STREAMING TRANSFORMER TRANSDUCER BASED SPEECH RECOGNITION USING NON-CAUSAL CONVOLUTION

Yangyang Shi, Chunyang Wu*, Dilin Wang*, Alex Xiao, Jay Mahadeokar, Xiaohui Zhang, Chunxi Liu, Ke Li, Yuan Shangguan, Varun Nagaraja, Ozlem Kalinli, Mike Seltzer*

Facebook AI

ABSTRACT

This paper improves the streaming transformer transducer for speech recognition using non-causal convolution. Many works apply the causal convolution to improve streaming transformer ignoring the lookahead context. We propose to use non-causal convolution to process the center block and lookahead context separately. This method leverages the lookahead context in convolution and maintains similar training and decoding efficiency. Given the similar latency, using the non-causal convolution with lookahead context gives better accuracy than causal convolution, especially for open-domain dictation. Besides, this paper applies talking-head attention and a novel history context compression scheme to further improve the performance. The talking-head attention improves the multi-head self-attention by transferring information among different heads. The history context compression method introduces more extended history context compactly. On our in-house data, the proposed methods improve a small Emformer baseline with lookahead context by relative WERR 5.1%, 14.5%, 8.4% on open-domain dictation, assistant general scenarios, and assistant calling scenarios respectively.

***Index Terms*—** Non-causal convolution, talking heads, augmented memory

1. INTRODUCTION

Nowadays, sequence transducer networks [1, 2] are widely used for streaming automatic speech recognition due to their superior performance and compactness. A sequence transducer model has an encoder to capture the context information from acoustic signals, a predictor to model the grammar, syntactic, and semantic information, and a joiner to combine the two parts. The work [3, 4] showed replacing the LSTM encoder with the self-attention-based transformer [5] yielded the state-of-the-art of accuracy on public benchmark datasets, which is consistent to the trend in applying transformer in various scenarios for automatic speech recognition [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17].

One popular variant of transformer models for speech recognition tasks is Conformer [15], which adds the depth separable convolutions and macaron network structure [18] into the transformer. The work [19, 20] simplified the depth-wise convolution in conformer to causal convolution to support streaming scenarios. In [17], the non-causal convolution is used to support streaming speech recognition which uses sequential block processing to avoid training and decoding inconsistency. The sequential block processing segments the input sequences into multiple blocks. And the model is sequentially trained on each block. The sequential block process is inefficient in training, especially for low latency scenarios, and incapable of dealing with the large-scale dataset.

The multi-head self-attention [5] in transformer uses different heads that conduct the attention computation separately. The attention outputs are concatenated at the end. The work [21] proposed a talking-heads attention method to break the separation among different heads by inserting two other learnable lightweight linear projections are transferring information across these heads.

The Emformer [14] and the augmented memory transformer [16] equips the streaming transformer with an augmented memory scheme to store the compact long-form context while maintaining limited computation and runtime memory consumption in inference.

This work advances the streaming transformer model from the following aspects. First, we improve the streaming conformer with a non-causal convolution and parallel block processing. The non-causal convolution equips the streaming conformer to leverage the lookahead context, enhancing ASR accuracy. The parallel block processing proposed in [14] is vital for efficient streaming transformer model training, especially for low latency and large-scale datasets. Second, this work replaces the attention with the talking-heads attention scheme to reduce the ASR word error rate further. Third, we simplify the augmented memory extraction process similar to [22], referred to as context compression. Rather than using the self-attention output from the mean of each block, we directly utilize the linear interpolation of each block as memory. We conducted large-scale speech recognition experiments to evaluate the above three techniques in this work.

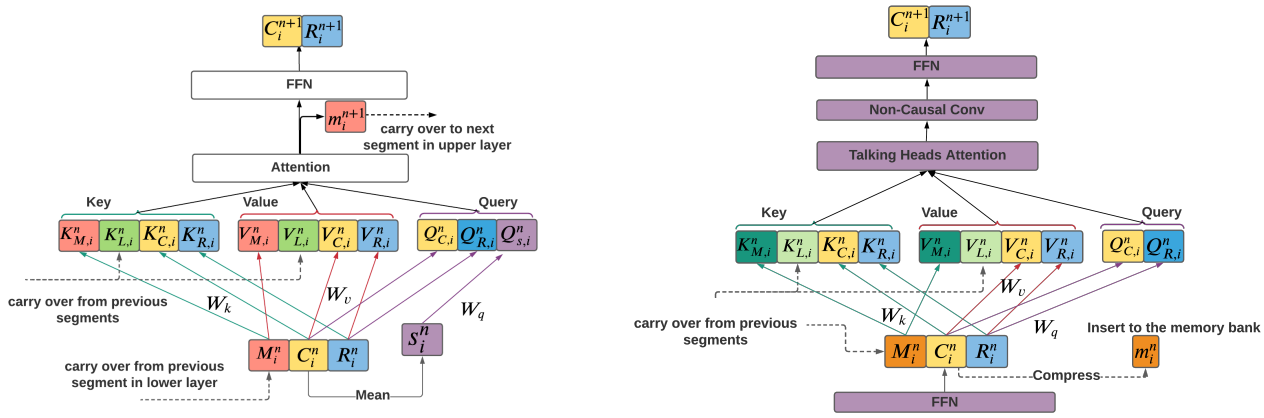
The rest of this paper is organized as follows. In Section 2, we present the methods to advance the Emformer model. Section 3 demonstrates and analyzes the experimental results, followed by a conclusion in Section 4.

2. METHODS TO ADVANCE EMFORMER

Fig. (1a) illustrates forward logic in one Emformer[14] layer. To support streaming speech recognition, Emformer applies the parallel block processing to segment a input sequence into multiple non-overlapping blocks $\mathbf{C}_1^n, \cdots, \mathbf{C}_{i-1}^n$, where i denotes the index of current block, and n denotes the layer's index. In order to reduce boundary effect where the most right vector in $\mathbf{C}_i^n$ has no lookahead context information, a right contextual block $\mathbf{R}_i^n$, is concatenated with $\mathbf{C}_i^n$ to form a contextual block $\mathbf{X}_i^n = [\mathbf{C}_i^n, \mathbf{R}_i^n]$. At the i-th block, the n-th Emformer layer takes $\mathbf{X}_i^n$ and a bank of memory vector $\mathbf{M}_i^n$ as the input, and produces $\mathbf{X}_i^{n+1} = [\mathbf{C}_i^{n+1}, \mathbf{R}_i^{n+1}]$ and $\mathbf{m}_i^n$ as the output, whereas $\mathbf{X}_i^{n+1}$ is fed to the next layer and $\boldsymbol{m}_i^n$ is inserted into the memory bank to generate $\mathbf{M}_{i+1}^{n+1}$ and carried over to the next block and next layer.

The modified attention mechanism in emformer attends to the

⋆ Equal contribution.



(a) Emformer

(b) Emformer with non-causal convolution

Fig. 1: Advance Emformer with non-causal convolution, talking-heads attention and simplified compact augmented memory

memory bank and yields a new memory slot at each block:

$$[\hat{\mathbf{C}}_i^n, \hat{\mathbf{R}}_i^n] = \text{LayerNorm}([\mathbf{C}_i^n, \mathbf{R}_i^n]), \tag{1}$$

$$\mathbf{K}_i^n = \left[\mathbf{K}_{L,i}^n, \mathbf{W}_\text{k}[\mathbf{M}_i^n, \hat{\mathbf{C}}_i^n, \hat{\mathbf{R}}_i^n]\right], \tag{2}$$

$$\mathbf{V}_i^n = \left[\mathbf{V}_{L,i}^n, \mathbf{W}_\text{v}[\mathbf{M}_i^n, \hat{\mathbf{C}}_i^n, \hat{\mathbf{R}}_i^n]\right], \tag{3}$$

$$\mathbf{Z}_{\text{C},i}^n = \text{Attn}(\mathbf{W}_\text{q}\hat{\mathbf{C}}_i^n, \mathbf{K}_i^n, \mathbf{V}_i^n) + \mathbf{C}_i^n, \tag{4}$$

$$\mathbf{Z}_{\text{R},i}^n = \text{Attn}(\mathbf{W}_\text{q}\hat{\mathbf{R}}_i^n, \mathbf{K}_i^n, \mathbf{V}_i^n) + \mathbf{R}_i^n, \tag{5}$$

$$\mathbf{M}_i^n = [\mathbf{m}_{i-U}^{n-1}, \cdots, \mathbf{m}_{i-1}^{n-1}], \tag{6}$$

$$\mathbf{m}_i^n = \text{Attn}(\mathbf{s}_i^n, \mathbf{K}_i^n, \mathbf{V}_i^n), \tag{7}$$

$$\mathbf{s}_i^n = \text{Mean}(\mathbf{C}_i^n), \tag{8}$$

where $\mathbf{K}_{L,i}^n$ and $\mathbf{V}_{L,i}^n$ are the *key* and *value* copies from previous blocks. $\mathbf{Z}_{\text{C},i}^n$ and $\mathbf{Z}_{\text{R},i}^n$ are the attention output for $\mathbf{C}_i^n$ and $\mathbf{R}_i^n$ respectively; $\mathbf{s}_i^n$ is the mean of center block $\mathbf{C}_i^n$; $\text{Attn}(\mathbf{q}, \mathbf{k}, \mathbf{v})$ is the attention operation defined in [5] with $\mathbf{q}$, $\mathbf{k}$ and $\mathbf{v}$ being the query, key and value, respectively. U specifies the number of slots in augmented memory; the most recent slots are used.

$\mathbf{Z}_{\text{C},i}^n$ and $\mathbf{Z}_{\text{R},i}^n$ are passed to a point-wise feed-forward network (FFN) with layer normalization and residual connection to generate the output of this Emformer layer, i.e.,

$$\hat{\mathbf{X}}_i^{n+1} = \text{FFN}(\text{LayerNorm}([\mathbf{Z}_{\text{C},i}^n, \mathbf{Z}_{\text{R},i}^n])), \tag{9}$$

$$\mathbf{X}_i^{n+1} = \text{LayerNorm}(\hat{\mathbf{X}}_i^{n+1} + [\mathbf{Z}_{\text{C},i}^n, \mathbf{Z}_{\text{R},i}^n]), \tag{10}$$

where FNN is a two-layer feed-forward network with ReLU.

2.1. Streaming Non-causal Convolution

Fig. (1b) illustrates the improvements applied to advance the Emformer. Different from Eq. (1), the input to attention goes through one step of FFN in macaron structure:

$$[\hat{\mathbf{C}}_i^n, \hat{\mathbf{R}}_i^n] = \text{LayerNorm}\left(\frac{1}{2}\text{FFN}(\mathbf{X}_i^n) + \mathbf{X}_i^n\right). \tag{11}$$

Different from Eq. (9-10), the second FFN in macaron gets the input from the convolution layer.

$$\hat{\mathbf{X}}_i^{n+1} = \text{Conv}(\text{LayerNorm}([\mathbf{Z}_{\text{C},i}^n, \mathbf{Z}_{\text{R},i}^n])), \tag{12}$$

$$\mathbf{X}_i^{n+1} = \text{LayerNorm}\left(\hat{\mathbf{X}}_i^{n+1} + \frac{1}{2}\text{FFN}(\hat{\mathbf{X}}_i^{n+1})\right). \tag{13}$$

The convolution layer in Fig. (1b) has a similar structure as [15], except the layer norm is used right after depth-wise convolution rather than the batch norm. In our experiments, the layer norm gives better performance than the batch norm.

The work [16, 17] uses sequential block processing where the training and streaming decoding do the forward logic in the same way. The self-attention and convolution receptive field is limited by the block size and surrounding context size. It is trivial to use a non-causal convolution operation in this way. However, the sequential block processing is slow in training as it doesn't utilize GPU parallel computation capacity. For low latency situations where the block size is tiny, sequential block processing is not practical to use.

To use the lookahead context in streaming speech recognition, Emformer [14] uses the right-context-hard-copy methods in training. The right-context-hard-copy method copies and concatenates each block C_i^n's lookahead context R_i^n. Then it puts the concatenated lookahead context at the beginning of the input sequence. The right-context-hard-copy method is essential to avoid the lookahead context leaking issue in training, where the higher transformer layer has a larger lookahead context than the bottom layer when multiple transformer layers are stacking on top of the other.

Fig. 2 shows the forward logic of using non-causal convolution operation in Emformer. The output from the attention operation $[\mathbf{Z}_{\text{R},1}^n \dots \mathbf{Z}_{\text{R},t}^n, \mathbf{Z}_{\text{C},1}^n \dots \mathbf{Z}_{\text{C},t}^n]$ is first splitted into two parts: right context $[\mathbf{Z}_{\text{R},1}^n \dots \mathbf{Z}_{\text{R},t}^n]$ and center block $[\mathbf{Z}_{\text{C},1}^n \dots \mathbf{Z}_{\text{C},t}^n]$. Then the same depth-wise convolution is applied to both parts. For the center block part, it is straightforward to directly apply the convolution operation as shown in Eq. (14). The right context part needs to go through reshape, padding, convolution operation and finally be reshaped to its original shape. In padding operation, each right context block is padded with its corresponding block.

$$[\hat{\mathbf{Z}}_{\text{C},1}^n \dots \hat{\mathbf{Z}}_{\text{C},t}^n] = \text{Conv}([\mathbf{Z}_{\text{C},1}^n \dots \mathbf{Z}_{\text{C},t}^n]) \tag{14}$$

$$\mathbf{P}_{\text{R},i}^n = \mathbf{Z}_{\text{C},i}^n[m-k+1:m] \tag{15}$$

$$[\hat{\mathbf{Z}}_{\text{R},1}^n \dots \hat{\mathbf{Z}}_{\text{R},t}^n] = \text{Conv}([\mathbf{P}_{\text{R},1}^n, \mathbf{Z}_{\text{R},1}^n] \dots [\mathbf{P}_{\text{R},t}^n, \mathbf{Z}_{\text{R},t}^n]) \tag{16}$$

where k is the kernel size used in depth-wise convolution. The padding $\mathbf{P}_{\text{R},i}^n$ in Eq. (15) is the ending $k-1$ feature vectors from center block $\mathbf{Z}_{\text{C},i}^n$.

2.2. Talking-heads Attention

Self-attention forms the foundation of transformers. Assume a set of tokens $\mathbf{V} \in \mathbb{R}^{f \times d}$ that is packed into a matrix form and consider

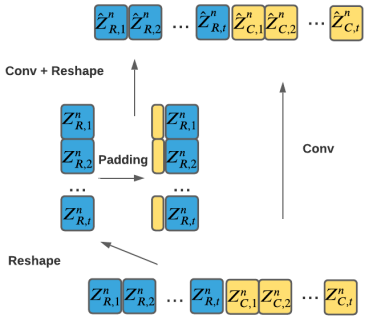


Fig. 2: Training procedure of non-causal convolution in parallel block processing. The same depth-wise convolution does the forward separately for lookahead context (blue) and the center block (yellow). The lookahead context is padded by feature vectors from its corresponding block.

$\mathbf{K} \in \mathbb{R}^{f \times d}$ and $\mathbf{V} \in \mathbb{R}^{f \times d}$ its corresponding keys and queries, respectively. Here f denotes the length of tokens and d is the dimension of each token. Self-attention aggregates information across different tokens and transforms V as follows,

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\mathbf{Q}\mathbf{K}^\top / \sqrt{d}\right)\mathbf{V}.$$

Multi-heads attention assembles multiple standard self-attention blocks for better representation learning,

$$\text{MultiHeadAttn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}\left(\left\{\text{Att}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i)\right\}_i^h\right),$$

where h denotes the number of heads and $\mathbf{Q}_i$, $\mathbf{K}_i$ and $\mathbf{V}_i$ represents the queries, keys and values from different heads, respectively.

One potential drawback of multiple-attention is that different heads are trained independently without coordination. Talking-heads attention [21] improves on multi-heads attention by allowing information fusion among different attention heads. Assume $\text{Softmax}(\mathbf{A})$ the attention weights learned by different heads in multi-head attention ($\mathbf{A} \in \mathbb{R}^{f \times f \times h}$). Talking-heads attention introduces two additional linear layers immediately before and after the softmax and computes the new self-attention weights as follows,

$$\text{Softmax}(\mathbf{A} * \mathbf{W}_l) * \mathbf{W}_r. \tag{17}$$

Here $\mathbf{W}_l \in \mathbb{R}^{h \times h}$ and $\mathbf{W}_r \in \mathbb{R}^{h \times h}$ are trainable parameters, and Softmax is applied on the second dimension. In practice, the two linear projections introduced by talking-heads attention are computationally efficient as the number of heads h used is often small.

2.3. Context Compression

The augmented memory is designed to introduce long-form information into the attention. As shown in Eq. (6-7), the information is introduced via the queries of the previous segments in the previous layer. This inter-layer strategy gets rid of the auto-regression property if it is on the same layer, preventing inefficient block processing in training. However, one potential issue of this design is the representation mismatch between successive layers. In the attention operation, the augmented memory slots from the previous layer and the frames from the current layer are equally treated in key and value, which depends on the similar representations on the two layers. Otherwise, long-form information can be misleadingly introduced.

To address the potential mismatch between memory slots and frames, we put forward the context compression strategy in this paper. The context compression directly introduces compact memory to the key and value in the attention, not to the query. It is formalized as follows,

$$\mathbf{m}_i^n = \text{Compress}(\mathbf{C}_i^n), \tag{18}$$

$$\mathbf{M}_i^n = [\mathbf{m}_{i-U}^n, \cdots, \mathbf{m}_{i-1-O}^n] \tag{19}$$

where the Compress operation stands for a function that can compress the segment into one single vector, e.g., linear interpolation or average pooling; this work chooses the linear interpolation. Contrasting to Eq. (19), an offset term O is introduced in Eq. (6), which is intended to prevent the overlap between the short-form left context and this long-form compressed context. For instance, on a model with a segment size of 4 and a left context of 8, we set an offset of 2 to skip the interval covered by the left context. According to Eq. (18), the context compression operates the input C of each layer, preventing the auto-regression between successive segments; thus the whole sequence can be trained in parallel, thoroughly taking advantage of the graphics computing resource.

3. EXPERIMENTS

3.1. Data

Our training data is a large-scale speech recognition dataset composed of two scenarios. The *assistant* scenario consists of three parts. One is 13K hours of recordings collected from third-party vendors via crowd-sourced volunteers responding to artificial prompts with mobile devices. The content varies from voice assistant commands to a simulation of conversations between people. The second is 1.3K hours of voice commands from production. The last is 4K hours of speech for calling names and phone numbers generated by an in-house TTS model. The *open domain* dictation has 18K hours of human transcribed data from video and 2M hours of unlabeled videos transcribed by a high-quality in-house model. The data was augmented with various distortion methods: speed perturbation [23], simulated reverberation SpecAugment [24], and randomly sampled additive background noise extracted from videos.

In evaluation, we use *assi*, *call* and *dict* dataset. The *assi* and *call* are 13.6K manually transcribed utterances from in-house volunteer employees, and each utterance starts with a wake word. The *dict* is 8 hours open domain dictation from crowd-sourced workers recorded via mobile devices.

3.2. Experiment Setting

The input features are 80-dim log Mel filter bank features at a 10ms frame rate; The network's input is a 640-dim superframe consists of 8 consistent frames with a downsampling factor of 8 to 80ms frame rate. This paper explored models with 32M parameters and 73M parameters. In the 32M parameter baseline model, a projection layer maps the superframe to a 320-dim vector. The encoder consists of 21 Emformer layers. Each layer uses four heads for self-attention, and its FFN-block dimension is 1280. The predictor consists of a 256-dim embedding layer with 4096 sentence pieces [25], 1 LSTM layer with 512 hidden nodes, and a linear projection layer with 1024 output nodes. The baseline with 32M parameters uses a left context of 640ms (10 slots) in the left context. For the block size and right context, two settings are investigated. One is the block size of 320ms (4 slots) and right context of 80ms (1 slot); the other is a block size of 400ms (5 slots) and a right context of 0 (0 slots). In the 73M parameter model, the superframe is mapped to a 512-dim vector. The

encoder has 20 layers of Emformer. Each layer has an 8-head self-attention and a 2048-dim FFN block. Its predictor has the same layer configuration as the 32M baseline, but the number of LSTM layers is 3. The left context is set to 2.4s, i.e., 30 slots in the left context. In training, on the 73M parameter model, *SpecAugment* [24] without time warping, and dropout 0.1 are used. We found that the 32M parameter models are underfitting a large amount of training data. The best performance is obtained by not using either scheme.

For our proposed models, we first investigate the non-causal convolution. A kernel size seven is used for depth-wise convolution operations. In the 32M parameter model, the superframe is projected to a 256-dim vector. In the 73M parameter model, the superframe is projected to a 384-dim vector. It consists of 20 layers containing an 8-head self-attention and a 1456-dim FFN block in each layer. It consists of 18 layers containing a 4-head self-attention and a 1024-dim FFN block in each layer. Other settings are the same as the baselines. The block size and right context are fixed as 320ms and 80ms, respectively. For the context compression scheme, we use a regular left context of 8 slots, implying 640ms. The compressed left context is set to 2 slots, implying 640ms; also, it uses an offset of 2, O in Eq. (19), to skip the same interval of the 640ms regular left context. In total, ten slots are used but implying a history of 1280ms.

In all the experiments, alignment restrict RNNT [26] is used. The training of all the models uses 32 Nvidia V100 GPUs. We evaluate the models by word error rate (WER) for accuracy and the real-time factors (RTFs) and speech engine perceived latency (SPL) for latency. The SPL measures the time the speech engine gets the last word from user utterance to the speech engine transcribes the last word and gets the endpoint signals.

3.3. Improvement from Non-causal Convolution

Table. 1 gives the WER, RTF, and SPL results for models with 32M and 73M parameters. The results show that by keeping the overall context size (the sum of block size and lookahead context size) the same, using lookahead context gives WER improvement over not using it, especially for open domain dictation scenarios. We also observe that convolution and macaron structure improves the baseline using the same context configuration. Table. 1 also show that the direct application of causal convolution with 400ms block size does not improve the baseline model which leverages 320ms block size with 80ms lookahead context for both 32M and 73M models.

Using lookahead context adds more computation for encoders in transducer model, as the forward logic has duplicated computation for the lookahead context. For the 73M model, using right side context 80ms shows 10% relative RTF increase. However, lookahead context provides more accurate ASR results and yields slightly better speech perceived latency (SPL).

3.4. Improvement from Talking-heads Attention and Context Compression

Table. 2 shows the impact of applying talking-heads attention and context compression on top of Emformer with non-causal convolutions. For the model with 32M parameters, the talking-heads attention generates 4.6%, 3.8%, and 2.8% relative WER reductions on open-domain dictation, assistant general queries, and assistant calling queries, respectively. Using two slots of context compression outperforms the model with only regular left context. Combining non-casual convolution, talking-heads, and context compression in the 32M model improves the WER by 5.1%, 14.5%, 8.4% relatively on open-domain dictation, assistant general, and assistant calling test sets, while maintaining similar SPL and RTF as the Emformer baseline. For the model with 73M parameters, talking-heads attention and context compression obtain on par WER as the Emformer with non-causal convolution. Note the 73M parameters baseline uses 30 slots of left context, while context compression uses 8 slots of left context and 2 slots of memory which slightly improves the RTF and SPL. However, the 73M model already has a much stronger model capacity than the 32M model. The lightweight optimizations of talking-heads attention and context compression do not generate obvious improvement.

Table 1: WER, RTF and SPL impact from non-causal convolution and lookahead context.Column '#p' gives the number of parameters in each model. Column 'w/C' denotes whether the convolution is applied or not. 'C' and 'R' represents the block size and lookahead context size. The unit for 'C', 'R' and 'SPL' is millisecond.

#p	w/c	C	R	*dict*	*ass*	*call*	SPL	RTF
73M	N	400	0	16.78	4.18	6.19	606	0.27
		320	80	15.49	3.98	5.81	599	0.30
73M	Y	400	0	16.11	4.05	5.94	615	0.27
		320	80	14.67	3.65	5.85	595	0.30
32M	N	400	0	18.31	5.17	6.57	635	0.21
		320	80	17.09	5.05	6.68	588	0.22
32M	Y	400	0	17.70	4.78	6.76	626	0.22
		320	80	17.02	4.66	6.53	605	0.22

Table 2: WER and RTF and SPL impact from the context compression. Column 'L' stands for the length of left context. Column 'CL' stands for the length of compressed left context. The unit of both columns is slot: for 'L', 1 slot means 80ms; for 'CL', 1 slot implies 320ms, i.e. block size.

Model	*dict*	*ass*	*call*	SPL	RTF
73M Emformer (baseline)	15.49	3.98	5.81	599	0.30
+ Non-causal	14.67	3.65	5.85	595	0.30
+ Talk heads	14.69	3.64	5.79	574	0.30
+ Context Compression	14.69	3.66	5.77	554	0.29
32M Emformer (baseline)	17.09	5.05	6.68	588	0.22
+ Non-causal	17.02	4.66	6.53	605	0.22
+ Talk heads	16.25	4.48	6.35	620	0.23
+ Context Compression	16.22	4.32	6.12	589	0.24

4. CONCLUSIONS

In this work, we proposed to use non-causal convolution, talking heads attention, and context compression to improve the streaming transformer transducer for speech recognition. This work managed to apply non-causal convolution with lookahead context in streaming transformer by separating the forward logic for the center block and lookahead context. The talking-heads attention coordinates the training of different heads in self-attention. The context compression keeps the representation in the long-form and short-form history similar, providing a compact way of introducing long-form information. The experiments on 32M parameter and 73M parameter models show that the proposed model outperforms the Emformer baseline on open-domain dictation, assistant general, and assistant calling scenarios while maintaining comparable RTF and latency.

5. REFERENCES

[1] Alex Graves, "Sequence Transduction with Recurrent Neural Networks," *arXiv preprint arXiv:1211.3711*, 2012.

[2] Y He, T N Sainath, R Prabhavalkar, et al., "Streaming End-to-end Speech Recognition for Mobile Devices," in *Proc. ICASSP*, 2019.

[3] Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, and Shankar Kumar, "Transformer Transducer: A Streamable Speech Recognition Model with Transformer Encoders and RNN-T Loss," in *Proc. ICASSP*, 2020.

[4] Ching-Feng Yeh, Jay Mahadeokar, Kaustubh Kalgaonkar, and Others, "Transformer-Transducer: End-to-End Speech Recognition with Self-Attention," *arXiv preprint arXiv:11910.12977*, 2019.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Proc. NIPS*, 2017.

[6] L Dong, S Xu, and B Xu, "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition," in *Proc. ICASSP*, 2018.

[7] S Karita, N Chen, T Hayashi, and Others, "A Comparative Study on Transformer vs RNN in Speech Applications," *arXiv preprint arXiv:1909.06317*, 2019.

[8] M Sperber, J Niehues, G Neubig, et al., "Self-attentional acoustic models," *arXiv preprint arXiv:1803.09519*, 2018.

[9] S Zhou, L Dong, S Xu, and B Xu, "Syllable-based sequence-to-sequence speech recognition with the transformer in mandarin Chinese," *arXiv preprint arXiv:1804.10752*, 2018.

[10] Chengyi Wang, Yu Wu, Shujie Liu, Jinyu Li, Liang Lu, Guoli Ye, and Ming Zhou, "Low Latency End-to-End Streaming Speech Recognition with a Scout Network," *arXiv preprint arXiv:12003.10369*, 2020.

[11] Frank Zhang, Yongqiang Wang, Xiaohui Zhang, Chunxi Liu, Yatharth Saraf, and Geoffrey Zweig, "Fast, Simpler and More Accurate Hybrid ASR Systems Using Wordpieces," *InterSpeech*, 2020.

[12] D Povey, Hossein Hadian, P Ghahremani, and Others, "A time-restricted self-attention layer for asr," in *Proc. ICASSP*, 2018.

[13] Yongqiang Wang, Abdelrahman Mohamed, Due Le, Chunxi Liu, Alex Xiao, Jay Mahadeokar, Hongzhao Huang, Andros Tjandra, Xiaohui Zhang, Frank Zhang, Christian Fuegen, Geoffrey Zweig, and Michael L. Seltzer, "Transformer-Based Acoustic Modeling for Hybrid Speech Recognition," in *Proc. ICASSP*, 2019.

[14] Yangyang Shi, Yongqiang Wang, Chunyang Wu, Ching-Feng Yeh, and Others, "Emformer: Efficient Memory Transformer Based Acoustic Model For Low Latency Streaming Speech Recognition," in *Proc. ICASSP*, 2021.

[15] Anmol Gulati, James Qin, Chung Cheng Chiu, et al., "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. INTERSPEECH*, 2020.

[16] Chunyang Wu, Yangyang Shi, Yongqiang Wang, and Ching-Feng Yeh, "Streaming Transformer-based Acoustic Modeling Using Self-attention with Augmented Memory," in *Proc. INTERSPEECH*, 2020.

[17] Ching Feng Yeh, Yongqiang Wang, Yangyang Shi, et al., "Streaming attention-based models with augmented memory for end-to-end speech recognition," in *Proc. SLT*, 2020.

[18] Yiping Lu, Zhuohan Li, Di He, Zhiqing Sun, Bin Dong, Tao Qin, Liwei Wang, and Tie-Yan Liu, "Understanding and Improving Transformer From a Multi-Particle Dynamic System Point of View," *arXiv preprint arXiv:1906.02762*, 2019.

[19] Xie Chen, Yu Wu, Zhenghao Wang, Shujie Liu, and Jinyu Li, "Developing Real-Time Streaming Transformer Transducer for Speech Recognition on Large-Scale Dataset," in *Proc. ICASSP*, 2021, pp. 5904–5908.

[20] Jiahui Yu, Chung-Cheng Chiu, Bo Li, and Others, "Fastemit: Low-Latency Streaming Asr With Sequence-Level Emission Regularization," in *Proc. ICASSP*, 2021, vol. 53.

[21] Noam Shazeer, Zhenzhong Lan, Youlong Cheng, Nan Ding, and Le Hou, "Talking-Heads Attention," *arXiv preprint arXiv:2003.02436*, 2020.

[22] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap, "Compressive Transformers for Long-Range Sequence Modelling," *arXiv preprint arXiv:1911.05507*, 2019.

[23] T Ko, V Peddinti, D Povey, and Others, "Audio augmentation for speech recognition," in *Proc. INTERSPEECH*, 2015.

[24] D S Park, W Chan, Y Zhang, et al., "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[25] Taku Kudo and John Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *Proc. EMNLP*, 2018.

[26] Jay Mahadeokar, Yuan Shangguan, Duc Le, and Others, "Alignment restricted streaming recurrent neural network transducer," in *Proc. SLT*, 2021.