IRIT-UPS DCASE 2021 AUDIO CAPTIONING SYSTEM

Technical Report

Etienne Labbé, Thomas Pellegrini

IRIT (UMR 5505), Université Paul Sabatier, CNRS, Toulouse, France
{etienne.labbe, thomas.pellegrini}@irit.fr

ABSTRACT

This document provides a description of our seq-to-seq models used for audio captioning in the task 6 of the DCASE 2021 challenge. Four submissions were made with two different models, a "Listen-Attend-Tell" and a "CNN-Tell", and two different algorithms for inference, greedy and beam search.

Index Terms— Automated audio captioning, convolutional encoder, recurrent decoder

1. INTRODUCTION

This year, the IRIT-UPS approach to the DCASE 2021 audio captioning Task 6 uses a sequence-to-sequence model Listen-Attend-Spell [1] architecture, renamed "Listen-Attend-Tell" (LAT), and a convolutional recurrent model that we refer to as "CNN-Tell". Our implementation uses PyTorch [2] and PyTorch-Lightning [3] and will be available after the end of the challenge on Github [1].

2. SYSTEMS DESCRIPTIONS

2.1. Data processing

Clotho [4] is the dataset provided by the organizers of the challenge for this year. We used Clotho v2.1, which contains 6974 audios clips of 15 to 30 seconds with 5 captions describing each audio file. We resample all the audio files from 44.1KHz to 32KHz, and we use log-Mel spectrograms as input to the models. The parameters of the spectrogram depends on the model used. We removed all punctuation in the captions. The vocabulary is built based on the Clotho development subset, and contains 4368 words in all our experiments.

2.2. Models

We used the same Listen-Attend-Tell (LAT) [1] as our last year participation [5]. The encoder is a pyramidal bidirectional LSTM and the decoder an attention-based LSTM network. The model contains 2.9 million parameters. All the details about the model architecture can be found in our last year technical report.

We introduced a new architecture, this year, called CNN-Tell. This model is comprised of an encoder-decoder architecture, with a convolutional model as encoder to produce audios features, and the same decoder as the LAT one. The encoder is taken from the Pretrained Audio Neural Networks study [6], more specifically the "Cnn14_DecisionLevelAtt" model, originally used for frame-wise sound event detection. The global architecture is described in Table 1 and the ConvBlock component in Table 2. This CNN is made up of 14 layers, hence the name.

We used the weights of this CNN14 pre-trained on AudioSet, available on Zenodo [2]. We kept the same parameters and layers, except that we removed the original classifier layer used for AudioSet. All the weights of the encoder are frozen and only the decoder is trained. We tried to also fine-tune the CNN14 encoder, but no gain was obtained doing so.

Layer	Architecture
input	Log-Mel spectrogram
bn0	BatchNorm2d
conv_block1	ConvBlock(64)
conv_block2	ConvBlock(128)
conv_block3	ConvBlock(256)
conv_block4	ConvBlock(512)
conv_block5	ConvBlock(1024)
conv_block6	ConvBlock(2048)
fc1	Linear(2048)
output	Audios embeddings

Table 1: The encoder architecture used as encoder of CNN-Tell.

Layer	Architecture
conv1	Conv2d(N)
bn1	BatchNorm2d
conv2	Conv2d(N)
bn2	BatchNorm2d
avgpool	AvgPool2d

Table 2: The ConvBlock architecture with N filters used in CNN-Tell.

The CNN14 audio output embeddings have a shape (batch_size, embedding_size, nb_frames). These embeddings are used as input of two projection layers in parallel to compute the key and value inputs used in the LSTM decoder. The final model contains 84.5 million parameters, but only 1.6 million trainable parameters (the decoder part).

In both models, the optimization criterion used is standard cross-entropy. The validation loss was used for monitoring the best model kept for evaluation.

[1] https://github.com/Labbeti/dcase2021task6

[2] https://zenodo.org/record/3987831

Submission	Model	Beam	BLEU1	BLEU2	BLEU3	BLEU4	METEOR	ROUGEL	CIDEr	SPICE	SPIDEr
-	Baseline	-	0.378	0.119	0.050	0.017	0.078	0.263	0.075	0.028	0.051
1	LAT	-	0.435	0.229	0.129	0.069	0.252	0.195	0.136	0.067	0.101
2	LAT	10	0.452	0.262	0.168	0.102	0.249	0.193	0.172	0.071	0.122
3	CNN-Tell	-	0.523	0.316	0.191	0.109	0.309	0.231	0.287	0.104	0.195
4	CNN-Tell	10	0.541	0.358	0.243	0.159	0.327	0.235	0.351	0.110	**0.231**

Table 3: Results on the evaluation subset of Clotho v2.1. Higher score is better.

2.3. Hyper-parameters

In all our experiments, we used the Adam optimizer [7] with a learning rate set to $5 \cdot 10^{-4}$, weight decay set to 10^{-6}, β_1 set to 0.9, β_2 set to 0.999 and ϵ set to 10^{-8}. We trained ours models for 50 epochs and the batch size is set to 8.

We also used a cosine-scheduler which decreases the learning rate at each epoch using the following equation:

$$lr_k = lr_0 \frac{1}{2}\left(1 + \cos\left(\frac{k\pi}{K}\right)\right) \quad (1)$$

with k being the current epoch index, and K the total number of epochs.

The log-Mel spectrograms have 64 Mel bands and a window size of 1024 samples. The hop length is equal to 512 for the LAT model and 320 for CNN-Tell. The spectrogram power is set to 1 for LAT model and to 2 for CNN-Tell.

2.4. Beam search

For inference, we used the beam search algorithm to produce results better than with greedy search. We compute a log-score described in Eq. 2 based on the log-probabilities. We tried sentence length normalization, but it did not improve performance in our experiments.

$$\log p(w_1, ..., w_L|x) = \sum_{i=1}^{L} \log p(w_i|x, w_1, ..., w_{i-1}) \quad (2)$$

3. RESULTS

Table 3 shows the results of the baseline and the four submissions, with the nine metrics on the Clotho v2.1 evaluation subset. The best SPIDEr score is 0.231 and is achieved by the CNN-Tell model with a beam size of 10. Larger beam sizes did not bring improvements.

4. ACKNOWLEDGMENT

We used the OSIRIM platform, administered by IRIT and supported by CNRS, the Region Midi-Pyrénées, the French Government and ERDF (http://osirim.irit.fr/site/en). We also used the HPC resources from CALMIP (Grant 2020-p20022).

5. REFERENCES

[1] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, Attend and Spell," *arXiv:1508.01211 [cs, stat]*, Aug. 2015, arXiv: 1508.01211. [Online]. Available: http://arxiv.org/abs/1508.01211

[2] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *proc. NeurIPS*, 2019, pp. 8026–8037.

[3] W. Falcon and .al, "Pytorch lightning," *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, vol. 3, 2019.

[4] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: An Audio Captioning Dataset," *arXiv:1910.09387 [cs, eess]*, Oct. 2019, arXiv: 1910.09387. [Online]. Available: http://arxiv.org/abs/1910.09387

[5] T. Pellegrini, "IRIT-UPS DCASE 2020 audio captioning system," DCASE2020 Challenge, Tech. Rep., June 2020.

[6] 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," *arXiv:1912.10211 [cs, eess]*, Aug. 2020, arXiv: 1912.10211. [Online]. Available: http://arxiv.org/abs/1912.10211

[7] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Jan. 2017, arXiv: 1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980