CONTINUAL LEARNING FOR AUTOMATED AUDIO CAPTIONING USING THE LEARNING WITHOUT FORGETTING APPROACH

Jan Berg and Konstantinos Drossos

Audio Research Group, Tampere University, Finland
{firstname.lastname}@tuni.fi

ABSTRACT

Automated audio captioning (AAC) is the task of automatically creating textual descriptions (i.e. captions) for the contents of a general audio signal. Most AAC methods are using existing datasets to optimize and/or evaluate upon. Given the limited information held by the AAC datasets, it is very likely that AAC methods learn only the information contained in the utilized datasets. In this paper we present a first approach for continuously adapting an AAC method to new information, using a continual learning method. In our scenario, a pre-optimized AAC method is used for some unseen general audio signals and can update its parameters in order to adapt to the new information, given a new reference caption. We evaluate our method using a freely available, pre-optimized AAC method and two freely available AAC datasets. We compare our proposed method with three scenarios, two of training on one of the datasets and evaluating on the other and a third of training on one dataset and fine-tuning on the other. Obtained results show that our method achieves a good balance between distilling new knowledge and not forgetting the previous one.

***Index Terms*—** Automated audio captioning, continual learning, learning without forgetting, WaveTransformer, Clotho, AudioCaps

1. INTRODUCTION

Automated audio captioning (AAC) is the inter-modal translation task, where a method takes as an input a general audio signal and generates a textual description of the contents of the audio signal [1]. AAC methods learn to describe sound sources/events, spatiotemporal relationships of events, textures and sizes, and higher-level knowledge like counting [1, 2], but not speech transcription [3, 4]. In a typical AAC scenario, a deep learning method is optimized in a supervised or reinforcement learning scheme and using an AAC dataset [5, 6, 7, 8, 9]. Audio clips are given as an input to the AAC method and the method generates captions for its inputs. Then, the method is optimized by trying to reduce the difference between the predicted and the actual (i.e ground truth) captions. Given that the existing AAC datasets are limited, the above scheme creates some limitations. For example, since the available information from the audio clips in the different datasets are most likely not overlapping and the described information and expression variability differs given that different annotators have been used [3, 10], then an AAC method optimized with one dataset will have problems when evaluated with another AAC dataset. Even if some technique is used for adapting an AAC method to another dataset, e.g. like transfer learning, it would be required to have all the new data for the adaptation. This creates limitation of continuously adapting an AAC method to new information.

The above presented problem of continuously adapting is not new and has been attacked using continual learning, sometimes also called lifelong learning [11, 12], which is the process of continuously adapting a method to new data and/or tasks. The advantage of continual learning over other techniques, e.g. transfer learning, is that the latter usually introduces the phenomenon of catastrophic forgetting, where the method is adapted to the new information but forgets the initially learned one [11, 13, 14, 15]. Though, continual learning methods seem that tackle this phenomenon [14, 15]. There are different approaches for continual learning, e.g. like joint training [12], though our focus is on the cases where the new data are not required a priori, because it is often not possible to have all data beforehand due to storing reasons (e.g. cannot store all the data) or to degradation of data (e.g. data have been lost over time). Approaches that do not require having the data to do the adaptation, can be roughly divided into three categories [11], namely regularization methods like learning without forgetting (LwF) [15] and elastic weight consolidation (ECW) [14], dynamic architectures like dynamically expandable networks (DEN) [16], and replay models like gradient episodic memory (GEM) [17].

In this paper we consider the scenario where an AAC method continuously adapts to new and unseen data, using unseen ground truth captions. This scenario can resemble, for example, an online platform where new audio data and captions can be provided by human users and the AAC method continuously learn from the new data. Focusing on this, we present a first method for continual learning for AAC, adopting the LwF approach. Although there are published continual learning approaches for audio classification using different approaches [18, 19], we employ LwF due to its simplicity, reduced need for resources, and the facts that LwF is model agnostic and no modifications are needed for the employed AAC model. Although, previous research has shown that one of the weaknesses of LwF is that its effectiveness is dependent on the similarity of the tasks at hand [20, 11, 21], we deem that this is not applicable to our case since we use LwF for continuously adapting to new data on the same task.

For our work presented here, we employ a freely available and pre-optimized AAC method called WaveTransformer (WT) [22] and two freely available AAC datasets, namely Clotho [3] and AudioCaps [10]. Since WT method has achieved state-of-the-art results on Clotho, we use AudioCaps as the new data that the AAC method will adapt. Given that there are no other published continual learning approaches for AAC, in this paper we do not consider the case of the mismatched set of words in the two employed datasets. The rest of the paper is organized as follows. Section 2 presents our method and Section 3 presents the adopted evaluation process. Obtained results are in Section 4 and Section 5 concludes the paper.

2. METHOD

Our method is model agnostic, based on LwF and knowledge distillation [15, 23]. It employs a pre-optimized AAC model, a copy of the AAC model, an iterative process, a regularization-based loss, and a stream of new audio data with captions that are used for learning the new information. The stream of new audio data and captions is used to provide input to the original and copy AAC models. The output of both models is used against the provided captions from the stream, but only the parameters of the copy model is updated. At every update of the parameters, the copy model can be used as an output of our continual learning method. An illustration of our continual learning approach is in Figure 1.

In more detail, we start by having a pre-optimized AAC model $\mathrm{M}_{\text{base}}(\cdot;\theta_{\text{base}})$, having the pre-optimized parameters θ_{base}. M_{base} is pre-optimized using a dataset of K input-output examples $\mathbb{D}_{\text{ori}} = \{(\mathbf{X}', \mathbf{Y}')_k\}_{k=1}^{K}$, where $\mathbf{X}' \in \mathbb{R}^{T_{\text{a}} \times F}$ is a sequence of T_{a} audio feature vectors having F features and $\mathbf{Y}' \in \{0,1\}^{T_{\text{w}} \times W}$ a sequence of T_{w} one-hot encoded vectors of W elements, that indicate the most probable word for each t_{w}-th word index. M_{base} generates its output as

$$\hat{\mathbf{Y}}'_k = \mathrm{M}_{\text{base}}(\mathbf{X}'_k;\theta_{\text{base}}), \tag{1}$$

where $\hat{\mathbf{Y}}'_k$ is the predicted caption by M_{base} when having as input the $\mathbf{X}'_k$. The optimization of θ_{base} is performed by minimizing the loss

$$\mathcal{L}(\theta_{\text{base}}, \mathbb{D}_{\text{ori}}) = \sum_{k=1}^{K} \text{CE}(\mathbf{Y}'_k, \hat{\mathbf{Y}}'_k), \tag{2}$$

where CE is the cross-entropy loss between $\mathbf{Y}'_k$ and $\hat{\mathbf{Y}}'_k$.

Then, we create a copy of M_{base}, $\mathrm{M}_{\text{new}}(\cdot;\theta_{\text{new}})$, having same hyper-parameters as M_{base} and the parameters θ_{new}. Our target is to continuously update θ_{new} for new data, without making M_{new} to deteriorate its performance on $\mathbb{D}_{\text{ori}}$. The new data are coming from a stream of data, $\mathcal{S}$, which continually produces new and unseen data (i.e. data not in $\mathbb{D}_{\text{ori}}$). We sample data from $\mathcal{S}$ in batches of B examples, creating the input-output examples as

$$\mathbb{D}_{\text{new}} = \{(\mathbf{X}, \mathbf{Y})_b : (\mathbf{X}, \mathbf{Y}) \sim \mathcal{S} \wedge b = 1, \dots, B\}, \tag{3}$$

where $\mathbf{X} \in \mathbb{R}^{T_{\text{a}} \times F}$ is a sequence of audio features, similar to $\mathbf{X}'$, and $\mathbf{Y} \in \{0,1\}^{T_{\text{w}} \times W}$ is a sequence of one-hot encoded vectors similar to $\mathbf{Y}'$. Here has to be noted that the captions coming from $\mathcal{S}$ can (and most likely will) have different set of words with $\mathbf{Y}'$. Though, our approach is not considering the problem of the different set of words. For that reason, we consider from $\mathbf{Y}$ only the words that are common with $\mathbf{Y}'$.

We use the sampled data $\mathbb{D}_{\text{new}}$ as an input to both M_{base} and M_{new}, resulting to

$$\hat{\mathbf{Y}}_b^{\text{base}} = \mathrm{M}_{\text{base}}(\mathbf{X}_b;\theta_{\text{base}}), \text{ and} \tag{4}$$

$$\hat{\mathbf{Y}}_b^{\text{new}} = \mathrm{M}_{\text{new}}(\mathbf{X}_b;\theta_{\text{new}}), \tag{5}$$

where $\hat{\mathbf{Y}}_b^{\text{base}}$ and $\hat{\mathbf{Y}}_b^{\text{new}}$ are the predicted outputs of M_{base} and M_{new}, respectively, when having as an input $\mathbf{X}_b$.

Having $\hat{\mathbf{Y}}_b^{\text{base}}$ and $\hat{\mathbf{Y}}_b^{\text{new}}$, we define the loss

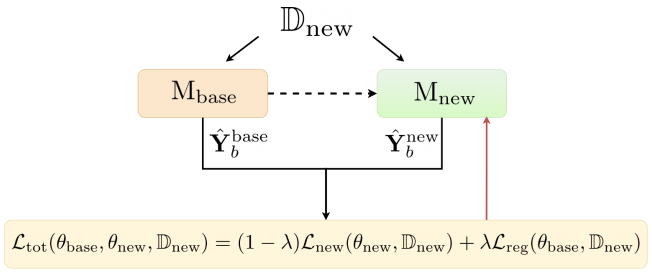
$\mathbb{D}_{\text{new}}$
M_{base}
M_{new}
$\hat{\mathbf{Y}}_b^{\text{base}}$
$\hat{\mathbf{Y}}_b^{\text{new}}$
$\mathcal{L}_{\text{tot}}(\theta_{\text{base}}, \theta_{\text{new}}, \mathbb{D}_{\text{new}}) = (1-\lambda)\mathcal{L}_{\text{new}}(\theta_{\text{new}}, \mathbb{D}_{\text{new}}) + \lambda\mathcal{L}_{\text{reg}}(\theta_{\text{base}}, \mathbb{D}_{\text{new}})$

Figure 1: Our proposed continual learning method for AAC. The dotted line represents the copying of the parameters of M_{base} to M_{new}, and it takes place only once at the beginning of the process. Red line indicates backpropagation for updating the parameters of M_{new}.

$$\mathcal{L}_{\text{tot}}(\theta_{\text{base}}, \theta_{\text{new}}, \mathbb{D}_{\text{new}}) = (1-\lambda)\mathcal{L}_{\text{new}}(\theta_{\text{new}}, \mathbb{D}_{\text{new}}) + \lambda\mathcal{L}_{\text{reg}}(\theta_{\text{base}}, \mathbb{D}_{\text{new}}), \text{ where} \tag{6}$$

$$\mathcal{L}_{\text{new}}(\theta_{\text{new}}, \mathbb{D}_{\text{new}}) = \sum_{b=1}^{B} \text{CE}(\mathbf{Y}_b, \hat{\mathbf{Y}}_b^{\text{new}}), \tag{7}$$

$$\mathcal{L}_{\text{reg}}(\theta_{\text{base}}, \theta_{\text{new}}, \mathbb{D}_{\text{new}}) = \sum_{b=1}^{B} \text{KL}(\hat{\mathbf{Y}}_b^{\text{base}}, \hat{\mathbf{Y}}_b^{\text{new}}), \text{ and} \tag{8}$$

λ is a factor that weights the contribution of $\mathcal{L}_{\text{new}}$ and $\mathcal{L}_{\text{reg}}$ to $\mathcal{L}_{\text{tot}}$, and $\text{KL}(a,b)$ is the KL-divergence between a and b. We use λ in order to balance the learning of the new information and the non-forgetting of the old information. The non-forgetting is implemented with the $\mathcal{L}_{\text{reg}}$, where the predictions of M_{new} are sought to be as similar to the predictions of M_{base}.

Finally, after calculating the $\mathcal{L}_{\text{tot}}$ for each sampling of data from $\mathcal{S}$, we obtain new optimized parameters for M_{new} as

$$\theta^{\star}_{\text{new}} = \underset{\theta_{\text{new}}}{\text{argmin}}\ \mathcal{L}_{\text{tot}}(\theta_{\text{base}}, \theta_{\text{new}}, \mathbb{D}_{\text{new}}), \tag{9}$$

where $\theta^{\star}_{\text{new}}$ are the new, optimized parameters. After obtaining $\theta^{\star}_{\text{new}}$, we update θ_{new} as

$$\theta_{\text{new}} = \theta^{\star}_{\text{new}}. \tag{10}$$

Thus, M_{new} is updated with the new information and also remembers old learned information, after applying (10). The iterative process of our continual method for AAC is the process described by Equations (3) to (10). The result of our method is the M_{new} after the application of Eq. (10).

3. EVALUATION

In order to evaluate our method, we use a freely available and pre-optimized method as our M_{base} and a freely available dataset different from $\mathbb{D}_{\text{ori}}$ to simulate $\mathcal{S}$, namely WaveTransformer (WT) and AudioCaps, respectively. $\mathbb{D}_{\text{ori}}$ used for WT is Clotho. We use minibatches of size B from AudioCaps to simulate $\mathbb{D}_{\text{new}}$, using only one epoch over AudioCaps. The performance of the continual learning is evaluated using metrics adopted usually in AAC task. Our code used for the implementation of our method can be found online[1].

[1]`https://github.com/JanBerg1/AAC-LwF`

3.1. Datasets and pre-processing

Clotho [3] is a freely available dataset for AAC, containing 3840 audio clips for training, 1046 for validation, and 1046 for evaluation. Each audio clip is of 15-30 seconds long and is annotated with five captions of eight to 20 words. This results to 19 200, 5230, and 5230 input-output examples for training, validating, and evaluating an AAC method, respectively. AudioCaps [10] is also a freely available AAC dataset, based on AudioSet [24]. AudioCaps has 38 118 audio clips for training, 500 for validation, and 979 for testing. All audio clips are 10 seconds long, and clips for training are annotated with one caption while clips for validation and testing with five captions. These result to 38 118, 2500, and 4895 input-output examples for training, validating, and evaluating, respectively. In all experiments, as $\mathbb{D}_{\text{ori}}$ we use the training split of the corresponding dataset and as $\mathbb{D}_{\text{new}}$ the training split from the other AAC dataset. During the stage of hyper-parameter tuning we used as the validation split from $\mathbb{D}_{\text{ori}}$ and $\mathbb{D}_{\text{new}}$ to evaluate the performance of our method, while during testing we used the evaluation split as $\mathbb{D}_{\text{new}}$, from the corresponding dataset. These result to $K = 19200$ for Clotho and $K = 38118$ for AudioCaps.

From all audio clips we extract $F = 64$ log mel-band energies, using a 46 second long Hamming window with 50% overlap. This results to $1292 \leq T_{\text{a}} \leq 2584$ for Clotho and $T_{\text{a}} = 862$ for AudioCaps. Additionally, for Clotho there are $8 \leq T_w \leq 20$ words in a caption and there are $W = 4367$ unique words, while for AudioCaps there are $2 \leq T_w \leq 51$ words in a caption and there are $W = 4506$ unique words. But, when M_{base} is optimized on either Clotho or AudioCaps the M_{new} is evaluated at the other dataset (i.e. M_{base} trained on Clotho and M_{new} evaluated on AudioCaps, and vice-versa). Since in our method we do not consider the case of learning new words, we keep only the common words from the dataset used for evaluation. For example, in the case of training on Clotho and evaluating on AudioCaps, we keep from AudioCaps only the words that exist in Clotho. The amount of words that we remove from AudioCaps is 1715.

3.2. M_{base} model

As M_{base} we use the WT AAC model, presented in [22]. WT consists of four learnable processes, three used for audio encoding and one for decoding the learned audio information to captions. WT takes as an input a sequence of audio features, e.g. $\mathbf{X}'$ or $\mathbf{X}$, and generates a sequence of words, e.g. $\mathbf{Y}'$ or $\mathbf{Y}$. Input audio features are processed in parallel by two different learnable processes, one for learning temporal patterns, $E_{\text{temp}}(\cdot)$, and one for learning time-frequency patterns, $E_{\text{tf}}(\cdot)$. E_{temp} consists of 1D convolutional neural networks (CNNs), set-up after the WaveNet model [25] and using gated and dilated convolutions. E_{tf} is based on 2D depth-wise separable CNNs, capable to learn time-frequency information and proven to give state-of-the-art results in sound event detection [26]. Both E_{temp} and E_{tf} do not alter the temporal resolution of their input and their output is concatenated and given as an input to a third learnable process, $E_{\text{merge}}(\cdot)$. E_{merge} learns to intelligently merge the information from E_{temp} and E_{tf}, producing as an output an encoded sequence of the input audio, containing both temporal and time-frequency information.

The output of E_{merge} is given as an input to a decoder, $D(\cdot)$ that is based on the Transformer model [27], using three stacked multi-head attention blocks. Each attention block takes as an input a sequence of tokens/words and uses two different multi-head attention processes. The first is a masked self-attention, for each token/word attending only to its previous ones in the input sequence. The second multi-head attention is a cross-modal attention, attending to the output of E_{merge} given the output of the first, self-attention process. The first multi-head attention block D takes as an input its outputs shifted right and applies a positional encoding. The output of the last multi-head attention block is given as an input to a classifier, which shares its weights through time and predicts the most probable word in each time-step of the output caption. WT is illustrated in Figure 2, after [22].

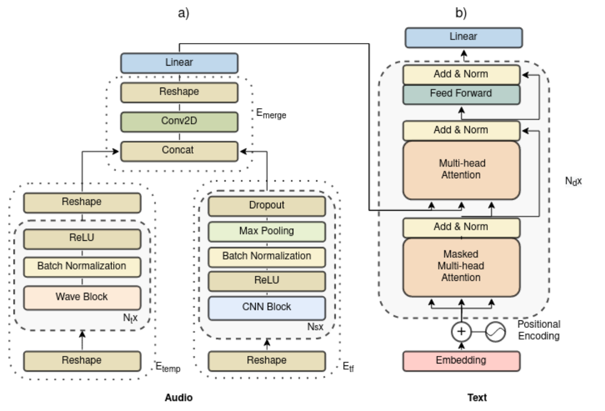

Figure 2: WT architecture, where a) is the encoder and b) the decoder, after [22].

3.3. Training, hyper-parameters, and evaluation

We compare the performance of our proposed method against the following baseline scenarios: i) WT pre-trained on Clotho and evaluated on Clotho and AudioCaps, ii) WT pre-trained on AudioCaps and evaluated on Clotho and AudioCaps, and iii) WT pre-trained on Clotho, fine-tuned on AudioCaps, and evaluated on Clotho and AudioCaps. We term the above cases as $\text{WT}_{\text{cl-au}}$, $\text{WT}_{\text{au-cl}}$, and $\text{WT}_{\text{cl-ft}}$, respectively. For pre-training M_{base}, we use the training split of the corresponding dataset, employing the early stopping policy by using the corresponding validation split and the associated SPIDEr score. For both datasets we use 10 consecutive epochs for early stopping, detecting not improving SPIDEr score. As an optimizer we use Adam [28] with the proposed values for the hyper-parameters. Additionally, we use a temperature hyper-parameter at the softmax non-linearity of the classifier of M_{new}, as this has been found to improve the performance [15]. We use the value of 2 for this hyper-parameter.

Using the above protocol, we evaluate the performance of our method using $\lambda = 0.70, 0.75, \ldots, 0.95, 1.0$ and $B = 4, 8, 12$. We use the pre-trained WT on Clotho, and we simulate $\mathcal{S}$ as mini-batches of size B from AudioCaps, as described by Eq. 3. We assess the performance of the M_{new} at the 50th, 75th, and 150th update, and after using only once all data from AudioCaps, using SPIDEr score [29]. SPIDEr [29] is the weighted average of CIDEr and SPICE metrics. CIDEr [30] employs weighted cosine similarity of n-grams, based on the term-frequency inverse-document-frequency (TFIDF), effectively quantifying the difference of the predicted and ground truth captions on using the same words to convey information. On the other hand, SPICE [31] analyzes the described scene and quantifies the differences of the predicted and ground truth caption in describing the same objects, attributes, and their relation-

Table 1: SPIDEr score of the baseline scenarios

Baseline scenario	**SPIDEr $\mathbb{D}_{\text{ori}}$**	**SPIDEr $\mathbb{D}_{\text{new}}$**
$\text{WT}_{\text{cl-au}}$	0.182	0.108
$\text{WT}_{\text{au-cl}}$	0.318	0.102
$\text{WT}_{\text{cl-ft}}$	0.065	0.247

ships.

4. RESULTS

In Table 1 are the results of M_{base}, regarding the three different baseline scenarios. In Table 2 are the obtained results of our method, for various values of B and λ, focusing on the SPIDEr score for $\mathbb{D}_{\text{ori}}$ and $\mathbb{D}_{\text{new}}$. As can be seen from Table 1 and from the cases of $\text{WT}_{\text{cl-au}}$ and $\text{WT}_{\text{au-cl}}$, the AAC method performs better on the $\mathbb{D}_{\text{ori}}$ than $\mathbb{D}_{\text{new}}$. This clearly shows that the model cannot perform equally well on the two different datasets, just by pre-training on one of them. Focusing on the $\text{WT}_{\text{cl-ft}}$, can be seen that the AAC method can perform good on the second dataset, i.e. $\mathbb{D}_{\text{new}}$, but the performance of the method on $\mathbb{D}_{\text{ori}}$ degrades considerably. This strengthens the need for our method, which aims at alleviating the degradation of performance on the $\mathbb{D}_{\text{ori}}$.

As can be seen from Table 2, it seems that the value of B has an observable impact on the performance on $\mathbb{D}_{\text{ori}}$. That is, lower values of B seem to not benefit the performance on $\mathbb{D}_{\text{ori}}$ for any value of λ. Specifically, for values of $B = 4$, the SPIDEr score on $\mathbb{D}_{\text{ori}}$ is lower than the SPIDEr score for $\mathbb{D}_{\text{ori}}$ and for $B > 4$, for any value of λ. The same stands mostly true for $B = 8$ and $B > 8$, with the exception where $\lambda = 0.7$. The above observation for B suggests that the batch size for sampling the stream of data $\mathcal{S}$ can also act as a regularizer for the not-forgetting of information from the $\mathbb{D}_{\text{ori}}$. Regarding the impact of λ, one can directly see the effect of the $1 - \lambda$ and λ factors in Eq. (6), having $1 - \lambda$ for scaling the effect of $\mathcal{L}_{\text{new}}$ and λ for scaling the effect of $\mathcal{L}_{\text{reg}}$. Specifically, for $\lambda = 1$ the SPIDEr score for $\mathbb{D}_{\text{new}}$ is lower than the SPIDEr score for $\mathbb{D}_{\text{ori}}$. This trend is in accordance with the observations from Table 1, and is an expected trend since the loss from $\mathbb{D}_{\text{new}}$ is turned to 0 for $\lambda = 1$. Given the observations for B from the same Table 2, it is indicated that using just the loss $\mathcal{L}_{\text{reg}}(\theta_{\text{base}}, \theta_{\text{new}}, \mathbb{D}_{\text{new}})$ for updating θ_{new} can enhance, up to an extent, the performance of the M_{new} on the new data from $\mathcal{S}$. Similarly, for values of $\lambda < 1.00$ the performance of M_{new} on $\mathbb{D}_{\text{new}}$ increases for all values of B. Additionally, the value of λ and the SPIDEr score on $\mathbb{D}_{\text{new}}$ have a reverse analogous relationship.

In terms of better performing combination of λ and B, we see two trends. There is the combination of $B = 4$ and $\lambda = 0.85$, which yields the best performance on $\mathbb{D}_{\text{new}}$ of SPIDEr$= 0.239$. Additionally, there is the combination of $B = 12$ and $\lambda = 0.80$, which seems to act as the best regularizer for the performance on $\mathbb{D}_{\text{ori}}$, with SPIDEr$= 0.186$. These results are in accordance with the previous observations for B and λ, indicating some kind of trade-off for the values of B and λ. Finally, comparing Tables 1 and 2, one can see the benefit of our method, giving a good balance between the top performance on $\mathbb{D}_{\text{new}}$ and not deteriorating the performance on $\mathbb{D}_{\text{ori}}$.

5. CONCLUSIONS

In the paper we presented a first study of continual learning for AAC. Our method is based on the learning without forgetting method, which focuses on continuously updating the knowledge of a pre-trained AAC method on new AAC data, without degrading the performance of the AAC method on the originally used dataset during pre-training. For that reason, we employed a freely available and pre-trained AAC method and two freely available AAC datasets. We use the adopted AAC method which is pre-trained on one of the employed AAC datasets, and we use the other AAC dataset as a continuous stream of AAC data. We update the knowledge of the employed AAC method given the stream of AAC data. We compare our method against three baselines, two for training on one of the AAC datasets and evaluating on the other, and a third of training on one of the AAC datasets and fine-tuning the trained method to the other. Our results show that our method manages to not let the performance of the AAC method to deteriorate on the original AAC dataset, while, in the same time, manages to distil information from the new data to the employed AAC method.

For future research, utilizing AAC datasets set in more distinct domains and training those in consecutive way to the model would provide more data on how effective these methods can be when used for AAC. Recent years continuous learning has been a hot issue and more methods have been introduced just during last few years, many of which might effective when utilized for AAC as well.

Table 2: Results of continual learning using Learning without Forgetting for AAC, for various B and λ. With bold are indicated the best SPIDEr scores for each dataset.

batch size B	**λ**	**SPIDEr $\mathbb{D}_{\text{ori}}$**	**SPIDEr $\mathbb{D}_{\text{new}}$**
4	0.70	0.098	**0.239**
	0.75	0.102	0.215
	0.80	0.093	0.214
	0.85	0.115	0.230
	0.90	0.133	0.215
	0.95	0.155	0.192
	1.00	0.163	0.119
8	0.70	0.113	0.210
	0.75	0.119	0.223
	0.80	0.132	0.220
	0.85	0.133	0.190
	0.90	0.156	0.187
	0.95	0.178	0.157
	1.00	0.165	0.114
12	0.70	0.109	0.211
	0.75	0.160	0.197
	0.80	**0.186**	0.157
	0.85	0.171	0.179
	0.90	0.182	0.153
	0.95	0.185	0.145
	1.00	0.176	0.115

6. ACKNOWLEDGMENT

The authors wish to acknowledge CSC-IT Center for Science, Finland, for computational resources. K. Drossos has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957337, project MARVEL.

7. 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)*, Oct. 2017, pp. 374–378.

[2] Y. Koizumi, R. Masumura, K. Nishida, M. Yasuda, and S. Saito, "A transformer-based audio captioning model with keyword estimation," in *INTERSPEECH 2020*, 2020.

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

[4] S. Lipping, K. Drossos, and T. Virtanen, "Crowdsourcing a dataset of audio captions," in *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2019)*, 2019.

[5] D. Takeuchi, Y. Koizumi, Y. Ohishi, N. Harada, and K. Kashino, "Effects of word-frequency based pre- and post-processings for audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 190–194.

[6] E. Çakır, K. Drossos, and T. Virtanen, "Multi-task regularization based on infrequent classes for audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 6–10.

[7] X. Xu, H. Dinkel, M. Wu, and K. Yu, "A crnn-gru based reinforcement learning approach to audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 225–229.

[8] K. Chen, Y. Wu, Z. Wang, X. Zhang, F. Nian, S. Li, and X. Shao, "Audio captioning based on transformer and pre-trained cnn," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 21–25.

[9] K. Nguyen, K. Drossos, and T. Virtanen, "Temporal sub-sampling of audio feature sequences for automated audio captioning," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, Tokyo, Japan, November 2020, pp. 110–114.

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

[11] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[12] Z. Chen, B. Liu, R. Brachman, P. Stone, and F. Rossi, *Lifelong Machine Learning*, 2nd ed. Morgan & Claypool Publishers, 2018.

[13] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[14] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," 2017.

[15] Z. Li and D. Hoiem, "Learning without forgetting," 2017.

[16] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," 2018.

[17] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," *Advances in neural information processing systems*, vol. 30, pp. 6467–6476, 2017.

[18] X. Tao, X. Hong, X. Chang, S. Dong, X. Wei, and Y. Gong, "Few-shot class-incremental learning," 2020.

[19] Y. Wang, N. J. Bryan, M. Cartwright, J. Pablo Bello, and J. Salamon, "Few-shot continual learning for audio classification," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 321–325.

[20] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert gate: Lifelong learning with a network of experts," 2017.

[21] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 1–1, 2021.

[22] A. Tran, K. Drossos, and T. Virtanen, "Wavetransformer: An architecture for audio captioning based on learning temporal and time-frequency information," in *29th European Signal Processing Conference (EUSIPCO)*, Aug. 2021.

[23] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[24] 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 *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[25] A. den Oord et al., "Wavenet: A generative model for raw audio," in *9th International Speech Communication Association (ISCA) Speech Synthesis Workshop*, 2016.

[26] K. Drossos, S. I. Mimilakis, S. Gharib, Y. Li, and T. Virtanen, "Sound event detection with depthwise separable and dilated convolutions," in *2020 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2020.

[27] A. Vaswani, L. Jones, N. Shazeer, N. Parmar, J. Uszkoreit, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *31st Conference on Neural Information Processing Systems (NeurIPS 2017)*, 2017.

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

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

[30] 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.

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