WaveTransformer: An Architecture for Audio Captioning Based on Learning Temporal and Time-Frequency Information

An Tran
Audio Research Group
Tampere University
Tampere, Finland
an.tran@tuni.fi

Konstantinos Drossos
Audio Research Group
Tampere University
Tampere, Finland
konstantinos.drossos@tuni.fi

Tuomas Virtanen
Audio Research Group
Tampere University
Tampere, Finland
tuomas.virtanen@tuni.fi

***Abstract*—Automated audio captioning (AAC) is a novel task, where a method takes as an input an audio sample and outputs a textual description (i.e. a caption) of its contents. Most AAC methods are adapted from image captioning or machine translation fields. In this work, we present a novel AAC method, explicitly focused on the exploitation of the temporal and time-frequency patterns in audio. We employ three learnable processes for audio encoding, two for extracting the temporal and time-frequency information, and one to merge the output of the previous two processes. To generate the caption, we employ the widely used Transformer decoder. We assess our method utilizing the freely available splits of the Clotho dataset. Our results increase previously reported highest SPIDEr to 17.3, from 16.2 (higher is better).**

***Index Terms*—automated audio captioning, wavetransformer, wavenet, transformer**

I. Introduction

Automated audio captioning (AAC) is an intermodal translation task, where the system receives as an input an audio signal and outputs a textual description of the contents of the audio signal (i.e. outputs a caption) [1]. AAC is not speech-to-text, as the caption does not transcribe speech. In a nutshell, an AAC method learns to identify the high-level, humanly recognized information in the input audio, and expresses this information with text. Such information can include complex spatiotemporal relationships of sources and entities, textures and sizes, and abstract and high-level concepts (e.g. "several barnyard animals mooing in a barn while it rains outside").

There are different published approaches for AAC. Regarding input audio encoding, some approaches use recurrent neural networks (RNNs) [2], [3], [4], others 2D convolutional neural networks (CNNs) [5], [6], and some others the Transformer model [7], [8]. Though, RNNs are known to have difficulties on learning temporal information [9], 2D CNNs model time-frequency but not temporal patterns [10], and the Transformer was not originally designed for sequences of thousands time-steps [7]. For generating the captions, the Transformer decoder [6], [11], [8] or RNNs [1], [3], [5] are mostly employed, and the alignment of input audio and output captions is typically implemented with an attention mechanism [12], [11]. Also, some approaches adopt a multi-task approach, where the AAC method is regularized by the prediction of keywords, based on the input audio [6], [11], [13].

In this paper we present a novel AAC approach, based on a learnable representation of audio that is focused on encoding the information needed for AAC. We adopt existing machine listening approaches where sound sources and actions are well captured by time-frequency information [10], [14], and additionally exploit temporal information in audio using 1D dilated convolutions that operate on the time dimension [15], [16], for learning of high-level information (e.g. background vs foreground, spatiotemporal relationships). Additionally, we claim that these two types of information can be combined, providing a well-performing learned audio representation for AAC. To this end, we present an approach which is explicitly focusing on the above aspects. We employ three different encoding processes for the input audio, one regarding temporal information, a second that considers the time-frequency information, and a third that merges the previous two and its output is given as an input to a decoder which generates the output caption.

The contribution of our work is: i) we present *the first method that explicitly focuses on exploiting temporal and local time-frequency information for AAC*, ii) *we provide highest reported results* using only the *freely available splits of Clotho* dataset and *without any data augmentation and/or multi-task learning*, and iii) we show the impact on the performance of the different components of our method, i.e. the temporal and local time-frequency information, merging the previous two, or all of them. The rest of the paper is as follows. In Section II we present our method. Section III presents the evaluation process of our method, and the obtained results are in Section IV. Section V concludes the paper and proposes future research directions.

II. Proposed method

Our method takes as an input a sequence of T_{a} vectors with F audio features, $\mathbf{X} \in \mathbb{R}^{T_{\text{a}} \times F}$, and outputs a sequence of T_{w}

vectors having W one-hot encoded words, $\mathbf{Y}$. To do so, our method utilizes an encoder-decoder scheme, where the encoder is based on CNNs and the decoder is based on feed-forward neural networks (FFNs) and multi-head attention. Our encoder takes $\mathbf{X}$ as an input, exploits temporal and time-frequency structures in $\mathbf{X}$, and outputs the learned audio representation $\mathbf{Z} \in \mathbb{R}^{T_\text{a} \times F'}$, which is a sequence of T_a vectors of F' learned audio features. The decoder takes as an input $\mathbf{Z}$ and outputs $\mathbf{Y}$. Figure 1 illustrates our proposed method.

A. Encoder

Our encoder, $E(\cdot)$, consists of three learnable processes, $E_\text{temp}(\cdot)$, $E_\text{tf}(\cdot)$, and $E_\text{merge}(\cdot)$. E_temp learns temporal context and frame-level information in $\mathbf{X}$ [16], and is inspired by WaveNet [15] but with non-causal convolutions, since in AAC there is no restriction for causality in the encoding of input audio. E_tf learns time-frequency patterns in $\mathbf{X}$, and is inspired by SOTA methods for sound event detection [10], [14], and E_merge merges the information extracted by E_temp and E_tf.

N_t blocks of CNNs (called wave-blocks henceforth) in E_temp, sequentially process $\mathbf{X}$. Each wave-block consists of seven 1D CNNs, $\text{CNN}^{n_\text{t}}_{\text{t}_1}$ to $\text{CNN}^{n_\text{t}}_{\text{t}_7}$, with n_t to be the index of the wave-block. For example, $\text{CNN}^2_{\text{t}_3}$ is the third CNN of the second wave-block. The kernel size, stride, and dilation of $\text{CNN}^{n_\text{t}}_{\{\text{t}_1,\text{t}_4,\text{t}_7\}}$ are one and its padding zero. The kernel size of $\text{CNN}^{n_\text{t}}_{\{\text{t}_2,\text{t}_3\}}$ is three and its padding, dilation, and stride is one. The kernel size of $\text{CNN}^{n_\text{t}}_{\{\text{t}_5,\text{t}_6\}}$ is three, its padding and dilation are two, and stride is one. $\text{CNN}^{n_\text{t}}_{\text{t}_1}$ has $C^{n_\text{t}}_\text{in}$ and $C^{n_\text{t}}_\text{out}$ input and output channels, respectively, and the rest have $C^{n_\text{t}}_\text{out}$ input and output channels.

The above hyper-parameters are based on the WaveNet architecture [15]. The output of the n_t-th wave-block, $\mathbf{H}^{n_\text{t}}_\text{t}$, is obtained by

$$\mathbf{H}''^{n_\text{t}}_{\text{t}_1} = \text{CNN}^{n_\text{t}}_{\text{t}_1}(\mathbf{H}^{n_\text{t}-1}_\text{t}), \tag{1}$$

$$\mathbf{S}''^{n_\text{t}}_\text{t} = \tanh(\text{CNN}^{n_\text{t}}_{\text{t}_2}(\mathbf{H}''^{n_\text{t}}_{\text{t}_1})) \odot \sigma(\text{CNN}^{n_\text{t}}_{\text{t}_3}(\mathbf{H}''^{n_\text{t}}_{\text{t}_1})), \tag{2}$$

$$\mathbf{H}'^{n_\text{t}}_\text{t} = \text{CNN}^{n_\text{t}}_{\text{t}_4}(\mathbf{S}''^{n_\text{t}}_\text{t}) + \mathbf{H}''^{n_\text{t}}_{\text{t}_1}, \tag{3}$$

$$\mathbf{S}'^{n_\text{t}}_\text{t} = \tanh(\text{CNN}^{n_\text{t}}_{\text{t}_5}(\mathbf{H}'^{n_\text{t}}_\text{t})) \odot \sigma(\text{CNN}^{n_\text{t}}_{\text{t}_6}(\mathbf{H}'^{n_\text{t}}_\text{t})), \text{ and} \tag{4}$$

$$\mathbf{H}^{n_\text{t}}_\text{t} = \text{ReLU}(\text{BN}^{n_\text{t}}_\text{t}(\text{CNN}^{n_\text{t}}_{\text{t}_7}(\mathbf{S}'^{n_\text{t}}_\text{t}) + \mathbf{H}'^{n_\text{t}}_{\text{t}_1})), \tag{5}$$

where $\text{BN}^{n_\text{t}}_\text{t}$ is the batch normalization process at the n_t-th wave-block, ReLU is the rectified linear unit, $\sigma(\cdot)$ is the sigmoid non-linearity, $\odot$ is the Hadamard product, $\mathbf{H}^0_\text{t} = \mathbf{X}_\text{t}$, and $\mathbf{H}^{N_\text{t}}_\text{t} \in \mathbb{R}^{C^{n_\text{t}}_\text{out} \times T_\text{a}}_{\geq 0}$. The output of E_temp, $\mathbf{Z}_\text{t} = E_\text{temp}(\mathbf{X}_\text{t})$, is obtained by reshaping $\mathbf{H}^{N_\text{t}}_\text{t}$ to $\{1 \times T_\text{a} \times C^{n_\text{t}}\}$. All CNN^{n_t} operate along the time dimension of $\mathbf{X}_\text{t}$, allowing $\mathbf{H}^{N_\text{t}}_\text{t}$ to learn temporal information from $\mathbf{X}_\text{t}$ [15] and be used effectively in WaveTransformer for learning information that requires temporal context, e.g. spectro-temporal relationships. The time receptive field of each wave-block spans seven time-steps of its corresponding input, leading to a receptive field of $7N_t - 1$ time-steps of $\mathbf{X}$, for the output of the N_t-th wave-block.

E_tf employs N_{tf} blocks of 2D CNNs, called 2DCNN-blocks henceforth. Each 2DCNN-block consists of a 2D CNN

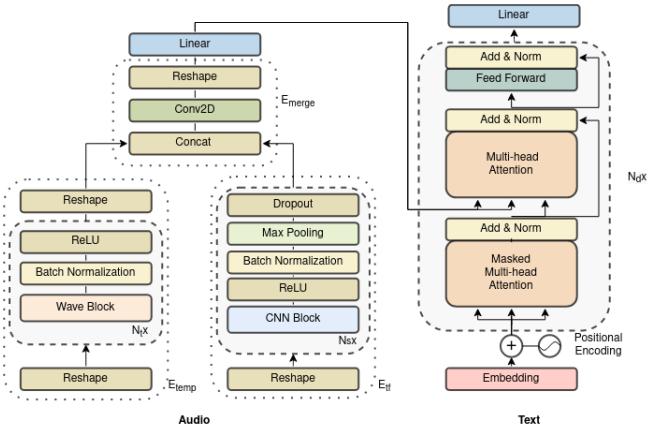

Fig. 1. The WaveTransformer, with the encoder on the left-hand side and the decoder on the right-hand side

(S-CNN$^{n_\text{tf}}$), a leaky ReLU (LU), and a 2D CNN (P-CNN$^{n_\text{tf}}_\text{tf}$). Each 2DCNN-block is followed by a ReLU, a BN (BN$^{n_\text{tf}}$) process, a max-pooling (MP$^{n_\text{tf}}$) process that operates only on the feature dimension (hyper-parameters according to [10]), and a dropout (DR) with probability of p_{n_tf}. The 2DCNN-blocks are inspired by AAC and sound event detection and classification methods, and the recent, successful adoption of depth-wise separable convolutions [13], [10]. The 2DCNN-blocks learn spatial time-frequency information from their input [10], allowing $\mathbf{H}^{N_\text{d}}_\text{d}$ to be used effectively for the identification of sources and actions [10].

S-CNN$^{n_\text{tf}}$ consists of $C^{n_\text{tf}}_\text{in}$ different (5, 5) kernels with unit stride, and padding of 2, focusing on learning time-frequency patterns from each channel of its input. Each kernel of S-CNN$^{n_\text{tf}}$ is applied to only one channel of the input to S-CNN$^{n_\text{tf}}$, according to the depthwise separable convolution model and to enforce the learning of spatial time-frequency patterns [10]. P-CNN$^{n_\text{tf}}_\text{tf}$ consists of a square kernel of size $K_\text{P-CNN} > 1$, with unit stride, and padding of 2, focusing on learning cross-channel information from the output of S-CNN$^{n_\text{tf}}$, since the kernels of P-CNN$^{n_\text{tf}}_\text{tf}$ operate on all channels of the input to P-CNN$^{n_\text{tf}}_\text{tf}$.

While hyper-parameters of S-CNN$^{n_\text{tf}}$ and S-CNN$^{n_\text{tf}}$ are based on [10], the usage of $K_\text{P-CNN} > 1$ is not according to a typical point-wise convolution (i.e. with a (1, 1) kernel, unit stride, and zero padding), as it was experimentally found that it performs better, using the training and validation data, and the protocol described in Section 3. S-CNN^1 has $C^{n_\text{tf}}_\text{in} = 1$ and $C^{n_\text{tf}}_\text{out} = C^{n_\text{t}}_\text{out}$ input and output channels, respectively. S-CNN$^{n_\text{tf}>1}$ and P-CNN$^{n_\text{tf}}$ have input and output channels equal to $C^{n_\text{t}}_\text{out}$. The output of the n_tf-th 2DCNN-block, $\mathbf{H}^{n_\text{tf}}_\text{tf} \in \mathbb{R}^{C^{n_\text{tf}}_\text{out} \times T_\text{a} \times F'_\text{tf}}_{\geq 0}$, is obtained by

$$\mathbf{S}'^{n_\text{tf}}_\text{tf} = \text{P-CNN}^{n_\text{tf}}(\text{BN}^{n_\text{tf}}(\text{LU}(\text{S-CNN}^{n_\text{tf}}(\mathbf{H}^{n_\text{tf}-1}_\text{tf})))) \text{ and} \tag{6}$$

$$\mathbf{H}^{n_\text{tf}}_\text{tf} = \text{DR}(\text{MP}^{n_\text{tf}}(\text{BN}^{n_\text{tf}}(\mathbf{S}'^{n_\text{tf}}_\text{tf}))), \tag{7}$$

where $\mathbf{H}^0_\text{tf} = \mathbf{X}_\text{tf}$ and $\mathbf{H}^{N_\text{tf}}_\text{tf} \in \mathbb{R}^{C^{N_\text{tf}}_\text{out} \times T_\text{a} \times 1}_{\geq 0}$. Then, $\mathbf{Z}_\text{tf} =$

$E_{\text{tf}}(\mathbf{X}_{\text{tf}})$ is obtained by reshaping $\mathbf{H}_{\text{tf}}^{N_{\text{tf}}}$ to $\{1 \times T_{\text{a}} \times C_{\text{out}}^{N_{\text{tf}}}\}$.

E_{merge} consists of a 2D CNN, CNN_{m} and a feed-forward neural network (FNN), FNN_{m}, with shared weights through time. Specifically, CNN_{m} has a (5, 5) kernel with unit stride and dilation, padding of 2, and two input and one output channels. Both $\mathbf{Z}_{\text{t}}$ and $\mathbf{Z}_{\text{tf}}$ have the same dimensionality, are concatenated in their channel dimension, and given as an input to CNN_{m}, as $\mathbf{Z}'' = [\mathbf{Z}_{\text{t}}; \mathbf{Z}_{\text{tf}}]$ and $\mathbf{Z}' = \text{CNN}_{\text{m}}(\mathbf{Z}'')$, where $\mathbf{Z}'' \in \mathbb{R}_{\geq 0}^{2 \times T_{\text{a}} \times C_{\text{out}}^{N_{\text{tf}}}}$, and $\mathbf{Z}' \in \mathbb{R}^{1 \times T_{\text{a}} \times C_{\text{out}}^{N_{\text{tf}}}}$ is the output of CNN_{m}. $\mathbf{Z}'$ is then reshaped to $\{T_{\text{a}} \times C_{\text{out}}^{N_{\text{tf}}}\}$ and given as an input to FNN_{m}, as $\mathbf{Z} = \text{FNN}_{\text{m}}(\mathbf{Z}')$, where $\mathbf{Z} \in \mathbb{R}^{T_{\text{a}} \times F'}$, with $F' = C_{\text{out}}^{N_{\text{tf}}}$.

B. Decoder

We employ the decoder of the Transformer model [7] as our decoder, $D(\cdot)$. During training D takes as an input $\mathbf{Y}$ and $\mathbf{Z}$, and outputs a sequence of T_{w} vectors having a probability distribution over W words, $\hat{\mathbf{Y}} \in [0, 1]^{T_{\text{w}} \times W}$. We follow the implementation in [7], employing an FFN as embedding extractor for one-hot encoded words, $\text{FNN}_{\text{emb}}(\cdot)$, a positional encoding process, $P_{\text{enc}}(\cdot)$, N_{dec} decoder blocks, $D^{n_{\text{dec}}}(\cdot)$, and an FFN at the end which acts as a classifier, $\text{FNN}_{\text{cls}}(\cdot)$. FNN_{emb} and FNN_{cls} have their weights shared across the words of a caption. Each $D^{n_{\text{dec}}}$ consists of a masked multi-head self-attention, a layer-normalization (LN) process, another multi-head attention that attends at $\mathbf{Z}$, followed by another LN, an FNN, and another LN.

We model each $D^{n_{\text{dec}}}$ as a function taking two inputs, $\mathbf{U}^{n_{\text{dec}}} \in \mathbb{R}^{T_{\text{w}} \times V_{\text{e}}^{n_{\text{dec}}}}$ and $\mathbf{Z}$, and having as output $\mathbf{H}_{\text{dec}}^{n_{\text{dec}}} \in \mathbb{R}^{T_{\text{w}} \times V_{\text{e}}^{n_{\text{dec}}}}$, with $\mathbf{H}_{\text{dec}}^{0} = \mathbf{H}'_{\text{dec}}$, $\mathbf{U}^{0} = \mathbf{Y}$, and $V_{\text{e}}^{0} = W$. All FNNs of each $D^{n_{\text{dec}}}$ have input-output dimensionality of $V_{\text{e}}^{n_{\text{dec}}}$. We use N_{att} attention heads and for the multi-head attention layers and p_{d} dropout probability. For the implementation details, we refer the reader to the paper of Transformer model [7]. FNN_{emb} takes as an input $\mathbf{Y}$ and its output is processed by the positional encoding process, as

$$\mathbf{H}'_{\text{dec}} = P_{\text{enc}}(\text{FNN}_{\text{emb}}(\mathbf{Y}))), \tag{8}$$

where P_{enc} is according to the original paper [7]. $\mathbf{H}'_{\text{dec}}$ is processed serially by the N_{dec} decoder blocks, as $\mathbf{H}_{\text{dec}}^{n_{\text{dec}}} = D^{n_{\text{dec}}}(\mathbf{H}_{\text{dec}}^{n_{\text{dec}}-1}, \mathbf{Z})$, and then we obtain $\hat{\mathbf{Y}}$ as

$$\hat{\mathbf{Y}} = \text{FNN}_{\text{cls}}(\mathbf{H}_{\text{dec}}^{N_{\text{dec}}}). \tag{9}$$

We optimize jointly the parameters of the encoder and decoder, by minimizing the cross-entropy loss between $\mathbf{Y}$ and $\hat{\mathbf{Y}}$.

III. Evaluation

To evaluate our method, we employ the dataset and protocol defined at the AAC task at the DCASE2020 challenge. The code and the pre-trained weights of our method are freely available online[1]. We also provide an online demo of our method, with 10 audio files, the corresponding predicted captions, and the corresponding ground truth captions[2].

[1]https://github.com/haantran96/wavetransformer

[2]https://haantran96.github.io/wavetransformer-web-demo/

A. Dataset and pre- and post-processing

We employ the freely available and well curated AAC dataset, Clotho, consisting of around 5000 audio samples of CD quality, 15 to 30 seconds long, and each sample is annotated by human annotators with five captions of eight to 20 words, amounting to around 25 000 captions [4], [17]. Clotho is divided in three splits: i) development, with 14465 captions, ii) evaluation, with 5225, and iii) testing with 5215 captions. We employ development and evaluation splits which are publicly and freely available. We extract $F = 64$ log mel-band energies using Hamming window of 46ms with 50% overlap from the audio files, resulting to $1292 \leq T_{\text{a}} \leq 2584$, for audio samples whose length is between 15 and 30 seconds. During training, to mitigate the length difference of the audio samples in Clotho, in each mini-batch we make all input audio samples to have same length by pre-pending zeros to the shorter ones.

We process each caption and we prepend and append the <sos> (start-of-sentence) and <eos> (end-of-sentence) tokens, respectively. Additionally, we process the development split and we randomly select and reserve 100 audio samples and their captions in order to be used as a validation split during training. These 100 samples are selected according to the criterion that their captions do not contain a word that appears in the captions of less than 10 audio samples. We term the resulting training (i.e. development minus the 100 audio samples) and validation splits as Dev_{tra} and Dev_{val}, respectively. We also provide the file names from Clotho development split used in Dev_{val}, at the online repository of WaveTransformer[2]. We post-process the output of WaveTransformer during inference, employing both greedy and beam search decoding. Greedy decoding stops when <eos> token or when 22 words are generated. During training, to mitigate the length difference of the captions in Clotho, in each mini-batch we make all captions to have same length by appending <eos> tokens to the shorter ones.

B. Hyper-parameters, training, and evaluation

We employ the Dev_{tra} (as training split) and Dev_{val} (as validation split) to optimize the hyper-parameters of our method, using an early stopping policy with a patience of 10 epochs. We employ Adam optimizer [18], a batch size of 12, and clipping of the 2-norm of the gradients to the value of 1. The employed hyper-parameters of our method are $N_{\text{t}} = 4$, $N_{\text{tf}} = 3$, $C_{out}^{n_{\text{t}}} = V_{\text{e}} = 128$, $F'_{tf} = 1$, $N_{\text{dec}} = 3$, $N_{\text{att}} = 4$, $p_{n_{\text{tf}}} = p_{\text{d}} = 0.25$, and beam size of 2. This leads to the modelling of $7N_{\text{t}} - 1 = 27$ frames, equivalent to 0.7 seconds for current $\mathbf{X}$, for E_{temp}.

To assess the performance of WaveTransformer (WT) and the impact of E_{temp}, E_{tf}, E_{merge}, and beam search, we employ the WT, WT without E_{tf} and E_{merge} (WT_{temp}), without E_{temp} and E_{merge} (WT_{tf}), and without E_{merge} (WT_{avg}), where we replace E_{merge} with an average between E_{temp} and E_{tf}. We evaluate the performance of WT with greedy decoding and with beam searching (indicated as WT-B) on Clotho evaluation split and using the machine translation metrics BLEU_1 to

TABLE I
RESULTS ON CLOTHO EVALUATION DATASET. B_n STANDS FOR $BLEU_n$. BOLDFACE FONTS INDICATE THE BEST VALUES FOR EACH METRIC

Model	B_1	B_2	B_3	B_4	METEOR	$ROUGE_L$	CIDEr	SPICE	SPIDEr
TRACKE (w/o MT) [6]	50.2	29.9	18.3	10.2	14.1	**33.7**	23.3	09.1	16.2
NTT (w/o MT, DA, and PP) [11]	**52.1**	29.4	17.4	10.3	13.8	33.5	23.2	08.5	15.8
NTT (MT+PP, w/o DA) [11]	52.0	**31.2**	**20.0**	**12.7**	14.0	33.7	26.1	08.2	17.2
WT_{temp}	45.8	25.9	15.4	08.8	13.9	32.0	19.8	08.7	14.2
WT_{tf}	47.9	28.0	17.1	10.2	14.7	33.1	24.7	09.3	17.0
WT_{avg}	47.9	28.1	17.1	10.3	14.8	33.0	24.7	09.4	17.0
WT	48.4	28.2	17.4	10.2	**14.8**	33.2	24.7	**09.9**	17.3
WT-B	49.8	30.3	19.7	12.0	14.3	33.2	**26.8**	09.5	**18.2**

$BLEU_4$ scores, METEOR, and $ROUGE_L$ [19], [20], [21], and the captioning metrics CIDEr, SPICE, and SPIDEr [22], [23], [24]. In a nutshell, $BLEU_n$ measures a weighted geometric mean of modified precision of n-grams, METEOR measures a harmonic mean of recall and precision for segments between the two captions, and $ROUGE_L$ calculates an F-measure using the longest common sub-sequence. On the other hand, CIDEr calculates a weighted cosine similarity of n-grams, using term-frequency inverse-document-frequency weighting, SPICE measures how well the predicted caption recovers objects, attributes, and their relationships, and SPIDEr is the average of CIDEr and SPICE, exploiting the advantages of CIDEr and SPICE.

Additionally, we compare our method with the two highest-performing AAC methods, NTT [11] and TRACKE [6], developed and evaluated using only Clotho development and evaluation splits. NTT uses different components, like multi-task learning (MT), data augmentation (DA), and post-processing (PP), but authors provide results without these components. TRACKE is the current SOTA, it also uses MT but the authors provide results without MT. We compare our WT against TRACKE without MT and NTT without (w/o) DA.

IV. RESULTS

Table I presents the results of WT, NTT, and TRACKE, where our comparison is limited to methods that are using only the publicly available splits of Clotho. It must be noted that both the systems presented at the papers of the NTT and TRACKE methods, employ data augmentation (DA) and/or multi-task learning (MT) schemes, achieving higher SPIDEr. Since WT is not employing MT and DA, in Table I we compare to the version of NTT and TRACKE methods that have similar set-ups as the WT. As can be seen, the learning of time-frequency information (WT_{tf}) can lead to better results than learning temporal information (WT_{temp}) instead. We hypothesize that this is because the decoder can learn an efficient language model, filling the connecting gaps (e.g. interactions of objects) between sound events learned from E_{tf}. However, from the results it can be seen that employing both E_{temp} and E_{tf} increases more the performance of the WaveTransformer (WT).

Comparing the different scores for the employed metrics and for the WT_{tf} and WT cases, shows that the utilization of E_{temp} is not contributing much in the ordering of words, as indicated by the difference of BLEU metrics between WT_{tf} and WT. We can see that with the E_{temp}, our method learns better attributes of objects and their relationships, as indicated by CIDEr and SPICE scores. Thus, we argue that E_{temp} contributes in learning attributes and interactions of objects, while E_{tf} contributes information about objects and actions (e.g. sound events). Also, by observing the results for WT_{avg}, we can see that a simple averaging of the learned information by E_{temp} and E_{tf} leads to a better description of objects, attributes, and their relationships (indicated by SPICE). Though, as can be seen by comparing WT_{avg} and WT, the E_{merge} manages to successfully merge the information by E_{temp} and E_{tf}. The utilization of beam search (WT-B) gives a significant boost to the performance, reaching up to 18.2 SPIDEr. Compared to TRACKE and NTT methods, we can see that when excluding DA, MT, and PP, our method (WT) performs better. Additionally, WT-B performs better than NTT with MT and PP. Our post-processing consists only on using beam search, where the NTT method involves a second post-processing technique by augmenting the input data and averaging the predictions. Thus, WT surpasses the other methods that is compared against.

Finally, two, high SPIDEr-scoring, captions are for the files *Flipping pages.wav*, and *110422_village_dusk.wav* of the evaluation split of Clotho. Our predicted captions for each of these files, using WT-B, are: "a person is flipping through the pages of a book" and "a dog is barking while birds are chirping in the background", respectively, and the best matching ground truth captions are "a person is flipping through pages in a notebook" and "a dog is barking in the background while some children are talking and birds are chirping", respectively.

V. CONCLUSION

In this paper we presented a novel architecture for AAC, based on convolutional and feed-forward neural networks, called WaveTransformer (WT). WT focuses on learning long temporal and time-frequency information from audio, and expressing it with text using the decoder of the Transformer model. We evaluated WT using the dataset and the metrics adopted in the AAC DCASE Challenge, and we compared our method against previous SOTA methods and the DCASE AAC baseline. The obtained results show that learning time-frequency information, combined with a good language model, can lead to good AAC performance, but incorporating long temporal information can boost the obtained scores.

ACKNOWLEDGEMENT

The authors wish to thank D. Takeuchi and Y. Koizumi for their input on previously reported results, and to acknowledge CSC-IT Center for Science, Finland, for computational resources. Part of the needed computations was implemented on a GPU donated from NVIDIA to K. Drossos. Part of the work leading to this publication has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957337, project MARVEL.

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.

[2] M. Wu, H. Dinkel, and K. Yu, "Audio caption: Listen and tell," *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019.

[3] K. Nguyen, K. Drossos, and T. Virtanen, "Temporal sub-sampling of audio feature sequences for automated audio captioning," in *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2020)*, 2020.

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

[5] J. Naranjo-Alcazar, S. Perez-Castanos, P. Zuccarello, and M. Cobos, "Task 6 dcase 2020: Listen carefully and tell: An audio captioning system based on residual learning and gammatone audio representation," Tech. Rep., DCASE2020 Challenge, Jun. 2020.

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

[7] A. Vaswani et al., "Attention is all you need," in *31st Conference on Neural Information Processing Systems (NeurIPS 2017)*, 2017.

[8] A. Shi, "Audio captioning with the transformer automated audio captioning," Tech. Rep., DCASE2020 Challenge, 2020.

[9] D. Serdyuk, N.-R. Ke, A. Sordoni, A. Trischler, C. Pal, and Y. Bengio, "Twin Networks: Matching the future for sequence generation," *CoRR*, vol. abs/1708.06742, 2017.

[10] 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)*, 2020.

[11] D. Takeuchi, Y. Koizumi, Y. Ohishi, N. Harada, and K. Kashino, "Effects of word-frequency based pre- and post- processings for audio captioning," in *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2020)*, 2020.

[12] H. Wang, B. Yang, Y. Zou, and D. Chong, "Automated audio captioning with temporal attention," Tech. Rep., DCASE2020 Challenge, 2020.

[13] E. Çakır, K. Drossos, and T. Virtanen, "Multi-task regularization based on infrequent classes for audio captioning," in *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2020)*, 2020.

[14] E. Çakır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, 2017.

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

[16] Hyungui Lim, J. Park, K. Lee, and Yoonchang Han, "Rare sound event detection using 1d convolutional recurrent neural networks," Tech. Rep., DCASE2020 Challenge, 2017.

[17] 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.

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

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

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

[21] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, 2004.

[22] 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.

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

[24] 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.