TEMPORAL SUB-SAMPLING OF AUDIO FEATURE SEQUENCES FOR AUTOMATED AUDIO CAPTIONING

Khoa Nguyen[1]*, Konstantinos Drossos*[2]*, Tuomas Virtanen*[2]

[1] 3D Media Group, Tampere University, Tampere, Finland
{firstname.lastname}@tuni.fi
[2] Audio Research Group, Tampere University, Tampere, Finland
{firstname.lastname}@tuni.fi

ABSTRACT

Audio captioning is the task of automatically creating a textual description for the contents of a general audio signal. Typical audio captioning methods rely on deep neural networks (DNNs), where the target of the DNN is to map the input audio sequence to an output sequence of words, i.e. the caption. Though, the length of the textual description is considerably less than the length of the audio signal, for example 10 words versus some thousands of audio feature vectors. This clearly indicates that an output word corresponds to multiple input feature vectors. In this work we present an approach that focuses on explicitly taking advantage of this difference of lengths between sequences, by applying a temporal sub-sampling to the audio input sequence. We employ a sequence-to-sequence method, which uses a fixed-length vector as an output from the encoder, and we apply temporal sub-sampling between the RNNs of the encoder. We evaluate the benefit of our approach by employing the freely available dataset Clotho and we evaluate the impact of different factors of temporal sub-sampling. Our results show an improvement to all considered metrics.

Index Terms— audio captioning, recurrent neural networks, temporal sub-sampling, hierarchical sub-sampling networks

1. INTRODUCTION

Audio captioning is the task of automatically describing the contents of a general audio signal, using natural language [1, 2]. It can be considered an inter-modal translation task, where the contents of the audio signal are translated to text [2, 3]. Audio captioning offers the ability for developing methods that can learn complex information from audio data, like spatiotemporal relationships, differentiation between foreground and background, and higher and abstract knowledge (e.g. counting) [2, 4].

Audio captioning started in 2017 [1] and all published audio captioning methods (up to now) are based on deep neural networks (DNNs) [3, 5]. The usual set-up of methods is according to sequence-to-sequence architectures, where an encoder (usually based on recurrent neural networks, RNNs) takes a sequence of audio feature vectors as an input, and a decoder (also usually RNN-based) takes as an input the output of the encoder and outputs a sequence of words. A common element to sequence-to-sequence architectures is the alignment of the input and output sequences [6, 7, 8]. Three main alternative techniques have been proposed for the alignment of the input and output sequences, namely the employment of a fixed-length vector representation of the output of the encoder [6], the usage of attention mechanism [7, 8], and self-attention [9]. The former two approaches have been widely adopted in the audio captioning field. Specifically, [1] presents a method that employs an RNN encoder, an RNN decoder, and an attention mechanism between the encoder and encoder, to align the input and output sequences. Study [3] used again an RNN encoder and an RNN decoder, but the alignment of the input and output sequences performed with the usage of a fixed-length vector. This vector was the mean, over time, of the output of the encoder. Finally, [5] presented an approach for audio captioning, employing the attention mechanism presented in [8].

Although the above-mentioned techniques seem to be essential for the audio captioning task, they are applied only at the output of the encoder. This means that the encoder processes the whole input audio sequence, and only at its output there is the association (through the alignment mechanisms) of the different parts of the input sequence with the different parts of the output sequence. If we could adopt a policy for effectively collate sequential parts of the input sequence, then we might be able to let the encoder learn better, entities that exhibit long time presence, i.e., long temporal patterns that correspond to one output class like the sound of a car and the word "car". This need for DNNs has been identified at least since 2012 [10] and the hierarchical sub-sampling networks were introduced, and also adopted more recently, e.g. [11].

In this paper we employ the hierarchical sub-sampling networks and we present a novel approach for audio captioning methods employing multi-layered and RNN-based encoders. We draw inspiration from the empirical observation that each word in the caption corresponds to multiple time-steps of the input sequence to a DNN-based audio captioning method, and we hypothesize that the performance of the method could be enhanced by reducing the temporal length of the sequence after each RNN layer in an RNN-based encoder. To assess our hypothesis and our method, we employ a method that is not using temporal sub-sampling, and we alter it by solely employing the sub-sampling. The obtained results show that, indeed, temporal sub-sampling can enhance the performance of the audio captioning methods.

The rest of the paper is organized as follows. In Section 2 we present our method and the temporal sub-sampling method. In Section 3 we present the followed evaluation procedure, and the obtained results are in Section 4. Section 5 concludes the paper.

K. Drossos and T. Virtanen would like to acknowledge CSC-IT Center for Science, Finland, for computational resources. Part of the computations leading to these results were performed on a TITAN-X GPU donated by NVIDIA to K. Drossos.

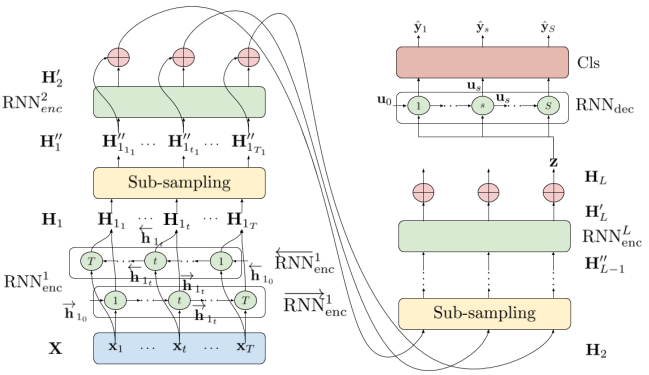

Figure 1: Illustration of the proposed method, with L bi-directional RNN layers at the encoder. Details are shown for $\text{RNN}_{\text{enc}}^{l=1}$ and for the result l are similar. Details for sub-sampling are in Figure 2.

2. PROPOSED METHOD

Our proposed method employs a multi-layered, bi-directional RNN-based encoder, an RNN-based decoder, accepts as an input a sequence of T audio feature vectors with F features, $\mathbf{X} \in \mathbb{R}^{T \times F}$, and outputs a sequence of S vectors $\hat{\mathbf{Y}} = [\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_S]$, where each vector $\hat{\mathbf{y}} \in [0,1]^D$ contains the predicted probability for each of the D words available to the method, where D is the amount of unique words in the captions of the training dataset. After each bi-directional RNN layer in the encoder (apart from the last one), our method applies a temporal sub-sampling of the output sequence of the bi-directional RNN layer, before use the sequence as an input for the next bi-directional RNN layer in the encoder. Figure 1 illustrates the method. Our method is otherwise the same as the baseline method of the DCASE 2020 automated audio captioning task (task 6)[1], enhanced with the of temporal sub-sampling of the latent representations in the encoder.

2.1. Encoder with temporal sub-sampling

Our encoder consists of L_{enc} bi-directional RNNs, where $\overrightarrow{\text{RNN}}_{\text{enc}}^{l}$ and $\overleftarrow{\text{RNN}}_{\text{enc}}^{l}$ are the l-th forward and backward RNNs of the encoder, respectively. $\overrightarrow{\text{RNN}}_{\text{enc}}^{1}$ and $\overleftarrow{\text{RNN}}_{\text{enc}}^{1}$ process the input sequence $\mathbf{X}$ as

$$\overrightarrow{\mathbf{h}}_{1_t} = \overrightarrow{\text{RNN}}_{\text{enc}}^{1}(\mathbf{x}_t, \overrightarrow{\mathbf{h}}_{1_{t-1}}) \text{ and} \quad (1)$$

$$\overleftarrow{\mathbf{h}}_{1_t} = \overleftarrow{\text{RNN}}_{\text{enc}}^{1}(\overleftarrow{\mathbf{x}}_t, \overleftarrow{\mathbf{h}}_{1_{t-1}}), \quad (2)$$

where $\overleftarrow{\mathbf{x}}_t$ is the t-th feature vector of the time-reversed $\mathbf{X}$, $\overrightarrow{\mathbf{h}}_{1_t}, \overleftarrow{\mathbf{h}}_{1_t} \in [-1,1]^{\Xi}$ are the outputs of the $\overrightarrow{\text{RNN}}_{\text{enc}}^{1}$ and $\overleftarrow{\text{RNN}}_{\text{enc}}^{1}$, respectively, for the t-th time-step, $\overrightarrow{\mathbf{h}}_{1_0} = \overleftarrow{\mathbf{h}}_{1_0} = [0]^{\Xi}$, and Ξ is the amount of output features of each of the RNNs of the l-th bi-directional RNN of the encoder. Then, the outputs of the $\overrightarrow{\text{RNN}}_{\text{enc}}^{1}$ and $\overleftarrow{\text{RNN}}_{\text{enc}}^{1}$, $\overrightarrow{\mathbf{h}}_{1_t}$ and $\overleftarrow{\mathbf{h}}_{1_t}$, respectively, are concatenated as

$$\mathbf{h}_{1_t} = [\overrightarrow{\mathbf{h}}_{1_t}^{\top}, \overleftarrow{\mathbf{h}}_{1_t}^{\top}]^{\top}, \quad (3)$$

[1] http://dcase.community/challenge2020/task-automatic-audio-captioning

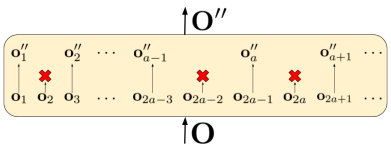

Figure 2: Illustration of the sub-sampling process with a factor $M = 2$, with an input of a sequence with A vectors of B features, $\mathbf{O} \in \mathbb{R}^{A \times B}$, and an output of another sequence $\mathbf{O}'' \in \mathbb{R}^{\lfloor A/M \rfloor \times B}$. With the red "X" we indicate the vectors that were discarded, according to the sub-sampling process.

and $\mathbf{H}_1 = [\mathbf{h}_{1_1}, \ldots, \mathbf{h}_{1_T}]$. Then, for $2 \leq l < L$, our method applies a temporal sub-sampling to $\mathbf{h}_{l-1}$, as

$$\mathbf{H}''_{l-1} = \{\mathbf{h}_{iM+1}^{l-1}\}_{i=0}^{i=\lfloor (T_l)/M \rfloor}, \quad (4)$$

where T_l is the amount of time-steps of $\mathbf{H}''_{l-1}$, $M \in \mathbb{N}^{\star}$ is the sub-sampling factor, and $\lfloor \cdot \rfloor$ is the floor function. Figure 2illustrates the sub-sampling process.

Then, $\mathbf{H}''_{l-1}$ is given as an input to $\overrightarrow{\text{RNN}}_{\text{enc}}^{l}$ and $\overleftarrow{\text{RNN}}_{\text{enc}}^{l}$, similarly to Eqs. (1), (2), and (3), obtaining $\mathbf{H}'_l$. $\mathbf{H}_l$ is obtained by a residual connection between $\mathbf{H}''_{l-1}$ and $\mathbf{H}'_l$, as

$$\mathbf{H}_l = \mathbf{H}'_l + \mathbf{H}''_{l-1}, \quad (5)$$

where $\mathbf{H}_l \in \mathbb{R}^{T_l \times \Delta}$ is the output of the l-th bi-directional RNN layer of the encoder. By utilizing Eq. (4), we enforce the RNNs of the encoder to squeeze the information in the input sequence to a smaller output sequence, effectively making the RNNs to learn a time-filtering and time-compression of the information in the input sequence [10, 11]. This is can be proven beneficial in the case of audio captioning, since one output class (i.e. one word) corresponds to multiple input time-steps. By temporal sub-sampling, we are enforcing the encoder to express the learnt information with lower temporal resolution, effectively providing a shorter sequence but with each of its time-step to represent longer temporal patterns [10]. The above presented scheme of temporal sub-sampling, results in reducing the length of the input audio sequence to M^{-L-1} times. For example, a sub-sampling factor of $M = 2$ and $L = 3$ RNN layers results in reducing the length of the input audio sequence 2^2 times (a reduction of 75.0%).

Finally, the output of the encoder. $\mathbf{z} \in \mathbb{R}^{\Delta}$, is formed as

$$\mathbf{z} = \mathbf{H}_{L_{T_L}}. \quad (6)$$

That is, $\mathbf{z}$ is the last time-step of the output sequence $\mathbf{H}_L$ of the L-th bi-directional RNN of the encoder.

2.2. Decoder and optimization

The decoder of our method consists of an RNN and a linear layer followed by a softmax non-linearity (the latter two will collectively referred to as classifier). The decoder takes as an input the $\mathbf{z}$ for every time step as

$$\mathbf{u}_s = \text{RNN}_{\text{dec}}(\mathbf{z}, \mathbf{u}_{s-1}), \quad (7)$$

where $\mathbf{u}_s \in [0,1]^{\Psi}$ is the output of the RNN_{dec} for the s-th time-step of the decoder, with $\mathbf{u}_s = [0]^{\Psi}$. $\mathbf{u}_s$ is used as an input to the classifier, Cls, as

$$\hat{\mathbf{y}}_s = \text{Cls}(\mathbf{u}_s). \quad (8)$$

The process described by Eqs (7) and (8) is repeated until $s = S$ or $\mathbf{y}_s$ matches to a predefined symbol, depending if the decoder is in optimization or inference process, respectively. The encoder, the decoder, and the classifier are jointly optimized to minimize the binary cross-entropy at each time-step and between the predicted, $\hat{\mathbf{y}}_s$, and ground truth, $\mathbf{y}_s = [y_{s,1}, \ldots, y_{s,D}]$, one-hot encoding of words.

3. EVALUATION

To evaluate our method, we employ a freely and well-curated audio captioning dataset, called Clotho [4], and the baseline of the DCASE 2020 audio captioning task. For assessing the performance of our method, we use machine translation and captioning metrics, employed by most of the existing audio captioning work.

3.1. Dataset and data pre-processing

Clotho contains audio clips of CD quality (44.1 kHz sampling rate, 16-bit sample width), and 5 captions for each audio clip. The time duration of the audio clips ranges from 15 to 30 seconds, and the amount of words in each caption ranges from eight to 20 words. Clotho provides three splits for developing audio captioning methods, namely development, evaluation, and testing. The development and evaluation splits are freely available online[2], while the testing split is withheld for scientific challenges. In this work, we employ the development and evaluation splits of Clotho, having 2893 and 1045 audio clips, yielding 14465 and 5225 captions, respectively. We choose Clotho because it is built to offer audio content diversity, and extra care has been taken for eliminating spelling errors, named entities, and speech transcription in the captions. Additionally, Clotho is already employed at the DCASE 2020 audio captioning task[1].

We extract $F = 64$ log-scaled mel-band energies from each of the 4981 audio clips of Clotho, using an 1024-sample long window (approximately 23 ms) with 50% overlap, and the Hamming windowing function. This results in having sequences from $T = 1292$ to $T = 2584$ audio feature vectors, for audio clips of 15 and 30 seconds duration, respectively. We process the captions of Clotho, starting by appending $\langle \text{eos} \rangle$ to all captions, where $\langle \text{eos} \rangle$ is a special token that signifies the end of the sequence (i.e. the caption). Then, we identify the set of words in the captions, include $\langle \text{eos} \rangle$ in that set as well, and represent each element of that set (called a token from now on) with an one-hot encoding vector $\mathbf{y} = \{0, 1\}^D$. This process yields a sequence of $S = 8$ to $S = 21$ one-hot encoded tokens for each caption. To implement the above, we employed the freely available code from the DCASE 2020 audio captioning task[3]. We use each audio clip with all of its corresponding captions as separate input-output examples, resulting to a total of 14465 and 5225 input-output examples for the development and evaluation splits, respectively. We use each of the sequences of audio feature vectors in the splits as our $\mathbf{X}$ and each of the corresponding sequences of one-hot encoded tokens as our $\mathbf{Y}$.

3.2. Hyper-parameters and training procedure

To optimize our method and fine tune its hyper-parameters, we employ the development split. By empirical observation on the values of the loss for the development split, we decided to round the loss value to three decimal digits and stop the training if the loss did not improve for 100 consecutive epochs. For the training of our method, we employed a batch size of 16 (mainly due to computational resources constraints). To apply a uniform T and S in a batch, we calculate the maximum T and S in the batch and we prepend vectors of $\{0\}^F$ to each $\mathbf{X}$ and append the one-hot encoded vector of $\langle \text{eos} \rangle$ to every $\mathbf{Y}$, in order to make them have the T and S equal to the maximum T and S in the same batch. Additionally, we observed that there is a considerable imbalance at the frequency of appearance of the tokens at the captions. That is, some w_i, for example "a"/"an" and "the", appear quite frequently (e.g. over 4000 times) at the captions, but some appear quite fewer times, e.g. five. To overcome this imbalance, each token, w_i, is inversely weighted by its frequency in the dataset, resulting in the following loss formulation

$$\mathcal{L}'(\hat{\mathbf{y}}_s, \mathbf{y}_s) = \Phi_s \mathcal{L}(\hat{\mathbf{y}}_s, \mathbf{y}_s), \tag{9}$$

where Φ_s is a weight for the loss calculation of the token represented by $\mathbf{y}_s$. But due to the quite large frequency of tokens like "a", we observed that Φ_s could get values as low as $1e-5$, which when compared to a value of $\Phi_s = 1$ for the non-frequent tokens, has a great difference. This difference at the values of Φ_s results in hampering significantly the learning of the frequent tokens and, in addition, will not ever contribute to the training of our method since we round $\mathcal{L}'$ to three decimal digits. Thus, we employed a clamping of Φ_s as

$$\Phi_s = \begin{cases} \frac{\min(f_w)}{f_{w_s}} & \text{if } \frac{\min(f_w)}{f_{w_s}} \geq \beta, \\ \beta & \text{otherwise} \end{cases}, \tag{10}$$

where β is a hyper-parameter that we set to $5e-1$ by following the above described process, f_{w_s} is the frequency of the w_s token in the development split, w_s is the token that the $\mathbf{y}_s$ corresponds to, and $\min(f_w)$ is the minimum frequency of all tokens in the Clotho dataset. We follow the baseline method of the DCASE 2020 audio captioning task, and we use $L = 3$, with $\Xi = 256$ and $\Psi = 256$, which are the same as in the baseline of DCASE 2020 audio captioning task. According to Clotho, $D = 4366$. We optimize the parameters of our method using $\mathcal{L}'$ and the Adam optimizer [12], with a learning rate of $1e-4$ and the values for β_1 and β_2 that are reported to the corresponding paper [12], and we employ dropout with a probability of $p = 0.25$ between $\text{RNN}^1_{\text{enc}}$ and $\text{RNN}^2_{\text{enc}}$, and between $\text{RNN}^2_{\text{enc}}$ and $\text{RNN}^3_{\text{enc}}$.

We choose hyper-parameters that yielded the lowest loss value for the development split, following the above mentioned policy of stopping the training process and implementing a random search over the combinations of Φ_s and learning rate of Adam. To evaluate the impact of the sub-sampling, we employ four different values for M, namely 2, 4, 8, and 16. Finally, the total amount of parameters of our method is 4 573 711, and the code of our method is based on the PyTorch framework and is freely available online[4].

3.3. Evaluation and metrics

We assess the performance of our method by using the above mentioned processes and hyper-parameters, and employing the metrics used in the DCASE 2020 audio captioning task. Each metric is calculated using the the predicted sequence of words $\hat{\mathbf{Y}}$ for a $\mathbf{X}$, and all the ground truth sequences $\mathbf{Y}$ for the same $\mathbf{X}$. We compare the obtained values against the ones reported by the baseline method

[2] https://zenodo.org/record/3490684

[3] https://github.com/audio-captioning/clotho-dataset

[4] https://github.com/DK-Nguyen/audio-captioning-sub-sampling

Table 1: Results for the baseline method, i.e. $M = 1$, and our proposed method with sub-sampling factor $M = \{2, 4, 8, 16\}$.

Metric	$M = 1$	$M = 2$	$M = 4$	$M = 8$	$M = 16$
BLEU_1	0.389	**0.426**	0.418	0.417	**0.426**
BLEU_2	0.136	0.151	0.151	**0.154**	0.147
BLEU_3	0.055	0.058	0.061	**0.063**	0.058
BLEU_4	0.015	0.020	0.018	**0.025**	0.022
ROUGE_L	**0.262**	0.274	0.275	0.274	0.274
METEOR	0.084	**0.092**	0.091	0.089	0.090
CIDEr	0.074	0.092	0.090	**0.093**	**0.093**
SPICE	0.033	**0.040**	0.037	**0.040**	0.036
SPIDEr	0.054	0.066	0.064	**0.067**	0.064

of the DCASE 2020 audio captioning task, which is our method with $M = 1$. The calculation of the metrics is performed using the available tools for DCASE 2020 audio captioning task[5].

Specifically, we use the machine translation metrics BLEU_n, ROUGE_L, and METEOR, and the captioning metrics CIDEr, SPICE, and SPIDEr. BLEU_n is a precision-based metrics that measures a weighted geometric mean of modified precision of n-grams between predicted and ground truth captions [13]. ROUGE_L [14] calculates an F-measure using a longest common sub-sequence (LCS) between predicted and ground truth captions, and METEOR [15] measures a harmonic mean of the precision and recall for segments between predicted and ground truth captions, which is shown to have high correlation with quality human-level translation. CIDEr [16] calculates a weighted cosine similarity using term-frequency inverse-document-frequency (TF-IDF) weighting for n-grams, and SPICE [17] measures the ability of the predicted captions to recover from the ground truth captions, objects, attributes, and the relationship between them. Finally, SPIDEr is a weighted mean between CIDEr and SPICE, exploiting the advantages of both metrics [18]. We assess the performance of our method versus the DCASE 2020 audio captioning task method using the values of the above mentioned metrics, evaluated on the evaluation split of Clotho.

4. RESULTS AND DISCUSSION

In Table 1 are the values of the employed metrics for the evaluation split of Clotho. As can be seen from Table 1, using a sub-sampling factor $M \geq 2$ always improves the values of the metrics. This fact clearly indicates that our proposed method of sub-sampling benefits the performance of audio captioning methods. The maximum value of SPIDEr is obtained for $M = 8$, is 0.067, and can be mainly attributed to the better SPICE score than the value of $M = 8$ yields. Though, the values of the metrics for the different sub-sampling factors, do not exhibit some systematic behaviour. That is, increasing above 2, i.e. $M > 2$ does not result in increasing or decreasing the performance. This could be attributed to the fact that the employed method employs a fixed-length output from the encoder. Thus the increased impact from reducing more the length of the sequence, cannot be observed since the output of the encoder is always a fixed-length vector. This fact strongly indicates that the impact of the increased M might be more visible in an audio captioning method, that employs an alignment mechanism which uses the whole output sequence of the encoder, and not only a fixed-length vector (e.g. attention).

[5] https://github.com/audio-captioning/caption-evaluation-tools

Table 2: Reduction of the sequence length (in percentages) compared to the input audio, required time for predictions on Clotho evaluation split, and minimum and maximum resulting amount of time-steps (T_L^{min} and T_L^{max}, respectively), according to sub-sampling factor $M = \{2, 4, 8, 16\}$, and $L = 3$.

M	T_L^{min}	T_L^{max}	**Reduction in length**	**Time (sec)**
1	1292	2584	0.00%	58.81
2	323	646	75.00%	34.13
4	80	161	93.75%	25.67
8	20	40	98.43%	21.73
16	5	10	99.60%	20.42

In Table 2 is the resulting reduction in the time needed for obtaining all the predicted outputs using the Clotho evaluation split, and the resulting length of the output sequence of the encoder compared to the input sequence $\mathbf{X}$. As can be observed from Table 2, the increase in M has also a clear impact at the time needed for obtaining the predicted captions. This is to be expected, since with temporal sub-sampling, the output of the encoder is 99.6% shorter compared to the length of the input audio.

Finally, an example of the output of our method with $M = 8$ is "*a person is walking a through something and*", for the file of the Clotho evaluation split "clotho_file_01 A pug struggles to breathe 1_14_2008" and with ground truth captions like "a man walking who is blowing his nose hard and about to sneeze" and "a small dog with a flat face snoring and groaning". Another example is the predicted caption "*a group of of birds and birds a*" for the file "clotho_file_sparrows" and with ground truth captions like "a flock of birds comes together with a lot of chirping" and "birds sing in different tones while in a large group". As it can be seen, out method manages to identify the sources and the actions, but it lacks in the language modelling (LM). The latter fact is to be expected, since our method did not focused on the LM perspective, e.g. by using attention or explicit LM.

5. CONCLUSIONS AND FUTURE WORK

In this paper we presented an approach for audio captioning that utilizes temporal sub-sampling, given the empirical observation that a word in a general audio caption refers to a sequence of audio samples. Our approach is focusing on methods that use a multi-layered and RNN-based encoder, utilizing a temporal sub-sampling of the output sequence of each RNN layer of the encoder. We evaluated our approach using the freely available audio captioning dataset, Clotho, using multiple factors of sub-sampling. The obtained results clearly indicate that temporal sub-sampling can benefit the audio captioning methods. We observed an increase at all metrics and with all sub-sampling factors greater than 2, compared to the case of not using sub-sampling (i.e. $M = 1$). The maximum benefit was observed for $M = 8$, where an increase of 1.3 is observed for SPIDEr. From the variation of the values of the utilized metrics, according to the different sub-sampling factors, we hypothesize that the temporal sub-sampling might have a more pronounced effect when is employed in a method that uses an alignment mechanism like attention. Though, more research is needed for verifying or disproving that.

6. REFERENCES

[1] 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.

[2] S. Lipping, K. Drossos, and T. Virtanen, "Crowdsourcing a dataset of audio captions," in *Detection and Classification of Acoustic Scenes and Events (DCASE) 2019*, Oct. 2019.

[3] M. Wu, H. Dinkel, and K. Yu, "Audio caption: Listen and tell," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 830–834.

[4] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: An audio captioning dataset," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 736–740.

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

[6] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. [Online]. Available: https://www.aclweb.org/anthology/D14-1179

[7] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the International Conference on Learning Representation (ICLR)*, 2014.

[8] M. J. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," in *Proceedings of the International Conference on Learning Representation (ICLR)*, vol. abs/1611.01603, 2016.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6000–6010.

[10] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, ser. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2012. [Online]. Available: https://books.google.fi/books?id=wpb-CAAAQBAJ

[11] F. Scheidegger, L. Cavigelli, M. Schaffner, A. C. I. Malossi, C. Bekas, and L. Benini, "Impact of temporal subsampling on accuracy and performance in practical video classification," in *2017 25th European Signal Processing Conference (EUSIPCO)*, 2017, pp. 996–1000.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representation (ICLR)*, 2014.

[13] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002, pp. 311–318.

[14] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. [Online]. Available: https://www.aclweb.org/anthology/W04-1013

[15] A. Lavie and A. Agarwal, "Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments," in *Proceedings of the second workshop on statistical machine translation*, 2007, pp. 228–231.

[16] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "CIDEr: Consensus-based image description evaluation," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2015, pp. 4566–4575.

[17] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "Spice: Semantic propositional image caption evaluation," in *European Conference on Computer Vision*. Springer, 2016, pp. 382–398.

[18] S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy, "Improved image captioning via policy gradient optimization of spider," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 873–881.