decoder, responsible for generating a caption [3], [4]. In this work, we describe our model referred to as CNext-trans, in which the audio encoder is a fully convolutional neural network adapted from the vision domain, the well-known ConvNeXt architecture [5]. The decoder is a vanilla Transformer decoder, trained from scratch on the AAC datasets at hand. As we will report here-after, CNext-trans performed very favorably on AudioCaps (AC) [6] and Clotho (CL) [7], two datasets widely used in AAC.

All the existing systems, including ours, use audio encoders pretrained on at least AudioSet (AS) [8]. This rises a bias issue when doing experiments on AC, since the AC audio recordings were selected from the training subset of AS. In this work, we compare the use of biased and unbiased audio encoders on AC, and we observe that the difference in AAC performance between the two is relatively small.

Next, we explore combining four AAC datasets (AC, CL, MACS [9], WavCaps [10]) for training. This was motivated by the fact that a model trained on AC performs badly on CL, and vice versa. We observe that simply combining all these data enhanced the overall performance across datasets, but it still fell short compared to models trained on a single target dataset, indicating the absence of a one-size-fits-all model. To bridge the performance gap between datasets, we introduce Task Embedding (TE) tokens at the input of the decoder, enabling the model to identify the dataset source for each input sample. We analyze the influence of these TEs on the form (words) and content (sound event types) of generated captions. Our final CoNeTTE model, an unbiased CNext-trans with dataset-specific Task Embeddings, achieved the best cross-dataset performance trade-off, and is a path towards a one-size-fits-all model.

After presenting recent AAC works from the literature, we describe our system in Section III and the datasets used to train and evaluate it with the different potential bias problems in Section IV. We detail all the hyperparameters, metrics and first audio tagging results to study data biases in Section V. Then, we present our final results on AC and CL, and compare to state-of-the-art systems. Finally, we report dataset combining experiments and the introduction of our task embedding strategy in Section VI-B.

II. Related Work

To address the AAC task, numerous architectures, data augmentation techniques, and training objectives have been

[1]https://github.com/Labbeti/conette-audio-captioning

proposed in the literature. In this study, we focus on the methods applied to the two main datasets used in our article: AC and CL. We exclude works that use a Reinforcement Learning procedure to artificially boost captioning performance metrics, as they tend to produce low-quality captions with repetitive n-grams [11]. Additionally, ensemble learning results, which combine several model predictions, have also been excluded, as they are commonly employed in AAC challenges.

All recent systems employ a pretrained audio encoder network, combined with a Transformer-like architecture [12] as a word decoder. The authors of [13] employed a test-time data augmentation technique named MM-TTA [14] to average the representations of multiple augmented versions of the same input. Gaussian noise and SpecAugment [15] were used to produce variants, and the method can average the representations at different levels in the network. The system also utilizes mixup [16] data augmentation during training between audio waveforms and spectrograms and concatenates the corresponding caption labels. They employed a full Transformer architecture with an encoder network named Audio Captioning Transformer [17], pretrained on AS for Audio Tagging like most models. Their system currently holds the state-of-the-art position on AC without external training data.

Some studies have proposed using a pretrained decoder to improve the language generation part, as seen in [18], where the authors employed a Transformer decoder named BART [19] to enhance caption quality. However, the audio embedding space is usually too different from the sentence embedding space, leading to the pretrained decoder forgetting most of its previous knowledge. To overcome this, they combined two audio encoders. The first one produces sound event tags detected in the audio file, which are given to the BART input embedding layer and added to the audio embeddings extracted from another audio encoder. This approach is expected to make the BART inputs closer to the ones expected by the pretrained weights. The first encoder is a YAMNet [20] architecture, and the second encoder is the `Wavegram-Logmel-CNN` from the Pretrained Audio Neural Networks study (PANN) [21].

Recently, the authors of [10] introduced a new captioning dataset named WavCaps. Each caption in this dataset is a raw description crawled from various websites and processed with ChatGPT[2]. Their dataset is an order of magnitude larger than other AAC datasets. They trained several models for different audio-language tasks, comparing CNN14 and HTS-AT [22] as pretrained audio encoders, and BART as a pretrained decoder. HTS-AT-BART achieved the current state-of-the-art score on AC with the use of external data.

On CL, the DCASE challenges compile most of the results obtained over the last years[3]. The authors of [23] proposed using a ResNet38 model from PANN as the encoder, with a standard Transformer decoder model. They also crawled audio files from the Web and processed uploaders' raw descriptions to create a larger training corpus for their model, obtaining data from several websites. Additionally, some data augmentation techniques, such as noise and reverberation, were used to improve generalization.

The PaSST-trans [24] system proposed using another pretrained encoder named PaSST [25]. They used a Transformer model with the Patchout method, randomly dropping patches of the spectrograms and flattening the results to obtain shorter sequences as input. Their models were trained on AC, CL, and MACS, with the use of mixup and label smoothing.

The authors of [26] presented a simple model based on CNN14 and a Transformer decoder, trained exclusively on CL, which makes this system similar to our CNN14-trans (see below). Three differences compared to our variant may explain their better results: they employed Stochastic Weight Averaging [27] to average models over several epochs for testing (similar to ensembling), they pretrained the word embeddings, and their decoder has a lower number of trainable parameters.

Finally, the system that achieved first place in the most recent DCASE 2023 Challenge Task 6a[4] is the BEATs+Conformer [28] system. It utilizes BEATs [29] as a pretrained encoder to produce audio features, which ranks among the current state-of-the-art audio tagging models on AS. The audio features are then downsampled by a trainable Conformer network [30] and given to a BART pretrained decoder to generate the captions. Moreover, they used an Instructor-XL model [31], based on a T5 Transformer [32], to generate embeddings from captions and used them as targets with Conformer outputs and InfoNCE [33] loss. Finally, these outputs are used to feed a pretrained BART decoder to generate the captions.

Our AAC system is similar to the ones described above, with two main differences: i) our encoder is a ConvNeXt model, ii) our decoder is a vanilla Transformer decoder randomly initialized. As described in details here-after, this combination allowed us to achieve high performance, while significantly reducing model size compared to many existing AAC systems.

III. Our system: CNext-trans

A. Architecture

Our system is a deep neural network that employs a standard encoder-decoder architecture. The encoder part produces frame-level audio embeddings, and the decoder predicts the next word according to the previous ones and to the audio representation. Each caption is preceded by a Begin-Of-Sentence token (`<bos>`) and followed by an End-Of-Sentence token (`<eos>`). We train our model using the teacher forcing method, which always gives the ground truth previous tokens to the model, in contrast to scheduled sampling methods which use the previous predicted tokens as inputs to the decoder. The model outputs give the probabilities of the next word that are compared with the ground truth next word using a standard Cross-Entropy (CE) loss. During inference, we loop on the decoder forward method to produce each word of the sentence sequentially by adding the most probable next word to the previous words until an EOS token or the maximal number of

[2]https://openai.com/blog/chatgpt/

[3]https://dcase.community/

[4]https://dcase.community/challenge2023/

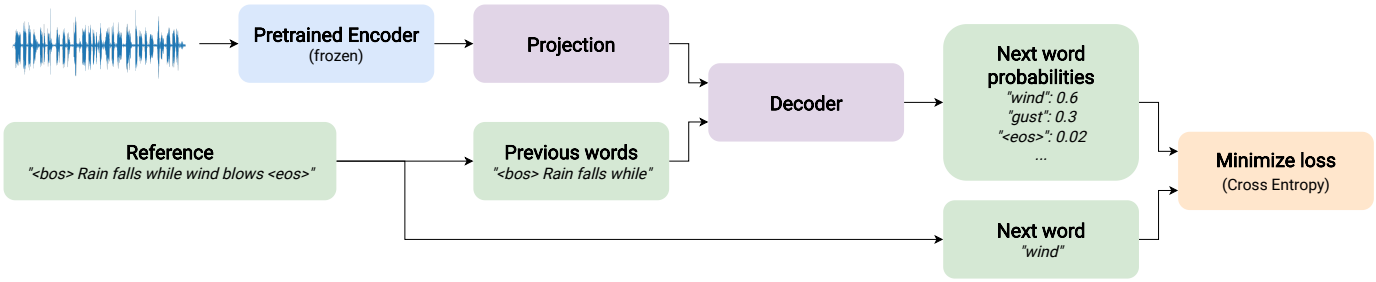


Fig. 1. Overview of our AAC training process. The model is composed by an audio encoder which produces a frame-level audio representation of the audio and a captioning decoder which produces the next word distribution according to the previous words and to the audio. This process is repeated for each word in the reference, except for the Begin-Of-Sentence (<bos>) token.

words is reached. An illustration of the training procedure is giving in Fig. 1.

B. Audio encoder: CNN14 and ConvNeXt-Tiny

In our experiments, we considered two types of pretrained encoders for AAC: the CNN14 model from PANN and ConvNeXt [5] (CNext), a computer vision convolutional neural network that we previously adapted for audio tagging in our work [34]. We observed that a superior encoder generally results in better audio representations, which are crucial for the decoder to produce accurate captions. Both models were pretrained on AS, the largest audio tagging corpus with approximately 2 million labeled audio files. The code and weights of our CNext audio encoder are available[5].

CNN14 is a standard vanilla convolutional-based network consisting of six ConvBlock layers. Each ConvBlock comprises two sequences of convolutions, batch normalization, and ReLU activation, followed by 2-dimensional average pooling. These layers generate a frame-level sequence of embeddings with a size of 2048. We removed the pooling, projection, and classification layers used to predict the AS classes.

The CNext models are based on depthwise separable convolutions [35] (DSC) and inverted bottleneck [36] (IB) layers. DSC involves a sequence of depthwise convolutions that process feature channels separately, followed by a pointwise convolution to mix them. This technique aims to produce results similar to standard convolutional layers while reducing the number of operations to speed up training and mitigate overfitting. The IB layer is a sequence of a pointwise convolution layer that increases the number of channels, followed by a GELU [37] activation, and by another pointwise convolution layer that reverts the number of channels back to its value at the bottleneck input. The output is then added to the original input to create a residual connection, preventing vanishing gradient issues and reducing the number of parameters compared to a standard residual block. All the hyperparameters and training details of the ConvNeXt-Tiny model that we used are described in [34].

We incorporated a small network on top of these encoders to project the audio frame embeddings onto a subspace with the same dimension d_{model} as that of the decoder network. This additional network consists of a 0.5 dropout, a linear layer, a ReLU activation function, and another 0.5 dropout. The linear layer takes embeddings of size 2048 for CNN14 and 768 for CNext.

[5]https://github.com/topel/audioset-convnext-inf

C. Word decoder

The decoder network is a standard Transformer decoder [12] with 6 layers, 8 attention heads per layer, a global embedding size d_{model} set to 256, a layer norm epsilon set to 10^{-5}, a global dropout probability of 0.2 and a feedforward dimension size set to 2048. We used the GELU activation layer in the decoder. These hyperparameters were obtained after a few tests on both data sets.

Like almost all AAC systems, we employed the well-known beam-search algorithm widely used in speech processing [38] during inference to improve subtitle quality and accuracy. We implemented a per-batch version of the beam search algorithm which speeds up the sentence generation by a factor of 10, but requires more memory to run. This algorithm usually leads to generic and repetitive content, which can be penalized by certain metrics. We introduced a constraint during inference to avoid repeats to words that are in previous tokens. We decided to allow repeating a pre-defined list of stop-words from the Natural Language ToolKit (NLTK) [39] package. We found that it can improve diversity and limit repetition while slightly decrease n-gram based metrics. This constraint can forbid a lot of repetitions in the predicted sentences like *"a man speaks while a man speaks"*, but we found that the model can still produce repetitive content with closely related words like in *"children speak and child speaks"*. In addition, we limit the minimal number of tokens predicted to 3 and the maximal number to 20 or 30 during inference to avoid few cases of degenerated sentences.

D. Tackling overfitting

We noticed in our first experiments that our model can easily overfit the training data, even with small networks with less than 10M parameters. We found that using a large weight decay value with the AdamW optimizer [40] drastically overcame this issue [41]. In addition, we searched to apply data augmentation during training. This case is particular for AAC task since a lot of audio transformations (reverb, background noise, pitch shifting...) of the audio can be described in the target captions. We decided to focus on the following three data-agnostic augmentations.

Mixup [16] is used on the decoder inputs (audio embeddings and previous token embeddings) as in [24], [42] to improve the robustness of our model. We also tried to mix target labels, but it did not bring any improvements. The algorithm is shown in Eq. 1 and resumes how mixup is used in our system. x_1 corresponds to an audio embedding with its label y_1 and x_2 is another audio embedding from the current batch, with its label y_2. α is a fixed hyperparameter, W denotes the input word embedding layer and f is the rest of the AAC decoder network.

$$\begin{aligned} \lambda &\sim \text{Beta}(\alpha, \alpha) \\ \lambda &= \max(\lambda, 1-\lambda) \\ x_{mix} &= \lambda x_1 + (1-\lambda) x_2 \\ w_1 &= W(y_{1,prev}) \\ w_2 &= W(y_{2,prev}) \\ w_{mix} &= \lambda w_1 + (1-\lambda) w_2 \\ z_{mix} &= f(x_{mix}, w_{mix}) \\ \mathcal{L} &= \text{CE}(z_{mix}, y_{1,next}) \end{aligned} \tag{1}$$

SpecAugment [15] is applied on the audio frame embeddings, outputted by the audio encoder. We found that using this augmentation on spectrograms or audio embeddings provides similar improvements, but applying it on embeddings allowed us to pre-compute the encoder outputs and drastically accelerate the experiments. We modified the behavior of this augmentation to mask a proportion of the time and feature axes instead of using an absolute mask size. Each axis is masked twice, with a mask size sampled between 0 and 10% of the total axis size (number of time steps or embedding size).

Label smoothing [43] is employed on target captions to limit the maximum probability of the model for each token and reduce overfitting.

IV. Datasets and potential biases

A. Descriptions

In a first series of experiments, we used AC and CL separately. In a second setting, we added training data from Multi-Annotator Captioned Soundscapes (MACS) and WavCaps. We report statistics about the number of word types, tokens, etc. about the training subsets in Table I, to show how these datasets differ in content.

AudioCaps (AC) is the largest human-labeled AAC dataset of originally 51,308 audio files, taken from a subset of the (unbalanced) training subset of AudioSet (AS). AC contains three splits, and because the original YouTube videos are removed for various reasons, our version contains only 46,213 out of 49,838 files in the training subset, 464 out of 495 in the validation subset and 912 out of 975 files in the test subset. The training subset files are described by only one caption per audio, while the validation and test files are described by five captions each.

Clotho (CL) is a smaller dataset containing files extracted from the FreeSound website. The training subset contains 3840 files and the validation and test subsets contain 1045 files each. Unlike some papers in the literature [44], we did not use the validation subset to train our model. All subsets contain five captions per audio and each caption was corrected by a second set of annotators to remove grammatical, subjectivity, fluency, or repetition errors.

Multi-Annotator Captioned Soundscapes (MA) is another AAC dataset containing audio files from the development subset of the TAU Urban Acoustic Scenes 2019 [45] dataset. The sound events have been recorded in different acoustic scenes like airports, public squares, and parks. Each annotator labeled audio files with a predefined set of sound event classes, then with a free-text caption.

WavCaps (WC) is the largest audio captioning dataset, containing 403,050 audio-caption pairs. It is a collection of audio recordings from the AudioSet Strongly Labeled subset, FreeSound, SoundBible and BBC Sound Effects. Captions have been generated by a ChatGPT model, using the audio event classes for the AudioSet subset or the original human descriptions for the other subsets. WC features a large diversity in audio length: from 1 second to 18 hours long.

We used our own source code to download and load the dataset files, available as a Python package named aac-datasets[6].

TABLE I
STATISTICS FOR AC, CL, MA AND WC SUBSETS USED FOR TRAINING.

	AC	CL	MA	WC
Sample rate (Hz)	32000	44100	48000	32000
Audio duration range (s)	0.5-10	15-30	10	1-67109
Nb audio	46,213	3839	3930	403,050
Audio size (h)	126.6	24.0	10.9	7563.3
Vocabulary size	4585	4369	2721	24600
Nb words	401,650	217,360	159,879	3,161,823
Caption length range	2-40	8-20	2-40	2-38
Caption length mean	8.7	11.3	9.3	7.8
Nb captions per audio	1	5	2-5	1

B. Potential training data biases in AAC

We identified two types of potential biases when carrying out AAC experiments on AC and CL, which concern either the pretrained audio encoders or the whole AAC systems. These biases may arise because of overlaps within the data sources, either between AAC and Audio Tagging (AT) datasets, or between AAC datasets. We report the overlap proportions between datasets CL, AC, WC, AS and FSD50K in Table II.

Most AAC systems use audio encoders pretrained to perform AT, either on AS or on the audio tagging dataset FSD50K [46]. However, the whole AC dataset is actually a subset of the AS training subset, which means that an encoder pretrained on AS has already seen 100% of the audio files of AC, even the validation and testing ones. This implies that the encoder already "knows" the sound events of AC. This bias concerns all audio-language tasks (audio captioning and audio retrieval) involving AS and AC in their procedure. Figure 2 summarizes this data bias (or data leak) that we want to highlight.

[6]https://github.com/Labbeti/aac-datasets

To a much smaller extent, the same problem occurs when pretraining on FSD50K and running AAC experiments on CL, since about 5% of the CL files are shared with FSD50K's training subset (extracted from the FreeSound Website).

The second type of bias, which may impact whole AAC systems, involves the WC dataset. WC comprises files from both FreeSound and AS, which are the data sources of CL and AC, respectively. One has to take care of removing these overlaps (18% in common with AC, and 89% in common with CL), when pretraining an audio encoder (first bias mentioned above), and when training an AAC system.

There could also be overlaps between FSD50K or WC with the CL private subsets used in the DCASE challenge task 6a, which means that the results of the models trained on these datasets could be overestimated. These potential overlaps remain unknown to us, since the audio file IDs are not available.

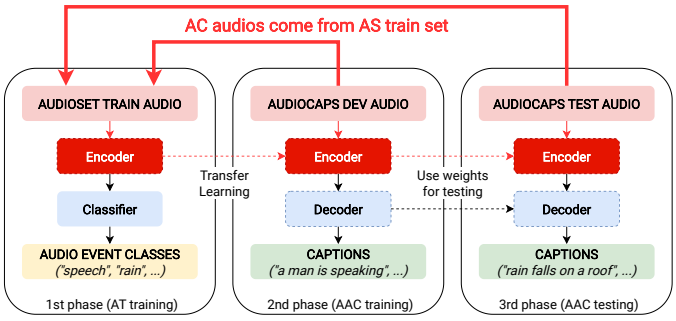


Fig. 2. Illustration of the bias that may occur when using an audio encoder pretrained on AS for experiments on AC. All the subsets of AC come from the AS training subset, which means that the encoder part of the AAC systems were trained on the validation and testing data for audio tagging, a task closely related to AAC.

TABLE II
OVERLAP PROPORTIONS BETWEEN AAC/AT DATASETS.

Dataset A	Dataset B	Overlap (%)	B labels type
AC	AS-train	100.0	Tags
CL	FSD50K-train	5.4	Tags
AC	WC	17.6	Captions
CL	WC	89.0	Captions

C. Task Embedding

After obtaining our results on AC and CL without external data, we attempted to improve performance by adding more training data. Surprisingly, we observed a decrease in results. Previous work [47] noted that the sound events and writing styles of AC and CL can form distinct domains, which can perturb models trained on both datasets compared to those trained separately. Many AAC studies [24], [10], [48] handle external data by splitting the training into two phases: the first one involves training on all available captioning datasets, while the second one entails fine-tuning on the target dataset only (AC or CL).

However, we believe that a single AAC system should be capable of leveraging multiple captioning datasets, given a hint to the model. To achieve this, we introduced a Task Embedding (TE) tag as input to the decoder, dependent on the audio dataset source, such as `<bos_ac>`, `<bos_cl>`, etc. This tag replaces the `<bos>` token used at the sentence's beginning and enables the model to generate an output closer to the desired writing style. We initially employed this method in our participation in the DCASE Challenge 2023 [49], where our best system achieved third place. In this study, we expanded on this approach by using more datasets and TE tokens. Interestingly, another DCASE participant [50] also utilized a TE approach to leverage different datasets (CL, AC, and AS), employing a system based on the speech automatic recognition system Whisper [51]. However, their system performed worse and was ranked seventh in the challenge.

To test this method, we attempted to add all publicly available AAC datasets: AC, CL, MA, and WC. We excluded overlapping data between AC and WC and between CL and WC to avoid biases. Additionally, we removed audio files lasting for more than 30 seconds in WC, as this is the maximum length of the test audio files of CL. As WC contains four different sources, we added a task embedding tag for each of them, resulting in seven tasks for all datasets. The concatenation of all the audio captioning datasets (CL+AC+MA+WC) yielded 316,122 training audio files. Moreover, we decided to balance data with the target dataset (AC or CL). For example, on CL, we took the 3,840 files of CL and randomly selected another 3,840 files from the other datasets, resulting in 7,680 audio training seen files per epoch. For AC, we used their 46,213 files and an equal number from other datasets to train with 92,426 files per epoch. We tested other balancing strategies, but this approach yielded the best results.

V. EXPERIMENTAL SETUP

A. Data pre-processing

All the captions are written in lowercase, and all punctuation characters are removed. Sentences are tokenized using spaCy [52]. For CL and MA, we resample all audio files from 44.1 kHz and 48 kHz to 32 kHz to match the input sampling rates of the encoders. During training, for CL, we randomly select one of the five captions for each audio file at each epoch in the training subset.

As for AC, we manually corrected a portion of the training captions. This subset contains various mistakes, such as typographic errors (e.g., *"Continous"* corrected to *"Continuous"*), named entities (e.g., *"Michael Jackson"*), invalid descriptions (e.g., *"Video is unplayable"*), speech content (e.g., *"A large crowd of people chanting "USA" [...]"*), or grammar errors (e.g., *"A engine"* corrected to *"An engine"*). Additionally, we excluded captions containing more than 40 words. To create a second version of the AC training subset, we manually fixed 968 captions and deleted 28 files.

B. Metrics

Evaluating AAC system outputs is a challenging task, as the predicted candidates should contain the same audio events as the references (ground truth), but not necessarily described in the exact same way. Historically, machine translation metrics

were used for system comparison, but they primarily rely on n-gram overlapping, which diminishes their correlation with human judgments in captioning evaluation.

The CIDEr-D [53] metric was developed for image captioning and takes word frequency into account. It calculates the cosine similarity of the TF-IDF scores for common n-grams in both candidates and references. SPICE [54] proposes a comparison of scene graphs containing semantic propositions extracted from sentences using a parser, handcrafted grammar, and custom rules. However, CIDEr-D tends to overvalue candidates containing infrequent n-grams, even if they lack syntactic correctness [11], while SPICE tends to give zero scores when the extracted propositions do not match [55]. SPIDEr [56] was introduced to consider both metrics by averaging their scores. Nevertheless, as CIDEr-D values range from 0 to 10 and SPICE values range from 0 to 1, CIDEr-D has a more significant influence on the SPIDEr score.

In several recent studies, SPIDEr has faced criticism, for example for its lack of sensitivity to repetitions [57], [58], [59]. To establish a more robust evaluation of AAC systems, the authors of [55] created FENSE, a metric based on two pretrained models for comparing sentence embeddings rather than n-grams. FENSE employs two models: a Sentence-BERT (SBERT) model and a Fluency Error Detector model. The SBERT model is trained to generate fixed-size embeddings that can be compared using cosine similarity (referred to as SBERT-sim). The Fluency Error Detector model is trained to detect common mistakes made by AAC systems, such as repetitive n-grams, incomplete sentences, repetitive adverbs, missing conjunctions, and missing verbs. The FENSE score is equal to SBERT-sim when no error is detected by the Fluency Error Detector model; otherwise, the score is divided by 10.

In this work, we report SPIDEr, FENSE and the number of unique words used (named #Words or sometimes Uniq. in the literature). We have regrouped all the code needed to compute these metrics into our package aac-metrics[7], which is publicly available online.

C. Hyperparameters

We provide a detailed account of all the training hyperparameter values in Table III. Throughout our experiments, we validate and select our models using the FENSE score on the validation subset. Unlike the validation likelihood loss, this metric evaluates the sentence generation of our models, and we have found it to be more stable than the loss or SPIDEr values. The latter tends to vary drastically among different seeds due to its sensitivity to predicted n-grams [59].

The weight decay is not applied to the bias weights of the networks. To prevent models from collapsing, we used gradient clipping [60]. During the AAC training phase, we freeze the encoder, enabling us to pre-compute audio embeddings and significantly reducing the training time. On a single GPU V100 with 32 GB of memory, AAC experiments take only one hour on AC and three hours on CL. The learning rate is decreased during training at the end of each epoch k using a cosine scheduler rule 2:

[7]https://github.com/Labbeti/aac-metrics

TABLE III
TRAINING AND DECODING HYPERPARAMETERS PER DATASET.

Name	Value	
	AC	CL
Batch size	512	
Optimizer	AdamW	
Initial learning rate (lr_0)	$5 \cdot 10^{-4}$	
β_1	0.9	
β_2	0.999	
ϵ	10^{-8}	
Weight decay	2	
Gradient clip norm type	ℓ^2	
mixup param. (α)	0.4	
Min prediction size	3	
Nb. Epochs (K)	100	400
Gradient clip norm value	10	1
Label smoothing	0.1	0.2
Max prediction size	30	20
Beam size	2	3

$$\text{lr}_k = \frac{1}{2}\left(1 + \cos\left(\frac{k\pi}{K}\right)\right)\text{lr}_0 \tag{2}$$

D. Audio tagging results on AudioCaps with and w/o bias

In this subsection, we evaluate the impact of the training data bias on AC first in terms of audio tagging. We pretrained CNN14 and CNext encoders on AS with and without the AC training, validation, and testing files. Table IV reports the mean Average Precision (mAP) scores of these classifiers on the AS evaluation subset, as well as on the validation and testing subset files of AC. We compare the CNN14 original weights from PANN with our own trained weights (denoted as CNN14*), as well as with our own encoder CNext.

TABLE IV
TAGGING MAP SCORES ON AS EVAL, AC VAL AND AC TEST.

Encoder	Train data	AS-eval	AC-val	AC-test
CNN14 [21]	AS	.431	.717	.647
CNN14*	AS	.441	.755	.727
CNext	AS	**.471**	**.774**	**.749**
CNN14*	AS-AC	.434	.642	.537
CNext	AS-AC	.465	.669	.585

The audio tagging results show that, as expected, models trained on the full AS (with bias) achieved significantly higher mAP scores than without bias (lines denoted AS-AC in the table). CNext's mAP on AC-test decreased from 0.749 to 0.585, representing a relative decrease of -21.9%. This suggests that the model already possesses good knowledge of the sound events in AC-val and AC-test subsets. However, the score on AS-eval remains stable, with only a slight decrease from 0.471 to 0.465 (-1.3% relative drop). Similar trends are observed with CNN14*, indicating the potential importance of considering this bias when evaluating AAC systems. In the next section, we report AAC results, and we will see that the

TABLE V
AAC RESULTS ON AC AND CL TESTING SUBSETS. ENC* MEANS THAT THERE IS A POTENTIAL BIAS IN THE ENCODER. CNN14 INDICATES THAT WE USED THE ORIGINAL WEIGHTS FROM PANN, WHILE CNN14* DENOTES OUR OWN TRAINED WEIGHTS. WC^{-AC} REFERS TO THE WAVCAPS DATASET WITHOUT SOURCES OVERLAPPING WITH EITHER THE AC VALIDATION OR TESTING SUBSETS. WC^{-CL} DENOTES THE WAVCAPS DATASET WITHOUT SOURCES OVERLAPPING WITH CL VALIDATION AND TESTING SUBSETS.

Test data	System	Pretraining data	Training data	Biased	SPIDEr	FENSE	#Words	Trainable params	Frozen params
AC	Cross-referencing	∅	∅	N/A	.559	.680	944.0	0	0
	HTSAT-BART [10]	AS	AC+WC^{-AC}	Enc*	.485	N/A	N/A	171M	0
	Multi-TTA [13]	AS	AC	Enc*	.475	N/A	N/A	108M	0
	PYB [18]	AS	AC	Enc*	.465	N/A	N/A	400M	0
	CNN14-trans (ours)	AS	AC	Enc	.443	.615	383.2	12.3M	75.5M
	CNN14*-trans (ours)	AS	AC	Enc	.456	.625	390.4	12.3M	75.5M
	CNext-trans (ours)	AS	AC	Enc	**.495**	.643	393.0	12.0M	28.2M
	CNN14*-trans (ours)	AS-AC	AC	No	.439	.607	354.0	12.3M	75.5M
	CNext-trans (ours)	AS-AC	AC	No	.466	.633	396.4	12.0M	28.2M
CL	Cross-referencing	∅	∅	N/A	.567	.574	1818.2	0	0
	BEATs+Conformer [28]	AS	CL+AC	No	**.326**	N/A	N/A	127M	1.5B
	CNN14-BART [10]	AS	CL+WC^{-CL}	No	.310	N/A	N/A	219M	0
	ResNet38-trans [23]	AS	CL+AC+4 others	N/A	.308	N/A	N/A	81M	0
	PaSST-trans [24]	AS	CL+AC+MA	No	.296	.511	N/A	119M	441M
	CNN14-trans [26]	AS	CL	No	.285	N/A	N/A	8M	75.5M
	CNN14-trans (ours)	AS	CL	No	.265	.482	500.2	12.2M	75.5M
	CNN14*-trans (ours)	AS	CL	No	.274	.491	545.2	12.2M	75.5M
	CNext-trans (ours)	AS	CL	No	.299	.512	636.8	11.9M	28.2M
	CNN14*-trans (ours)	AS-AC	CL	No	.259	.477	522.6	12.2M	75.5M
	CNext-trans (ours)	AS-AC	CL	No	.301	.516	628.2	11.9M	28.2M

impact of this specific bias on AC is somehow smaller than in audio tagging, with SPIDEr score relative reductions of about -5%.

VI. AUDIO CAPTIONING RESULTS ON AC AND CL

In this section, we discuss the results of our systems with and without bias, and the impact of using external data in the first subsection. We then explore the effect of adding new training data using TEs or not in the second subsection. Finally, we study the impact of TEs on the generated captions.

A. Results without external data

Table V presents results on the AC and CL datasets. "Human" top-line scores are computed by randomly excluding one of the five references for each audio file and using it as a candidate. This provides an inter-agreement score (also called cross-referencing score) between the ground truth references, which is repeated five times and averaged for each subset. These scores offer an idea of the upper bound for each metric. We also report results obtained by state-of-the-art methods presented in section II, and our own results. All of our results are averaged over the same five initialization seeds.

On AC, we report results from recent systems from the literature, in which we identified a presumable bias due to their pretrained encoders, since they did not mention that the AC files were removed from AS during the AT pretraining of their encoders. Regarding our systems, the CNext-trans model outperformed CNN14-trans in all contexts, as we expected given its higher performance in audio tagging. Our biased CNext-trans model outperformed the previous state-of-the-art method by 1.0% absolute, without using any external data and with an order of magnitude fewer trainable parameters. However, the unbiased CNext-trans model, which did not use the AC audio files in pretraining (AS-AC), experienced a 2.9% absolute points drop, representing only a -5.9% relative decrease. This indicates that the data bias in this pretrained model still had an impact on performance for the AAC task, related to the AT performance reduction of the model (which was a -21.9% relative drop). The FENSE scores seem to be correlated with the SPIDEr values and demonstrate that the CNext encoder provided better representations to the decoder part, both with and without data bias. The number of words used varies from 353 to 396 words, but this is not always correlated with SPIDEr.

On CL, the CNext-trans model continues to outperform other models and achieves a performance close to CNN14-BART, despite having fewer parameters and training data. We have attained the state-of-the-art score for models that do not use any external data. We also observe that the richer representation provided by CNext allows the decoder to produce captions with higher word diversity (545 word types with CNN14-trans compared to 636 with CNext-trans).

Furthermore, we note that our SPIDEr score on AC is close to the cross-reference one with only 6.4% absolute difference, but the SPIDEr scores on CL remains much lower than the cross-reference score, with a 26.2% absolute difference. However, the FENSE scores are both close (3.7% and 5.7%), indicating that the model produces captions that are semantically close to the references but struggle to find

TABLE VI
AAC RESULTS ON AC AND CL TESTING SUBSETS. A BALANCED DATASET REPRESENTS 50% OF THE EXAMPLES SEEN DURING AN EPOCH TO FOCUS TRAINING ON THAT SPECIFIC DATASET. WC* DENOTES THE WAVCAPS DATASET WITHOUT OVERLAPPING SOURCES WITH AC AND CL TRAINING, VALIDATION AND TESTING SUBSETS, AND WITHOUT AUDIO LASTING FOR MORE THAN 30 SECONDS.

System	TE	Training data	Balanced data	AC-test			CL-test		
				SPIDEr	FENSE	#Words	SPIDEr	FENSE	#Words
CNext-trans	No	AC	N/A	.466	**.633**	396.4	.146	.465	401.4
CNext-trans	No	AC+CL+MA+WC*	AC	.456	.627	**396.8**	.191	.484	**472.4**
CoNeTTE	Yes	AC+CL+MA+WC*	AC	**.468**	.630	379.8	**.252**	**.498**	467.4
CNext-trans	No	CL	N/A	.231	.521	**525.2**	.301	.516	628.2
CNext-trans	No	AC+CL+MA+WC*	CL	.364	.591	376.0	.295	.510	589.2
CoNeTTE	Yes	AC+CL+MA+WC*	CL	**.441**	**.609**	331.2	**.305**	**.517**	**639.0**

the correct n-grams or propositions. AC is much less diverse than CL in terms of vocabulary (944 and 1818 word types, respectively), and the SPIDEr score is mostly influenced by several predictions that almost fully match the references, resulting in a higher SPIDEr score.

B. Improved results with Task Embedding

To create a general-purpose AAC system, we evaluated different training methods for our CNext-trans model on the two evaluation sets, as shown in Table VI. The first approach involved training our model separately on each dataset. Then, we attempted to add other datasets and balance the target dataset. Finally, we introduced the TE method to improve overall performance.

When we added all the external data described in IV-C without TE, the results of our system decreased on both datasets, primarily due to the different writing styles present in the combined data. It became evident that a model trained on one specific dataset performed poorly on the other. For example, a model trained on AC achieved only a 14.6% SPIDEr score on CL. Similarly, a model trained on all the datasets without TE also performed inadequately, with a 19.1% SPIDEr score.

However, when we introduced TE, the results of CoNeTTE improved slightly, depending on the balanced dataset. More notably, when examining the performance on the non-balanced datasets, the performance showed significant improvement with TE, indicating that the model made better use of external data. As a result, the best global model performance was achieved when using CoNeTTE trained with CL balanced data.

C. Does Task Embedding change form or content, or both?

To assess the ability of CoNeTTE to produce captions in different writing styles, we tested its ability to generate captions on the CL testing subset with two TE tokens: the CL and the AC ones. Specifically, Tables VII and VIII present various outputs from our model with these two different TEs. These examples demonstrate that the model generates different captions depending on the TE provided. Additionally, we report the SPIDEr, FENSE, vocabulary size, and average sentence length in Table IX for both TEs and testing subsets. For instance, on the CL test subset, the sentence length decreased from 10.8 to 7.2, and the vocabulary size dropped from 639.0 to 412.2 words when using the CL TE versus the AC TE. As shown by the decrease in the SPIDEr score from .305 to .227 when using the AC TE, there was a slight drop in performance when using the wrong TE for a dataset. However, despite these changes, we calculated the SPIDEr between the two different outputs and found it to be relatively high at 1.148, indicating that the sentences still contain similar content.

To further explore the difference between the different TE, we reported as supplementary material the distributions of unigrams and trigrams of stemmed words, and of pos-tags in the candidates and references of AC and CL. We observed that the number of prepositions and conjunctions was twice as high when using the CL TE compared to the AC TE on AC and CL test subsets, suggesting that the captions adopt

TABLE VII
CAPTIONS GENERATED WITH TWO DIFFERENT TES WITH THEIR CORRESPONDING REFERENCES ON THE AUDIO FILE ID "35b9BSmN5JM".

Task	Candidates	SPIDEr	FENSE
AC	A vehicle engine idling and revving	**.283**	**.592**
CL	An engine is idling and revving up and down	.249	.577
References			
Loud vibrating followed by revving			
Truck in idle mode, door closing, engine revving and accelerating			
A wooden thud as an idle car engine runs then accelerates			
A motor vehicle accelerates and revs			
An engine running			

TABLE VIII
CAPTIONS GENERATED WITH TWO DIFFERENT TES WITH THEIR CORRESPONDING REFERENCES ON THE AUDIO FILE "Diving Bell 1.wav".

Task	Candidates	SPIDEr	FENSE
AC	A musical instrument is playing a note	.037	.325
CL	A gong is struck and echoes in a steady rhythm	**.198**	**.588**
References			
A bell is struck by a mallet, and the noise resonates for some time.			
A heavy chime is struck and rings loudly at an even tone.			
A mallet strikes a bell and the sound resonates for a time.			
A single bell is sounded and its reverberations felt all around.			
single bell sound followed by its vibration sound			

TABLE IX
VOCABULARY AND SENTENCE AVERAGE SIZES ON AC AND CL TEST SUBSETS, WITH AC AND CL TES AND THE CONETTE MODEL TRAINED WITH CL BALANCED DATA.

Test	Task	SPIDEr	FENSE	#Words	#Sent
AC	AC	.441	.609	331.2	7.5
AC	CL	.307	.576	517.2	10.8
CL	AC	.227	.493	412.2	7.2
CL	CL	.305	.517	639.0	10.8

different formulations. The TE also appears to imitate some specific trigrams like *"followed by a"* from AC and *"in the background"* from CL. Moreover, the trigrams corresponding to audio events are not always in the same ranking order when using different TEs, indicating that TEs also have an impact on the nature of the sound events described in the captions.

VII. CONCLUSIONS

In this article, we introduced our novel CNext-trans model and its refined version, CoNeTTE, which demonstrated state-of-the-art results on AudioCaps with 49.5% SPIDEr and closely approached the state-of-the-art on Clotho with 30.1% SPIDEr, all while utilizing fewer parameters than existing models. Additionally, we shed light on potential biases when using AudioCaps and showcased that employing an unbiased pretrained audio encoder has an adverse impact on performance, with 46.6% SPIDEr.

Notably, in CoNeTTE, we introduced Task Embedding (TE) tags as input to the model, allowing for the combination of several AAC datasets during training. The utilization of TEs effectively addresses the performance gaps observed across AAC datasets by efficiently merging data rather than relying solely on simple concatenation. However, we also uncovered that dataset balancing during training significantly influences the final performance, implying that the one-size-fits-all model still retains some dependency on a specific dataset.

To provide a comprehensive analysis, we presented examples and insights into the impact of TEs on the generated captions, supplemented by word stem and POS tag distributions in the supplementary material. Nevertheless, further investigation in this area would be valuable and intriguing. The combination of AAC datasets warrants additional exploration, either through discovering an optimal balancing policy across datasets or by extending the application of TEs beyond what was used in this study.

ACKNOWLEDGMENTS

This work was partially supported by the Agence Nationale de la Recherche the LUDAU (Lightly-supervised and Unsupervised Discovery of Audio Units using Deep Learning) project (ANR-18-CE23-0005-01) and the ANR-3IA Artificial and Natural Intelligence Toulouse Institute. This work was granted access to the HPC resources of IDRIS under the allocation 2022-AD011013739 made by GENCI.

REFERENCES

[1] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *2015 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2015, pp. 3156–3164.

[2] K. Drossos, S. Adavanne, and T. Virtanen, "Automated audio captioning with recurrent neural networks," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2017, pp. 374–378.

[3] X. Mei, X. Liu, M. D. Plumbley, and W. Wang, "Automated audio captioning: an overview of recent progress and new challenges," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2022, no. 1, p. 26, Oct 2022.

[4] X. Xu, M. Wu, and K. Yu, "A comprehensive survey of automated audio captioning," *ArXiv*, vol. abs/2205.05357, 2022.

[5] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *2022 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 11 966–11 976.

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

[7] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: an audio captioning dataset," in *2020 IEEE International Conf. on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*. IEEE, 2020, pp. 736–740.

[8] 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 *2017 IEEE International Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. New Orleans, LA: IEEE, Mar. 2017, pp. 776–780.

[9] I. Martin and A. Mesaros, "Diversity and bias in audio captioning datasets," in *Proc. of the 6th Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE2021)*, Barcelona, Spain, November 2021, pp. 90–94.

[10] X. Mei, C. Meng, H. Liu, Q. Kong, T. Ko, C. Zhao, M. D. Plumbley, Y. Zou, and W. Wang, "WavCaps: A ChatGPT-assisted weakly-labelled audio captioning dataset for audio-language multimodal research," *arXiv preprint arXiv:2303.17395v1*, 2023.

[11] X. Mei, Q. Huang, X. Liu, G. Chen, J. Wu, Y. Wu, J. Zhao, S. Li, T. Ko, H. L. Tang, X. Shao, M. D. Plumbley, and W. Wang, "An encoder-decoder based audio captioning system with transfer and reinforcement learning," 2021.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[13] E. Kim, J. Kim, Y. Oh, K. Kim, M. Park, J. Sim, J. Lee, and K. Lee, "Improving audio-language learning with mixgen and multi-level test-time augmentation," 2022.

[14] I. Shin, Y.-H. Tsai, B. Zhuang, S. Schulter, B. Liu, S. Garg, I. S. Kweon, and K.-J. Yoon, "MM-TTA: Multi-modal test-time adaptation for 3d semantic segmentation," in *CVPR*, 2022.

[15] 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 *Interspeech 2019*. ISCA, sep 2019.

[16] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *6th International Conf. on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conf. Track Proc.* OpenReview.net, 2018.

[17] X. Mei, X. Liu, Q. Huang, M. D. Plumbley, and W. Wang, "Audio captioning transformer," in *Proc. of the 6th Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE2021)*, Barcelona, Spain, November 2021, pp. 211–215.

[18] F. Gontier, R. Serizel, and C. Cerisara, "Automated audio captioning by fine-tuning bart with audioset tags," in *Proceedings of the 6th Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE2021)*, Barcelona, Spain, November 2021, pp. 170–174.

[19] 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. of the 58th Annual Meeting of the Association*

for Computational Linguistics. Online: Association for Computational Linguistics, Jul. 2020, pp. 7871–7880.

[20] M. Plakal and D. Ellis, "YAMNet."

[21] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, and M. 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, 01 2020.

[22] K. Chen, X. Du, B. Zhu, Z. Ma, T. Berg-Kirkpatrick, and S. Dubnov, "Hts-at: A hierarchical token-semantic audio transformer for sound classification and detection," in *ICASSP 2022 - 2022 IEEE International Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 646–650.

[23] 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. of the 6th Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE2021)*, Barcelona, Spain, November 2021, pp. 6–10.

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

[25] K. Koutini, J. Schlüter, H. Eghbal-zadeh, and G. Widmer, "Efficient Training of Audio Transformers with Patchout," in *Proc. Interspeech 2022*, 2022, pp. 2753–2757.

[26] H. Won, B. Kim, I.-Y. Kwak, and C. Lim, "CAU submission to DCASE 2021 task6: Transformer followed by transfer learning for audio captioning," DCASE2021 Challenge, Tech. Rep., July 2021.

[27] P. Izmailov, D. Podoprikhin, T. Garipov, D. P. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," in *Conf. on Uncertainty in Artificial Intelligence*, 2018.

[28] S.-L. Wu, X. Chang, G. Wichern, J.-w. Jung, F. Germain, J. L. Roux, and S. Watanabe, "BEATs-based audio captioning model with INSTRUCTOR embedding supervision and ChatGPT mix-up," DCASE2023 Challenge, Tech. Rep., May 2023.

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

[30] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," 2020.

[31] H. Su, W. Shi, J. Kasai, Y. Wang, Y. Hu, M. Ostendorf, W. tau Yih, N. A. Smith, L. Zettlemoyer, and T. Yu, "One embedder, any task: Instruction-finetuned text embeddings," 2023.

[32] 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," *J. Mach. Learn. Res.*, vol. 21, no. 1, jan 2020.

[33] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018.

[34] T. Pellegrini, I. Khalfaoui-Hassani, E. Labbé, and T. Masquelier, "Adapting a ConvNeXt model to audio classification on AudioSet," in *Accepted to Interspeech*. ISCA, sep 2023.

[35] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[36] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[37] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," 2016.

[38] B. T. Lowerre, "The harpy speech recognition system." Ph.D. dissertation, Carnegie Mellon University, USA, 1976.

[39] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[40] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *7th International Conf. on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[41] E. Labbé, J. Pinquier, and T. Pellegrini, "Multitask learning in audio captioning: a sentence embedding regression loss acts as a regularizer," 2023.

[42] D. Takeuchi, Y. Koizumi, Y. Ohishi, N. Harada, and K. Kashino, "Effects of word-frequency based pre- and post- processings for audio captioning," 2020.

[43] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826.

[44] X. Mei, X. Liu, J. Sun, M. D. Plumbley, and W. Wang, "Diverse audio captioning via adversarial training," 2022.

[45] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," in *Proc. of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, November 2018, pp. 9–13.

[46] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra, "Fsd50k: An open dataset of human-labeled sound events," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 30, p. 829–852, dec 2021.

[47] J. Berg and K. Drossos, "Continual learning for automated audio captioning using the learning without forgetting approach," in *Proc. of the 6th Detection and Classification of Acoustic Scenes and Events 2021 Workshop (DCASE2021)*, Barcelona, Spain, November 2021, pp. 140–144.

[48] 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-traing and word selection methods," DCASE2021 Challenge, Tech. Rep., July 2021.

[49] E. Labbé, T. Pellegrini, and J. Pinquier, "Irit-ups dcase 2023 audio captioning and retrieval system," DCASE2023 Challenge, Tech. Rep., May 2023.

[50] M. Kadlčík, A. Hájek, J. Kieslich, and R. Winiecki, "A whisper transformer for audio captioning trained with synthetic captions and transfer learning," DCASE2023 Challenge, Tech. Rep., May 2023.

[51] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *Proc. ICML*. PMLR, 2023, pp. 28 492–28 518.

[52] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, "spaCy: Industrial-strength Natural Language Processing in Python," 2020.

[53] R. Vedantam, C. L. Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," in *2015 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 4566–4575.

[54] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "Spice: Semantic propositional image caption evaluation," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 382–398.

[55] Z. Zhou, Z. Zhang, X. Xu, Z. Xie, M. Wu, and K. Q. Zhu, "Can audio captions be evaluated with image caption metrics?" in *ICASSP 2022 - 2022 IEEE International Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 981–985.

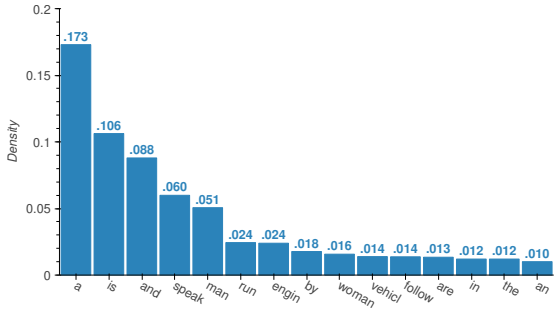
[56] S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy, "Improved image captioning via policy gradient optimization of spider," in *IEEE International Conf. on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 873–881.

[57] I. Martín-Morató, M. Harju, and A. Mesaros, "A summarization approach to evaluating audio captioning," in *Proc. of the 7th Detection and Classification of Acoustic Scenes and Events 2022 Workshop (DCASE2022)*, Nancy, France, November 2022.

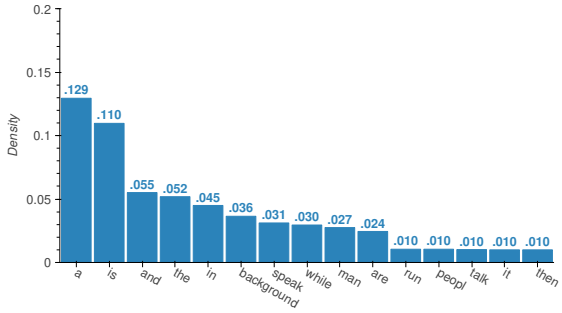
[58] F. Gontier, R. Serizel, and C. Cerisara, "SPICE+: Evaluation of Automatic Audio Captioning Systems with Pre-Trained Language Models," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[59] E. Labbé, T. Pellegrini, and J. Pinquier, "Is my Automatic Audio Captioning System so Bad? SPIDEr-max: A Metric to Consider Several Caption Candidates," in *Proc. of the 7th Detection and Classification of Acoustic Scenes and Events 2022 Workshop (DCASE2022)*, Nancy, France, November 2022.

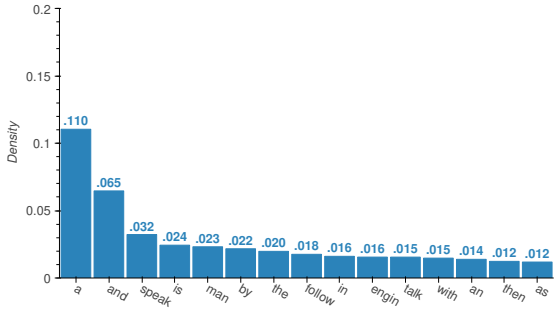
[60] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. of the 30th International Conf. on International Conf. on Machine Learning - Volume 28*, ser. ICML'13. JMLR.org, 2013, p. III–1310–III–1318.



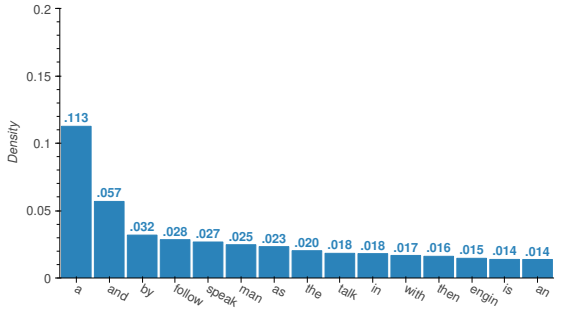
(a) Distribution of the unigrams in the candidate captions on AC-test with AC TE.



(b) Distribution of the unigrams in the candidate captions on AC-test with CL TE.

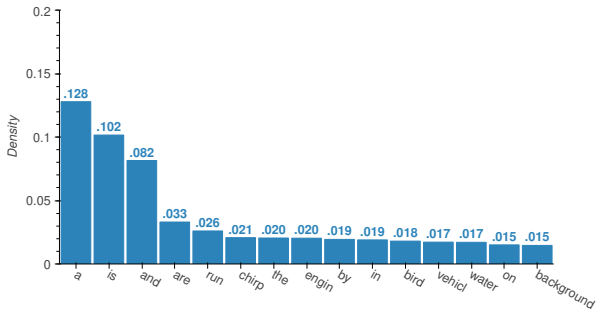


(c) Distribution of the unigrams in the reference captions on AC-train.

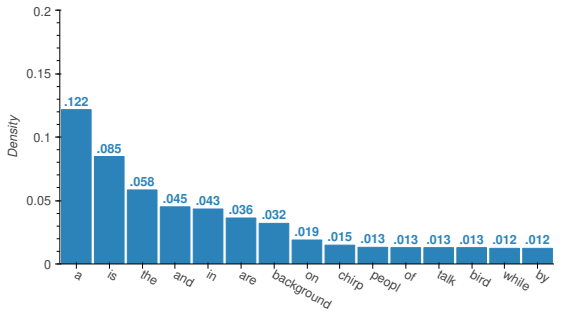


(d) Distribution of references unigrams on AC-test.

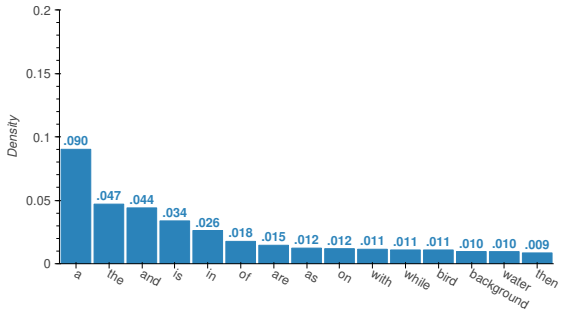
Fig. 3. Distributions of unigrams on AC-test and AC-train.



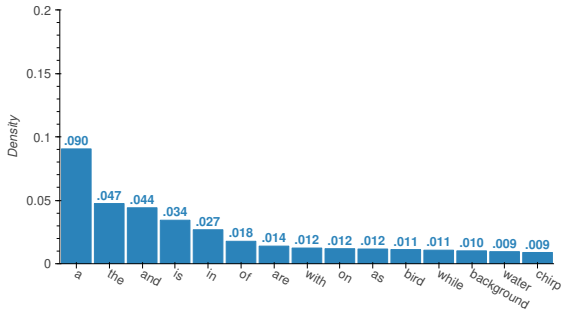
(a) Distribution of candidates unigrams on CL-eval with AC TE.



(b) Distribution of candidates unigrams on CL-eval with CL TE.

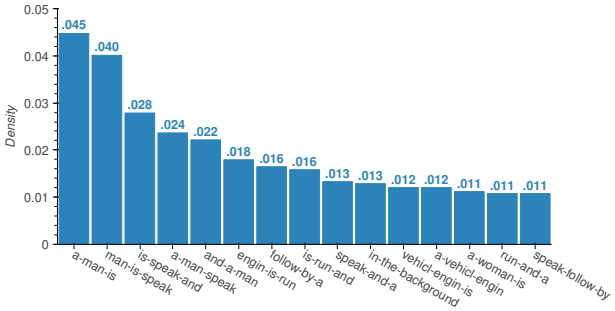


(c) Distribution of references unigrams on CL-dev.

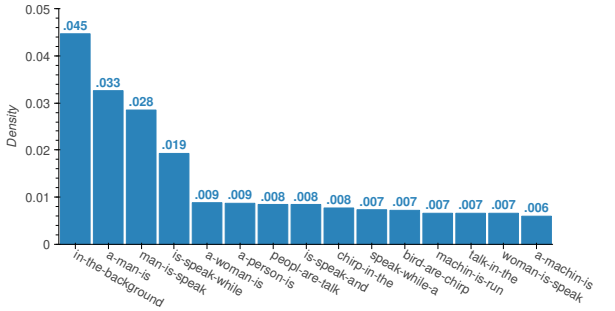


(d) Distribution of references unigrams on CL-eval.

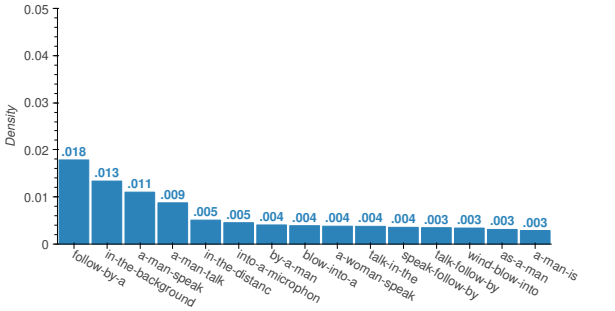
Fig. 4. Distributions of unigrams on CL-eval and CL-dev.



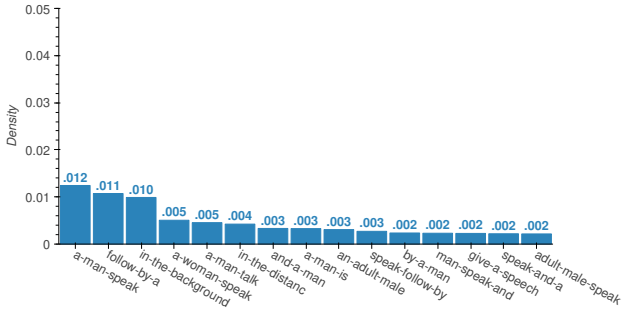
(a) Distribution of candidates trigrams on AC-test with AC TE.



(b) Distribution of candidates trigrams on AC-test with CL TE.

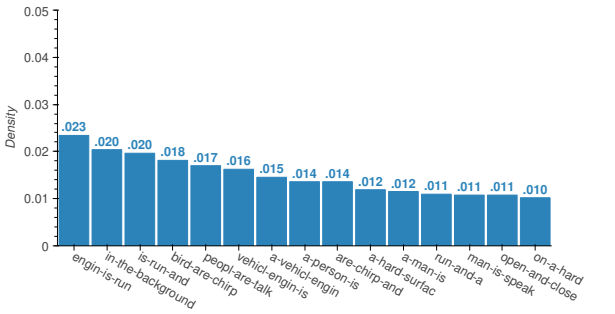


(c) Distribution of references trigrams on AC-train.

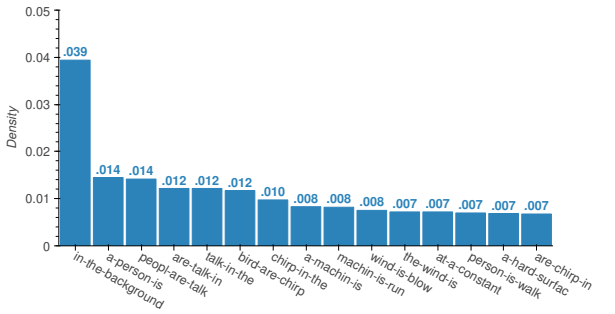


(d) Distribution of references trigrams on AC-test.

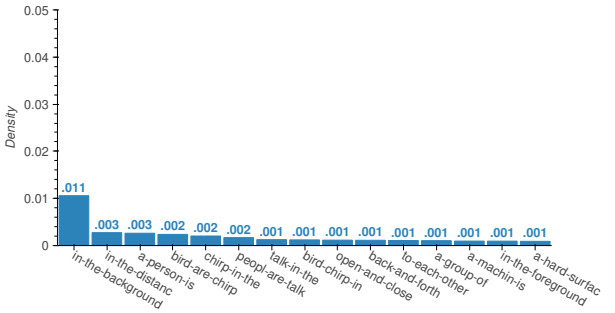
Fig. 5. Distributions of trigrams on AC-test and AC-train.



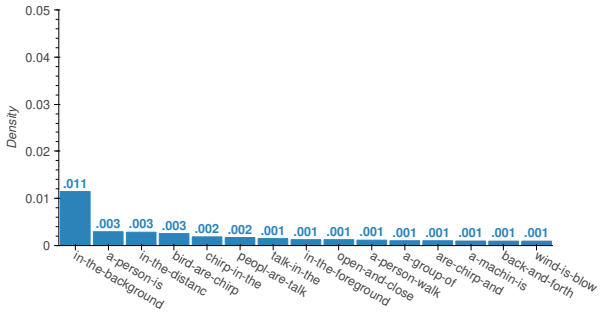
(a) Distribution of candidates trigrams on CL-eval with AC TE.



(b) Distribution of candidates trigrams on CL-eval with CL TE.

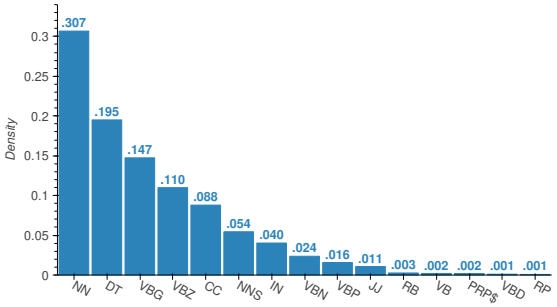


(c) Distribution of references trigrams on CL-dev.

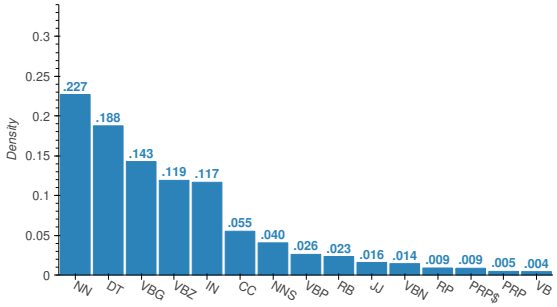


(d) Distribution of references trigrams on CL-eval.

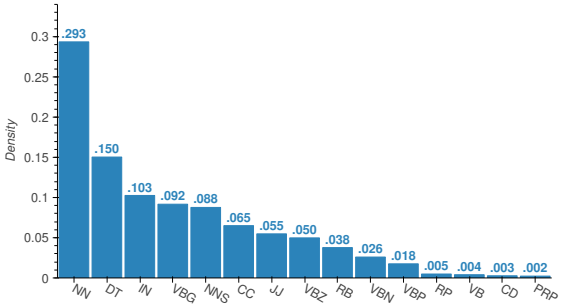
Fig. 6. Distributions of trigrams on CL-eval and CL-dev.



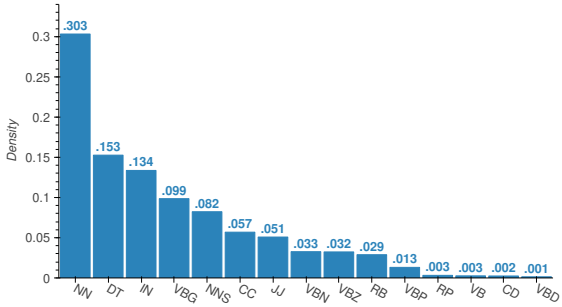
(a) Distribution of candidates POS-TAGs on AC-test with AC TE.



(b) Distribution of candidates POS-TAGs on AC-test with CL TE.

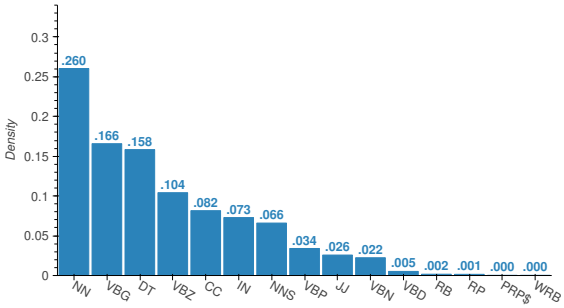


(c) Distribution of references POS-TAGs on AC-train.

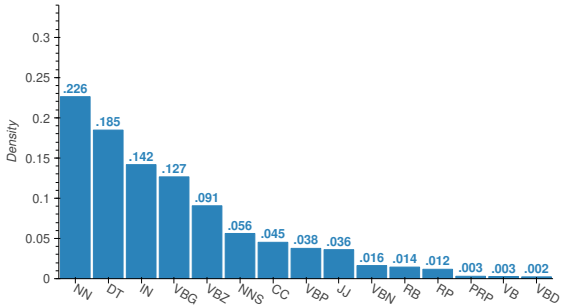


(d) Distribution of references POS-TAGs on AC-test.

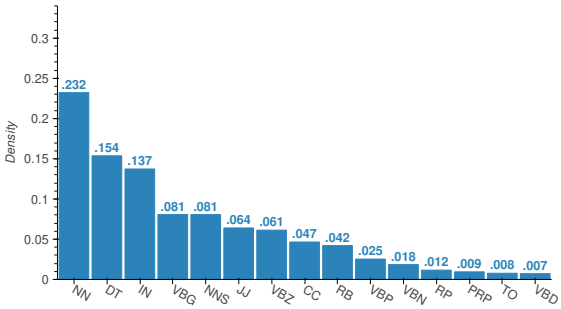
Fig. 7. Distributions of POS-TAGs on AC-test and AC-train.



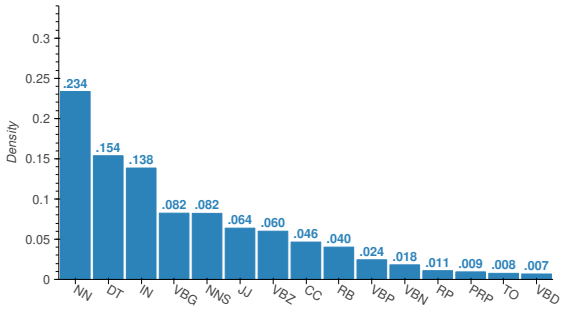
(a) Distribution of candidates POS-TAGs on CL-eval with AC TE.



(b) Distribution of candidates POS-TAGs on CL-eval with CL TE.



(c) Distribution of references POS-TAGs on CL-dev.



(d) Distribution of references POS-TAGs on CL-eval.

Fig. 8. Distributions of POS-TAGs on CL-eval and CL-dev.