INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Toward Streaming ASR with Non-Autoregressive Insertion-based Model

Yuya Fujita[1], *Tianzi Wang*[2], *Shinji Watanabe*[3], *Motoi Omachi*[1]

[1]Yahoo Japan Corporation, Tokyo, Japan
[2]Johns Hopkins University, USA
[3]Carnegie Mellon University, USA

{yuyfujit, momachi}@yahoo-corp.jp, wtianzi1@jhu.edu, shinjiw@iee.org

Abstract

Neural end-to-end (E2E) models have become a promising technique to realize practical automatic speech recognition (ASR) systems. When realizing such a system, one important issue is the segmentation of audio to deal with streaming input or long recording. After audio segmentation, the ASR model with a small real-time factor (RTF) is preferable because the latency of the system can be faster. Recently, E2E ASR based on non-autoregressive models becomes a promising approach since it can decode an N-length token sequence with less than N iterations. We propose a system to concatenate audio segmentation and non-autoregressive ASR to realize high accuracy and low RTF ASR. As a non-autoregressive ASR, the insertion-based model is used. In addition, instead of concatenating separated models for segmentation and ASR, we introduce a new architecture that realizes audio segmentation and non-autoregressive ASR by a single neural network. Experimental results on Japanese and English dataset show that the method achieved a reasonable trade-off between accuracy and RTF compared with baseline autoregressive Transformer and connectionist temporal classification.

Index Terms: ASR, end-to-end, non-autoregressive, audio segmentation

1. Introduction

End-to-end (E2E) models have become a promising option to realize practical automatic speech recognition (ASR) systems based on the significant improvement of the ASR performance [1–8]. In particular, the E2E model is suitable for ASR systems run on a device like a smartphone which only has limited computing capability. Most of the E2E models are composed of a single neural network and its computational optimization is simple. Therefore, many techniques to make the model footprint lower like quantization and compression, are easily applied. This leads to several works that aim to run E2E ASR models on a device like smartphones [9–11].

One important issue to be solved when realizing a practical ASR system is the segmentation of the input audio. If the input audio is long or is fed in a streaming way, the audio should be segmented into an appropriate length of utterances to avoid large memory consumption. In particular, estimating the end of the speech segment is important because it also affects latency which directly relates to the user's subjective impression of the performance of the ASR system. However, realizing low latency and high accuracy is a trade-off. One way to realize such an ASR system is to separately employ models for segmentation and ASR.

In such a scenario, it is obvious that making the decoding process faster leads to lower latency. Therefore, decoding with a small real-time factor (RTF) is important. Recently, non-autoregressive Transformer (NAT) is intensively investigated in the research field of neural machine translation [12–15]. It aims to realize faster decoding than an autoregressive Transformer at the expense of a small loss of accuracy. The main contribution to the RTF improvement of NAT is the ability to decode N-length token sequences with less than N iterations, which can not be achieved by autoregressive Transformer. Mask-predict, one of the NAT models, was applied to ASR and realized significant improvement of RTF with a small degradation in accuracy [16].

In this paper, we aim to realize ASR of streaming input or long recording with non-autoregressive ASR. Among several works of non-autoregressive ASR [16–19], we propose to use the insertion-based model recently proposed in [19] because of its high accuracy and small numbers of iterations required for decoding. The model jointly trains connectionist temporal classification (CTC) [20] and insertion-based models. It achieves better accuracy with approximately $\log_2(N)$ iterations during inference than autoregressive Transformer with greedy decoding, while the other non-autoregressive models like mask CTC [17] does not reach this performance. Our first attempt in this proposed framework is to concatenate audio segmentation and this insertion-based model toward streaming ASR.

In addition, we propose to integrate audio segmentation and non-autoregressive ASR in a single neural network by exploiting the CTC part of the insertion-based model with causal self-attention. Causal self-attention is realized by block self-attention which is similar to Transformer XL [21]. It computes attention weights inside a limited context (block). In general, introducing causal self-attention degrades accuracy. However, once the audio segment is fixed, the insertion-based model can refine the hypothesis by looking at the whole acoustic feature in the segment with a smaller number of iterations than the autoregressive Transformer. Therefore, the accuracy can be recovered while achieving faster inference than the autoregressive Transformer.

Experimental results showed the proposed method achieved a good balance between accuracy and RTF compared with baseline autoregressive Transformer and CTC. To the best of our knowledge, this is the first work to apply non-autoregressive ASR to streaming or long recording.

2. Related Work

Several works of non-autoregressive Transformer for ASR have been proposed [16–18]. However, all of them do not aim for recognizing streaming input or long recording. There are some works of realizing audio segmentation and ASR in a single model. In [22], it is proposed to use the CTC part of a jointly trained CTC and attention-based model. Integrating an endpointer into a single RNN-T model and second-stage rescoring with an attention-based model is proposed in [23]. It assumes a voice search application which only detects a single endpoint of

Copyright © 2021 ISCA

http://dx.doi.org/10.21437/Interspeech.2021-1131

streaming audio. Both of them uses autoregressive model while ours are non-autoregressive one.

3. Insertion-based model with CTC

In this section, the insertion-based model used in this work, KERMIT (Kontextuell Encoder Representations Made by Insertion Transformations) [24], is first introduced. Then, joint modeling with CTC proposed in [19] is explained.

3.1. Insertion-based model: KERMIT

First, the general formulation of E2E ASR models and insertion-based models is described. Suppose $X = (\mathbf{x}_t \in \mathbb{R}^d | t = 1, \cdots, T)$ is an acoustic feature sequence whose dimension is d. The output token sequence is defined as $C = (c_n \in \mathcal{V} | n = 1, \cdots, N)$. T is the length of the acoustic feature, N is the length of output token sequence, and $\mathcal{V}$ is a set of distinct tokens. The E2E ASR model is a probability distribution over the output token sequence C given the acoustic feature sequence X, i.e. $p^{\text{e2e}}(C|X)$. It is modeled by a single neural network.

In the case of insertion-based models, the probability distribution $p^{\text{e2e}}(C|X)$ is assumed to be marginalized over insertion order Z:

$$p^{\text{e2e}}(C|X) = \sum_Z p(C, Z|X) = \sum_Z p(C^Z|X)p(Z). \quad (1)$$

Insertion order Z represents the permutation of the token sequence C, e.g. if $C = (A, B, C, D)$ and $Z = (3, 1, 2, 4)$, $C^Z = (C, A, B, D)$. For the KERMIT case, let c_i^Z be a token to be inserted and l_i^Z be a position where the token is inserted at the i-th generation step under an insertion order Z, $p(C^Z|X)$ in Eq. (1) is defined as:

$$\begin{aligned} p(C^Z|X) &=: \prod_{i=1}^{I} p\left(\left(c_i^Z, l_i^Z\right) | \left(c_0^Z, l_0^Z\right), \cdots, \left(c_{i-1}^Z, l_{i-1}^Z\right), X\right) \\ &= \prod_{i=1}^{I} p\left(\left(c_i^Z, l_i^Z\right) | {c^{Z'}}_{0:i-1}, X\right), \end{aligned} \quad (2)$$

where $i = 1, \cdots, I$ is the index of generation step and ${c^{Z'}}_{0:i-1}$ is the sorted token sequence at the $(i-1)$-th generation step. The posterior in Eq. (2) is modeled by only the encoder block of Transformer. The output at the final layer of the encoder block is sliced as $\mathbf{H}^{\text{tok}} \in \mathbb{R}^{d^{\text{sa}} \times i}$ then used to calculate the posterior:

$$p\left(\left(c_i^Z, l_i^Z\right) | {c^{Z'}}_{0:i-1}, X\right) =: \underbrace{p(c_i^Z | l_i^Z, \mathbf{H}^{\text{tok}})}_{\text{Token prediction}} \underbrace{p(l_i^Z | \mathbf{H}^{\text{tok}})}_{\text{Position prediction}}, \quad (3)$$

where d^{sa} is the dimension of self-attention. The token and position prediction term are calculated by a linear transformation of $\mathbf{H}^{\text{tok}}$ and softmax. Non-autoregressive parallel decoding is possible using only the token prediction term in Eq. (3):

$$\hat{c} = \arg\max_c p\left(c | l, {c^{Z'}}_{0:i-1}, X\right). \quad (4)$$

When $p(Z)$ is the balanced binary tree (BBT) insertion order proposed in [15], decoding finishes empirically with $\log_2(N)$ iterations. The BBT order is to insert the centermost tokens of the current hypothesis. For example, given a sequence $C = (c_1, \cdots, c_9)$, the hypothesis grows based on the tree structure like $(c_5) \rightarrow (c_3, c_5, c_7) \rightarrow (c_2, c_3, c_4, c_5, c_6, c_7, c_8)$ $\rightarrow (c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9)$.

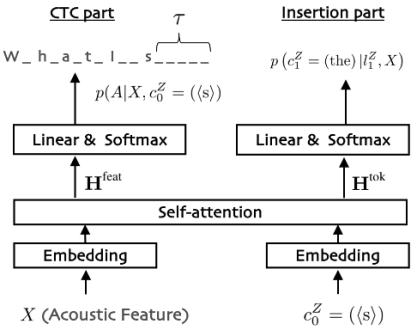

Figure 1: *Schematic diagram of KERMIT and joint modeling with CTC.*

Reference: フレーズ指令

Iteration	Hyp. of CTC part Y	Hyp. of Insertion part ${c^{Z'}}_{1:i-1}$
1	取レーズ指令	ズ
2	プレーズ指令	レズ指
3	フレーズ指令	フレーズ指令

Figure 2: *Decoding example of KERMIT and CTC.*

3.2. Joint modeling with CTC

Next, joint modeling of KERMIT and CTC is introduced [19]. In general, joint modeling is to model a joint distribution over different sequences. Suppose Y is another output token sequence for joint modeling. Then, the posterior in Eq. (2) is modified as:

$$p(C^Z|X) =: p(C^Z, Y|X) = p(Y|X, C^Z)p(C^Z|X). \quad (5)$$

The term $p(Y|X, C^Z)$ in Eq. (5) can be any kind of probability distribution. We set $Y = C$ and choose CTC for the term because joint modeling with CTC results in faster convergence and performance improvement [4]. Suppose A is a sequence of tokens including a blank symbol $\langle \text{b} \rangle$, i.e. $A = (a_t \in \mathcal{V} \cup \{\langle \text{b} \rangle\} | t = 1, \cdots, T)$. $\mathcal{F}(\cdot)$ is a function which deletes repetitions and the blank symbol from the sequence A hence $\mathcal{F}(A) = Y$. The CTC probability is formulated as:

$$p(Y|X, C^Z) =: \sum_{A \in \mathcal{F}^{-1}(Y)} p(A|X, C^Z). \quad (6)$$

Because all the output of KERMIT is dependent on both the acoustic feature sequence and the token sequence, the final output of the encoder block of KERMIT is sliced as $\mathbf{H}^{\text{feat}} \in \mathbb{R}^{d^{\text{sa}} \times T}$ and used:

$$p(A|X, C^Z) =: p(A|\mathbf{H}^{\text{feat}}). \quad (7)$$

This process is depicted in Figure 1. It can be seen as a multi-task training of the two terms in Eq. (5), i.e. $p(Y|X, C^Z)$ in Eq. (6) and $p(C^Z|X)$ in Eq. (2). The formulation means CTC is reinforced by insertion-based token sequence generation. During decoding, either of the CTC part $p(Y|X, C^Z)$ or the insertion part $p(C^Z|X)$ in Eq. (5) can be used as a final result. Figure 2 shows an example of the decoding process. The error in the original CTC result (Y in the first iteration) was refined after the iterations.

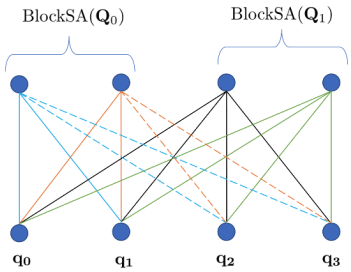

Figure 3: *Example of the block self-attention ($T_q = 4, L = 2$). In the case of block self-attention, dotted lines are not computed.*

4. Audio segmentation and non-autoregressive decoding

This section proposes to extend the model described in Section 3.2 to enable joint audio segmentation and non-autoregressive decoding.

4.1. Preliminary: Block self-attention

In the model described in Section 3.2, self-attention is the component that prevents the model being used for audio segmentation of long recordings. This is because self-attention computes attention weights over the whole input sequence. In order to avoid this, block self-attention, which computes attention weights inside a limited context (block), is introduced. This is almost the same structure as Transformer XL [21] except this formulation uses future context inside a block and passes gradients to the previous block.

First, we define multi-headed attention [25] as follows:

$$\mathbf{U}_h = \text{Attention}(\mathbf{Q}\mathbf{W}_h^{\text{Q}}, \mathbf{K}\mathbf{W}_h^{\text{K}}, \mathbf{V}\mathbf{W}_h^{\text{V}}), \quad (8)$$

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\mathbf{U}_1, \cdots, \mathbf{U}_H)\mathbf{W}^{\text{O}}, \quad (9)$$

where $\mathbf{Q} \in \mathbb{R}^{T^{\text{q}} \times d^{\text{att}}}$, $\mathbf{K} \in \mathbb{R}^{T^{\text{k}} \times d^{\text{att}}}$, and $\mathbf{V} \in \mathbb{R}^{T^{\text{v}} \times d^{\text{att}}}$ denote query, key, and value matrices, respectively. $T^{\text{q}}, T^{\text{k}}$, and T^{v}, are the length of each elements and d^{att} is the dimension of the input to MultiHead$(\cdot)$. $h = 1, \cdots, H$ is the index of the head. $\mathbf{W}_h^{\text{Q}}, \mathbf{W}_h^{\text{K}}, \mathbf{W}_h^{\text{V}} \in \mathbb{R}^{d^{\text{att}} \times d^{\text{att}}/H}$ and $\mathbf{W}^{\text{O}} \in \mathbb{R}^{d^{\text{att}} \times d^{\text{att}}}$.

Self-attention is the multi-headed attention whose inputs are the same, i.e. $\mathbf{Q} = \mathbf{K} = \mathbf{V}$:

$$\text{SA}(\mathbf{Q}) = \text{MultiHead}(\mathbf{Q}, \mathbf{Q}, \mathbf{Q}). \quad (10)$$

Block self-attention introduces block length B and its index b, and define query matrix of b-th block $\mathbf{Q}_b \in \mathbb{R}^{B \times d^{\text{att}}}$ as:

$$\mathbf{Q}_b = \left[\mathbf{q}_{b \times B}, \mathbf{q}_{b \times B+1}, \cdots, \mathbf{q}_{(b+1) \times B-1}\right]. \quad (11)$$

Then, block self-attention at b-th block is defined by setting $\mathbf{Q} = \mathbf{Q}_b, \mathbf{K} = \mathbf{V} = [\mathbf{Q}_{b-1}, \mathbf{Q}_b]$, i.e.:

$$\text{BlockSA}(\mathbf{Q}_b) = \text{MultiHead}(\mathbf{Q}_b, [\mathbf{Q}_{b-1}, \mathbf{Q}_b], [\mathbf{Q}_{b-1}, \mathbf{Q}_b]). \quad (12)$$

Note that when computing the b-th block, only the B frames of future context are required. This avoids computing attention weights over the whole input sequence, hence realizing segmentation of long or streaming input audio. Another benefit is computational cost. The self-attention of length T^{q} requires $\mathcal{O}\left((T^{\text{q}})^2\right)$ computations. In the block self-attention with block length B requires $\mathcal{O}\left(B^2 \times \lfloor T^{\text{q}}/B \rfloor\right)$. This computation is depicted in Figure 3.

4.2. Proposed joint modeling with block self-attention

First, in order to apply joint modeling of KERMIT and CTC in Section 3.2, we further extend block self-attention in Eq. (12) to consider extra input $\mathbf{M} \in \mathbb{R}^{T^{\text{m}} \times d^{\text{att}}}$. This extension is realized by setting $\mathbf{K} = \mathbf{V} = [\mathbf{Q}_{b-1}, \mathbf{Q}_b, \mathbf{M}]$ of multi-headed attention, i.e.:

$$\begin{aligned}&\text{ExtBlockSA}(\mathbf{Q}_b, \mathbf{M}) \\ &\quad = \text{MultiHead}(\mathbf{Q}_b, [\mathbf{Q}_{b-1}, \mathbf{Q}_b, \mathbf{M}], [\mathbf{Q}_{b-1}, \mathbf{Q}_b, \mathbf{M}]), \quad (13)\end{aligned}$$

where T^{m} is the length of extra input.

Now, we consider the acoustic feature sequence X. For ease of explanation, the acoustic feature sequence is assumed to be segmented, e.g. training phase where segment information is given or inference phase where the segment is estimated. The segmented acoustic feature sequence is passed to an audio embedding layer to obtain an embedding matrix $\mathbf{X}^{\text{E}} \in \mathbb{R}^{T^{\text{SS}} \times d^{\text{att}}}$, where T^{SS} is the segmented sequence length after subsampling. In order to apply block self-attention, same as Eq. (11), b-th block of $\mathbf{X}^{\text{E}}$ is defined as $\mathbf{X}_b^{\text{E}} = \left[\mathbf{x}_{b \times B}, \mathbf{x}_{b \times B+1}, \cdots, \mathbf{x}_{(b+1) \times B-1}\right]$ where $b = 0, \ldots, \lfloor \frac{T^{\text{SS}}}{B} \rfloor - 1$.

In our joint modeling, we use the partial hypothesis $C^{\text{hyp}} = (c_{1:i})$ at i-th generation step as an extra input, which is artificially generated according to $p(Z)$ in the case of the training phase. It is passed to a character embedding layer to obtain an embedding matrix $\mathbf{C}^{\text{hyp}} \in \mathbb{R}^{N^{\text{hyp}} \times d^{\text{att}}}$, where N^{hyp} is the length of the hypothesis.

Thus, the proposed forward pass of KERMIT using block self-attention defined by Eq. (13) is

$$\begin{cases} \mathbf{Z}_b^{(j)} = \text{ExtBlockSA}(\mathbf{Z}_b^{(j-1)}, \mathbf{Y}_{(j-1)}) \text{ for } b = 0, \ldots, \lfloor \frac{T^{\text{SS}}}{B} \rfloor - 1, \\ \mathbf{Y}^{(j)} = \text{MultiHead}(\mathbf{Y}^{(j-1)}, \mathbf{K}, \mathbf{V}) \\ \qquad \text{where } \mathbf{K} = \mathbf{V} = \left[[\mathbf{Z}_b^{(j-1)}]_{b=0,\cdots,\lfloor \frac{T^{\text{SS}}}{B} \rfloor - 1}, \mathbf{Y}^{(j-1)}\right], \end{cases} \quad (14)$$

where $j = 1, \cdots, J$ is the index of the encoder layer and $\mathbf{Z}_b^{(0)} = \mathbf{X}_b^{\text{E}}$ and $\mathbf{Y}^{(0)} = \mathbf{C}^{\text{hyp}}$. Note that ExtBlockSA$(\cdot)$ and MultiHead$(\cdot)$ in Eq. (14) share the parameters. In the final layer J, $\mathbf{Z}^{(J)}$ corresponds to $\mathbf{H}^{\text{feat}}$ in the CTC part of Figure 1 (also introduced in Eq. (7)), while $\mathbf{Y}^{(J)}$ corresponds to $\mathbf{H}^{\text{tok}}$ in the insertion part of Figure 1 (also introduced in Eq. (3)).

4.3. Audio segmentation and decoding

Once the model is trained, audio segmentation and decoding are processed as follows. Different to the segmented case explained in Section 4.2, when the input audio is long or is fed in a streaming way, b in Eq. (14) is very long or unbounded. Therefore, we proposed to segment audio by using only the first line of Eq. (14) because this operation is possible at every B frames of input are obtained by using the embedding $\mathbf{C}^{\text{sos}}$ of $C = \{\langle s \rangle\}$ as $\mathbf{Y}^{(0)}$. Then, output of CTC probability $p(a_{(b-1) \times B}, \cdots, a_{b \times B} | \mathbf{H}^{\text{feat}})$ in Eq. (7) is computed by using $\mathbf{Z}_b^{(J)}$ as $\mathbf{H}^{\text{feat}}$. If the blank symbol has the highest probability of more than τ consecutive frames, i.e.:

$$\arg\max_{a_t} p(a_t | \mathbf{H}^{\text{feat}}) = \langle \text{b} \rangle \text{ for } t = b \cdot B - \tau, \cdots, b \cdot B, \quad (15)$$

Table 1: *CER and RTF of CSJ when oracle segmentation and full-attention model is used.*

	Eval1	***Eval2***	***Eval3***	RTF
ART	**7.5**	**5.0**	**12.2**	0.98
CTC	8.6	5.9	13.9	0.27
KERMIT	**7.5**	**5.0**	**12.2**	**0.21**

Table 2: *CER and RTF of CSJ when separated TDNN-based audio segmentation and full-attention model is used.*

	Eval1	***Eval2***	***Eval3***	RTF
ART	**9.8**	**6.7**	**14.3**	0.92
CTC	10.5	6.8	15.3	0.31
KERMIT	10.1	**6.7**	14.4	**0.23**

the end of the current audio segment is detected as $b \times B$. This is the same strategy proposed in [22]. Suppose the index of current audio segment is r and its end as T_r^{end}, r-th segment is decoded with $b = T_{r-1}^{\text{end}}, \cdots, T_r^{\text{end}}$ by both line of Eq. (14) and iteratively updating $\mathbf{Y}^{(0)}$ by new hypothesis.

5. Experiments

5.1. Setup

The proposed framework is evaluated on the Corpus of Spontaneous Japanese (CSJ) [26] and TEDLIUM2 [27]. Note that for CSJ, only the Academic lecture data, whose amount is around 270 hours, is used for training. The unsegmented evaluation set is used to evaluate the performance of audio segmentation and ASR. The segmentation is performed in two ways. The first one is to employ a separated model for segmentation. We used a model provided as a baseline of CHiME6 challenge [28]. The model is based on time-delayed neural network (TDNN) architecture and trained on a target label generated by a highly tuned hybrid ASR model trained on CHiME6 training data. The second one is using the integrated CTC part as described in Section 4.3. Baseline models are autoregressive Transformer (ART) and CTC. Both of the baseline models can use integrated CTC part for segmentation in the same way as described in Section 4.3 by using block-SA. Note that there can be a mismatch of the length of the segmented audio compared with oracle segmentation used in training in both ways. Hence it is not apparent that the combination of audio segmentation and non-autoregressive ASR works or not.

Implementation is based on the recipe of ESPnet [29]. Most of the parameters are almost the same as used in [19] except the following two points. The learning rate and warmup steps were adjusted to stabilize training, and the number of epochs is set to 200 for the KERMIT with block-SA. The segmentation threshold τ, as discussed in Section 4.3, is set to 8, 16, or 24 depending on the models according to a preliminary experiment. The RTF is measured with Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz using two threads for the forward propagation of a neural network. The beam size of ART and CTC and the number of iteration of KERMIT is set to 5.

5.2. Results and Discussion

The character error rate (CER) and real time factor (RTF) of CSJ of each methods are shown in Table 1 to 3. The oracle segmentation results in Table 1 showed that KERMIT achieved the best CER, which is the same as ART, but the RTF is smaller than ART. This is the advantage of non-autoregressive ASR.

As shown in Table 2, when using the separated TDNN model for segmentation, ART achieved the best CER, but KERMIT also achieved competitive CER with around 1/4 RTF compared to ART. In the case of using integrated CTC as segmentation, as shown in Table 3, KERMIT is in a good balance between CER and RTF. The RTF of CTC is a bit smaller, but its CER is worse than KERMIT. ART was the worst CER and RTF. The integrated approach does not use any alignment nor segmentation criterion while training. This can lead to a significant mismatch between estimated segment length and oracle segment, and the autoregressive model would not be robust on such a mismatch. The word error rate (WER) and RTF of TEDLIUM2 are shown in Table 4. Same as the CSJ case, KERMIT is in a good balance between WER and RTF.

Table 3: *CER and RTF of CSJ when integrated CTC-based audio segmentation with block-SA is used.*

Future context [sec]		***Eval1***	***Eval2***	***Eval3***	RTF
0.19	ART	15.5	11.1	21.9	1.54
	CTC	14.8	11.2	21.8	**0.45**
	KERMIT	**11.7**	**8.6**	**17.3**	0.54
0.67	ART	14.1	9.9	19.6	1.62
	CTC	12.8	9.3	18.4	**0.39**
	KERMIT	**10.8**	**7.7**	**16.2**	0.45

Table 4: *WER and RTF of TEDLIUM2. For the integrated CTC-based segmentation, future context is set to 0.19 sec.*

Segmentation		***dev***	***test***	RTF
Oracle	ART	**10.2**	**9.1**	1.11
	KERMIT	11.1	9.9	**0.19**
Separated TDNN	ART	**10.4**	**13.4**	0.85
	KERMIT	11.8	14.2	**0.20**
Integrated CTC	ART	17.9	**19.5**	1.15
	KERMIT	**16.5**	19.9	**0.38**

In summary, the proposed combination of TDNN segmentation and KERMIT-based NAT achieved a reasonable performance trade-off in terms of RTF and CER. The performance degradation from the oracle segmentation is acceptable (less than 3%) while keeping around 0.2 RTF, which is suitable for a streaming scenario. Another proposal of the integrated CTC approach also shows promising results since it restricts the future context with 0.67 seconds while still keeping 0.45 RTF and within 4% CER degradation from the oracle segmentation result. Also, the integrated approach is based on a single neural network, and further optimization can mitigate the issue.

6. Conclusion

This paper proposed combining audio segmentation and non-autoregressive ASR toward streaming or long recording audio recognition. Also, we introduced a new architecture that realizes audio segmentation and non-autoregressive ASR by a single neural network. The insertion-based model, which is jointly trained with CTC, is used as non-autoregressive ASR. By employing causal self-attention, the CTC part is used for audio segmentation. Experimental results showed that a combination of audio segmentation and non-autoregressive ASR worked well and achieved a good balance between CER and RTF compared with baseline AR Transformer and CTC. The single model approach also shows promising results, and the improvement of this approach is left as future work.

7. References

[1] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Proc. Advances in Neural Information Processing Systems (NIPS) 28*, 2015, pp. 577–585. [Online]. Available: http://papers.nips.cc/paper/5847-attention-based-models-for-speech-recognition.pdf

[2] D. Amodei *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *Proc. of the 33rd International Conference on International Conference on Machine Learning (ICML)*, 2016, pp. 173–182.

[3] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 4960–4964.

[4] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid ctc/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, Dec 2017.

[5] C. Chiu *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," in *Proc. ICASSP 2018*, April 2018, pp. 4774–4778.

[6] C. Lüscher, E. Beck, K. Irie, M. Kitza, W. Michel, A. Zeyer, R. Schlüter, and H. Ney, "RWTH ASR Systems for LibriSpeech: Hybrid vs Attention," in *Proc. Interspeech 2019*, 2019, pp. 231–235. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-1780

[7] S. Karita *et al.*, "A comparative study on transformer vs RNN in speech applications," *arXiv preprint arXiv:1909.06317*, 2019.

[8] T. N. Sainath, R. Pang, D. Rybach, Y. He, R. Prabhavalkar, W. Li, M. Visontai, Q. Liang, T. Strohman, Y. Wu, I. McGraw, and C.-C. Chiu, "Two-Pass End-to-End Speech Recognition," in *Proc. Interspeech 2019*, 2019, pp. 2773–2777. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-1341

[9] Y. He *et al.*, "Streaming end-to-end speech recognition for mobile devices," in *Proc. ICASSP*, 2019, pp. 6381–6385.

[10] K. Kim *et al.*, "Attention based on-device streaming speech recognition with large speech corpus," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019, pp. 956–963.

[11] T. N. Sainath *et al.*, "A streaming on-device end-to-end model surpassing server-side conventional model quality and latency," in *Proc. ICASSP 2020*, 2020, pp. 6059–6063.

[12] J. Gu, J. Bradbury, C. Xiong, V. O. Li, and R. Socher, "Non-autoregressive neural machine translation," *arXiv preprint arXiv:1711.02281*, 2017.

[13] M. Ghazvininejad, O. Levy, Y. Liu, and L. Zettlemoyer, "Mask-predict: Parallel decoding of conditional masked language models," in *Proceedings of the EMNLP-IJCNLP*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6112–6121. [Online]. Available: https://www.aclweb.org/anthology/D19-1633

[14] J. Gu, C. Wang, and J. Zhao, "Levenshtein transformer," in *Proc. Advances in Neural Information Processing Systems (NIPS) 32*, 2019, pp. 11 181–11 191. [Online]. Available: http://papers.nips.cc/paper/9297-levenshtein-transformer.pdf

[15] M. Stern, W. Chan, J. Kiros, and J. Uszkoreit, "Insertion transformer: Flexible sequence generation via insertion operations," in *Proc. of International Conference on Machine Learning (ICML)*, 2019, pp. 5976–5985.

[16] N. Chen, S. Watanabe, J. Villalba, and N. Dehak, "Listen and fill in the missing letters: Non-autoregressive transformer for speech recognition." *arXiv preprint arXiv:1911.04908*, 2020.

[17] Y. Higuchi, S. Watanabe, N. Chen, T. Ogawa, and T. Kobayashi, "Mask ctc: Non-autoregressive end-to-end asr with ctc and mask predict," *arXiv preprint arXiv:2005.08700*, 2020.

[18] W. Chan, C. Saharia, G. Hinton, M. Norouzi, and N. Jaitly, "Imputer: Sequence modelling via imputation and dynamic programming," *arXiv preprint arXiv:2002.08926*, 2020.

[19] Y. Fujita, S. Watanabe, M. Omachi, and X. Chan, "Insertion-based modeling for end-to-end automatic speech recognition," *arXiv preprint arXiv:2005.13211*, 2020.

[20] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. of the 23rd International Conference on Machine Learning (ICML)*, 2006, pp. 369–376.

[21] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 2978–2988. [Online]. Available: https://www.aclweb.org/anthology/P19-1285

[22] T. Yoshimura, T. Hayashi, K. Takeda, and S. Watanabe, "End-to-end automatic speech recognition integrated with ctc-based voice activity detection," in *Proc. ICASSP*, 2020, pp. 6999–7003.

[23] B. Li *et al.*, "Towards fast and accurate streaming end-to-end asr," in *Proc. ICASSP*, 2020, pp. 6069–6073.

[24] W. Chan, N. Kitaev, K. Guu, M. Stern, and J. Uszkoreit, "Kermit: Generative insertion-based modeling for sequences," *arXiv preprint arXiv:1906.01604*, 2019.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Advances in Neural Information Processing Systems (NIPS) 30*, 2017, pp. 5998–6008. [Online]. Available: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[26] K. Maekawa, H. Koiso, S. Furui, and H. Isahara, "Spontaneous speech corpus of Japanese," in *Proc. the Second International Conference on Language Resources and Evaluation (LREC'00)*, 2000.

[27] A. Rousseau, P. Deléglise, and Y. Estève, "Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, May 2014, pp. 3935–3939.

[28] S. Watanabe *et al.*, "CHiME-6 Challenge: Tackling Multispeaker Speech Recognition for Unsegmented Recordings," in *Proc. The 6th International Workshop on Speech Processing in Everyday Environments (CHiME 2020)*, 2020, pp. 1–7. [Online]. Available: http://dx.doi.org/10.21437/CHiME.2020-1

[29] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "Espnet: End-to-end speech processing toolkit," in *Proc. Interspeech 2018*, 2018, pp. 2207–2211. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2018-1456