# Large-Context Automatic Speech Recognition Based on RNN Transducer

Atsushi Kojima*
* Advanced Media, Inc., Japan
E-mail: a-kojima@advanced-media.co.jp

***Abstract*—We propose large-context end-to-end automatic speech recognition (ASR) based on a recurrent neural network transducer (RNN–T) for conversational ASR. Typical end-to-end ASR models recognize independent utterances. Unlike the typical end-to-end ASR models, attention-based encoder-decoder models, which utilize a large context (i.e., a long-range context beyond the boundary of an utterance) have been recently proposed for conversational ASR. In this work, we introduce a large-context encoder for RNN–T to utilize hypotheses generated in previous utterances as a large context. The large-context encoder obtains concatenated hypotheses and converts them to hidden vectors. The hidden vectors are summarized as the available context based on an attention mechanism. During training, the large-context encoder and RNN–T components are jointly optimized using RNN–T loss. In our experiments, we evaluate the proposed method using medical conversations as a dataset. As a result of the experiment, we obtained 17.2% and 6.0% relative reductions in the character error rate compared with those of the original RNN–T on utterances spoken by the doctor and patient, respectively.**

## I. INTRODUCTION

There has been growing interest recently in building end-to-end automatic speech recognition (ASR) models such as an attention-based encoder-decoder [1], [2], recurrent neural network transducer (RNN–T) [3], Transformer transducer [4], [5] and connectionist temporal classification (CTC) [6]. Typically, they convert acoustic features to a sequence of characters or subwords for each utterance independently.

Unlike typical end-to-end ASR models, attention-based encoder-decoder models, which utilize a large context (i.e., a long-range context beyond the boundary of an utterance), such as hypotheses generated in previous utterances, have been recently proposed for conversational ASR [7], [8], [9], [10]. In these works, end-to-end ASR models that utilize a large context achieved better results than typical end-to-end ASR models.

In this work, we propose a large-context end-to-end ASR model based on RNN–T [3]. For some conversations such as medical conversations and confidential related internal meetings, the end-to-end ASR model, which can run on-device, is important to protect privacy. Among the end-to-end ASR models, RNN–T has been successfully applied for on-device ASR [11]. In this work, we introduce a large-context encoder for RNN–T to utilize hypotheses generated in previous utterances as a large context. A large-context encoder that consists of Transformer [12] obtains the concatenated hypotheses and converts them to hidden vectors. The hidden vectors are summarized as the available context based on an attention mechanism [13]. During training, the large-context encoder is optimized with RNN–T components using RNN–T loss.

In the experiment, we evaluate the proposed method using medical conversations by a doctor and patient as a dataset. As a result of the experiment, we obtained 17.2% and 6.0% relative reductions in the character error rate (CER) compared with those of the original RNN–T on utterances spoken by the doctor and patient, respectively [1].

## II. BACKGROUND

### A. RNN–T

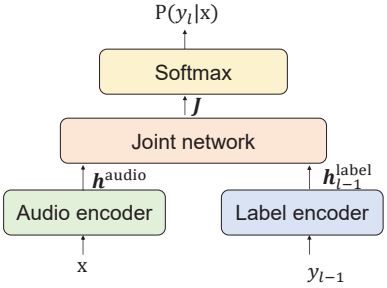


Fig. 1. Architecture of RNN–T.

RNN–T is an end-to-end ASR model. Fig. 1 shows an overview of RNN–T. The model consists of a label encoder, an audio encoder and a joint network. Given acoustic feature $\mathrm{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T)$ and previous tokens $\boldsymbol{y} = (y_1, y_2, \cdots, y_{L-1})$, the audio encoder converts acoustic feature $\boldsymbol{x}$ to hidden vector $\boldsymbol{h}^{\text{audio}}$, and the label encoder outputs a new hidden vector $\boldsymbol{h}^{\text{label}}$ based on previous tokens except for the blank token. The joint network outputs vector $J$ using two hidden vectors from the audio and label encoders and softmax outputs logits. The audio and label encoders consist of long short-term memory (LSTM) layers and the joint network consists of a feed-forward layer.

RNN–T is trained using RNN–T loss [3]. Given acoustic features $\boldsymbol{x}$ and label sequence $\boldsymbol{y}$, the neural transducer outputs $T \times L$ logits. RNN–T loss is calculated as the sum of probabilities for all paths using a forward–backward algorithm.

[1] Code available at https://github.com/atsushiKojima

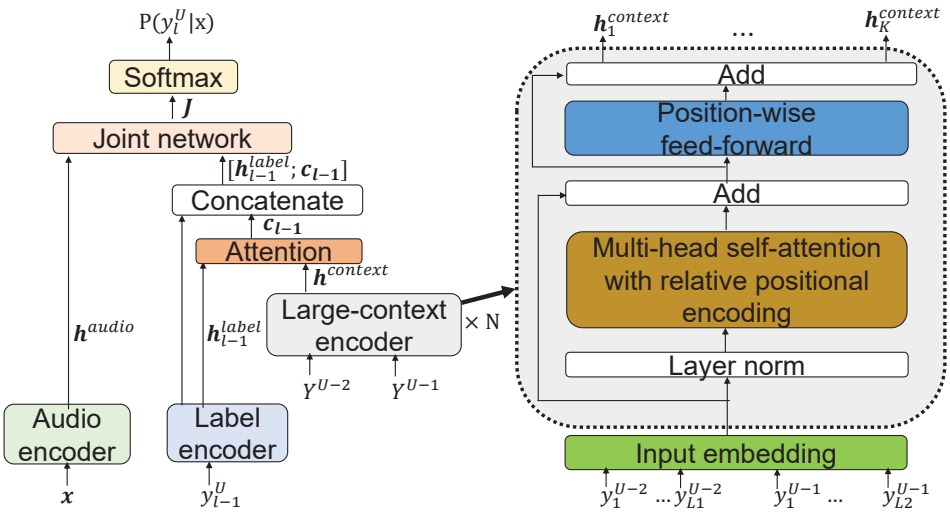


Fig. 2. Overview of large-context end-to-end ASR based on RNN–T.

The RNN–T loss function is written as

$$\lambda_{\mathrm{RNN-T}} = -\sum_i \log P(\boldsymbol{y}|\mathrm{x}), \tag{1}$$

$$P(\boldsymbol{y}|\mathrm{x}) = \sum_{\boldsymbol{J}\in\mathcal{Z}(\boldsymbol{y},T)} P(J|\mathrm{x}), \tag{2}$$

where $\mathcal{Z}(\boldsymbol{y}, T)$ is the set of all alignments of length $T$ for the token sequence.

### B. Transformer

The Transformer layer consists of multi-head self-attention and position-wise feed-forward layers. In the multi-head self-attention layer, self-attention is calculated as

$$\mathrm{SelfAttention}(Q,K,V) = \mathrm{Softmax}(\frac{QK^T}{\sqrt{d_k}})V, \tag{3}$$

where $Q \in \mathbb{R}^{T_q\times d_k}$, $K \in \mathbb{R}^{T_k\times d_k}$ and $V \in \mathbb{R}^{T_k\times d_v}$ are the query, key and value, respectively, and $d_k$ is a parameter. $\mathrm{Softmax}(\frac{QK^T}{\sqrt{d_k}})$ is called self-attention. Again, the Transformer layer has multiple heads:

$$\begin{aligned}&\mathrm{MultiHeadAttention}(Q,K,V)\\ &= \mathrm{Concat}(\mathrm{HEAD}_1,...,\mathrm{HEAD}_{\mathrm{head}})W^O,\end{aligned} \tag{4}$$

$$\mathrm{HEAD}_i = \mathrm{SelfAttention}(QW_i{}^Q, KW_i{}^k, QW_i{}^V), \tag{5}$$

where head is the number of heads and $W^O \in \mathbb{R}^{d_{in}\times d_{in}}$, $W_i^Q \in \mathbb{R}^{d_{in}\times d_k}$, $W_i^K \in \mathbb{R}^{d_{in}\times d_k}$ and $W_i^V \in \mathbb{R}^{d_{in}\times d_v}$ are model parameters.

## III. LARGE-CONTEXT END-TO-END ASR BASED ON RNN–T

### A. Architecture

In this paper, we propose large-context end-to-end ASR based on RNN–T. To utilize hypotheses generated in previous utterances as a large context, we introduce a large-context encoder for RNN–T. Fig. 2 shows an overview of the proposed method. In this figure, $\boldsymbol{h}^{\mathrm{context}}$ and $\boldsymbol{c}$ represent the hidden vector of the large-context encoder and the context vector, respectively. $K$ and $U$ represent the number of input tokens for the large-context encoder and the index of utterances, respectively.

The large-context encoder, which consists of Transformer, obtains concatenated hypotheses in the order of utterance start times and converts them to hidden vectors $(\boldsymbol{h}_1^{\mathrm{context}}, \boldsymbol{h}_2^{\mathrm{context}}, \cdots, \boldsymbol{h}_K^{\mathrm{context}})$. When the model recognizes the first utterance in the conversation, the large-context encoder obtains a special token, which represents that the hypothesis to refer to do not exist. When the large-context encoder obtains the hypothesis generated in the first utterance, this token is removed from the input for the large-context encoder. Attention is then calculated as

$$a_{p,l} = \mathrm{Softmax}(\boldsymbol{w}^T\tanh(U\boldsymbol{h}_l^{\mathrm{label}} + H\boldsymbol{h}_p^{\mathrm{context}} + \boldsymbol{b})), \tag{6}$$

where $l$ and $p$ represent the indexes of the hidden vectors of the label encoder and large-context encoder, respectively. $\boldsymbol{w}$, $U$, $H$ and $\boldsymbol{b}$ are model parameters. Also, context vector $\boldsymbol{c}$ is calculated using attention $a$ and the hidden vectors of the large-context encoder $\boldsymbol{h}^{\mathrm{context}}$ as

$$\boldsymbol{c}_l = \sum_{p=1}^{K} a_{p,l}\boldsymbol{h}_p^{\mathrm{context}}. \tag{7}$$

Then, the context vector is concatenated with the hidden vectors of the label encoder. The joint network outputs vector $J$ using the concatenated vector $[\boldsymbol{h}^{\mathrm{label}}; \boldsymbol{c}]$ and the hidden vectors of the audio encoder $\boldsymbol{h}^{\mathrm{audio}}$, and softmax outputs logits.

### B. Training

To train the proposed model, we calculate all possible context vectors. Given the hidden vectors of the label encoder $(\boldsymbol{h}_1^{\text{label}}, \boldsymbol{h}_2^{\text{label}} \cdots \boldsymbol{h}_L^{\text{label}})$ and the hidden vectors of the large-context encoder $(\boldsymbol{h}_1^{\text{context}}, \boldsymbol{h}_2^{\text{context}} \cdots \boldsymbol{h}_K^{\text{context}})$, possible context vectors $(\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots \boldsymbol{c}_L)$ can be calculated using Equations (6) and (7). Then, we concatenate the context vectors and the hidden vectors of the label encoder, and obtain $([\boldsymbol{h}_1^{\text{label}}; \boldsymbol{c}_1], [\boldsymbol{h}_2^{\text{label}}; \boldsymbol{c}_2], \cdots, [\boldsymbol{h}_L^{\text{label}}; \boldsymbol{c}_L])$. Using these concatenated vectors and the hidden vectors of the audio encoder, we can train the proposed model using RNN–T loss. During training, the large-context encoder obtains not hypotheses but reference transcripts. As the number of previous utterances when referring to reference transcripts, we randomly choose from the uniform distribution $[0, N_{\text{refer}}]$ for each sample, where $N_{\text{refer}}$ is a tunable parameter.

### C. Inference

For inference, the context vector is updated only when the model outputs a token except the blank token. Firstly, the hidden vector of the label encoder is uploaded. Second, attention is calculated, then the context vector is updated. Finally, the joint network obtains the two updated vectors and the hidden vector of the audio encoder, and softmax outputs logits. The input for the large-context encoder is updated using the best hypothesis yielded by a beam search.

## IV. EXPERIMENTS AND RESULTS

### A. Corpus and Setup

As the experimental dataset, we used real medical conversations consisting of 93.3 h of transcribed speech. The duration of one conversation was from 10 to 15 minutes and the duration of utterances was from 170 to 15000 ms. The number of utterances included in one medical conversation was 34 to 534. The speech spoken by the doctor and patient was recorded on separate channels at 16 kHz. We divided conversations into training and test datasets. TABLE I shows the details of each dataset. When training the model, we also used 518 h of speech data collected from conversations such as internal meetings because the size of the medical conversation dataset was small.

TABLE I
DATA SUBSETS.

| Subset | Speaker | Duration (h) | # of utterances | # of characters | # of conversations |
|---|---|---|---|---|---|
| Training | Doctor | 43.6 | 73059 | 845869 | 436 |
| | Patient | 48.0 | 77337 | 890103 | |
| Test | Doctor | 0.8 | 3808 | 177710 | 10 |
| | Patient | 0.9 | 4108 | 17080 | |

In all experiments, the ASR model output a character and a blank token. The total number of tokens was 3672. As input acoustic features, we used a 40-dimensional log Mel filter bank, computed with a 25 ms window and shifted every 10 ms. Every three frames were stacked and these features were downsampled to a 30 ms frame rate [14].

When the large-context encoder obtained hypotheses, special tokens <dr> and <pt> representing the doctor and patient were inserted in the beginning of their hypotheses, respectively. These tokens can be determined automatically because the speech of the doctor and patient was recorded on separate channels.

When training the model, gradient clipping was applied with a value of 5 to avoid an exploding gradient. Furthermore, we applied SpecAugment [15] and label smoothing [16] to improve robustness. All networks were implemented using PyTorch [17].

As the evaluation measure, we used CER and the 90-percentile real time factor (RT90). For CER, results are given as the relative character error rate reduction (CERR) [%]. For RT90, we evaluate inference speed by measuring decoding time over 534 utterances included in a conversation on a Intel Core® i7-6700 processors machine using 1 CPU per method to process an utterance at a time.

### B. Experimental Setup

For the network architecture, we used four unidirectional LSTM layers with 128 hidden nodes and a unidirectional LSTM layer with 128 hidden nodes for the audio encoder and label encoder, respectively. TABLE II shows the parameters of the large-context encoder. The joint network obtains 384-dimensional vectors from audio, label and large-context encoders, and outputs a 128-dimensional vector with Tanh activation. Finally, softmax outputs 3672-dimensional logits.

TABLE II
LARGE-CONTEXT ENCODER ARCHITECTURE.

| Parameter | Value |
|---|---|
| Number of layers | 4 |
| Number of heads | 4 |
| Head dimension | 32 |
| Number of hidden nodes | 128 |
| Position-wise feed-forward dimension | 256 |

To train the proposed model, we used the Transformer learning schedule [12]. We also used the Adam optimizer [18] and set $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-7}$. Also, we set $N_{\text{refer}} = 128$. For inference, we used a beam search with a beam size of 4.

In addition, we examined the effect of the number of utterances $N_{\text{hypothesis}}$ when referring to hypotheses on CER and RT90. For instance, the large-context encoder obtains only hypotheses generated in the last two utterances in the case of $N_{\text{hypothesis}} = 2$. This is worth exploring because the computational cost for outputting hidden vectors of the large-context encoder can be reduced by the limiting context. We remove the oldest hypothesis from the input for the large-context encoder if the number of hypotheses that must be kept reaches $N_{\text{hypothesis}}$.

As the baseline system, we used the original RNN–T. For the audio encoder and label encoder, we set the parameters to

TABLE III
RESULTS OF CERR AND RT90.

| Method | $N_{\text{hypothesis}}$ | Doctor | Patient | Overall | RT90 |
|---|---|---|---|---|---|
| RNN–T (baseline) [3] | - | 0.0 | 0.0 | 0.0 | 0.06 |
| + large-context encoder | 10 | 5.1 | 1.1 | 3.6 | 0.06 |
| + large-context encoder | 20 | 8.3 | 2.7 | 5.9 | 0.06 |
| + large-context encoder | 128 | 14.0 | 5.0 | 9.8 | 0.28 |
| + large-context encoder | $\infty$ | **17.2** | **6.0** | **11.9** | 0.89 |

TABLE IV
EFFECT OF THE RECOGNITION ERROR ($N_{\text{hypothesis}} = \infty$).

| Input for large-context encoder | Doctor | Patient |
|---|---|---|
| Hypothesis | 0.0 | 0.0 |
| Reference transcripts | 0.0 | 0.9 |

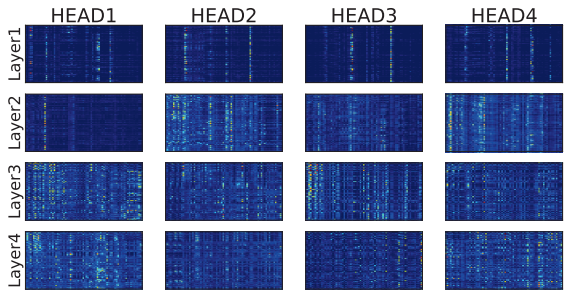


Fig. 3. Example of self-attention.

be the same as those in the proposed method. The total parameter sizes of the proposed model and the original RNN–T are 2.6 (M) and 1.6 (M), respectively.

### C. Results

The results are summarized in TABLE III. For CER, the proposed model outperforms the original RNN–T for the doctor and patient for all values of $N_{\text{hypothesis}}$. In the case of $N_{\text{hypothesis}} = \infty$, we obtained 17.2% and 6.0% relative reductions in CER. Regarding the effect of $N_{\text{hypothesis}}$, the larger the value of $N_{\text{hypothesis}}$, the more CER is reduced. However, increasing $N_{\text{hypothesis}}$ also leads to an increase in RT90. Therefore, there is a trade-off relation between $N_{\text{hypothesis}}$ and CER.

In addition, we examined the effect of the recognition error in the proposed method. The results are summarized in TABLE IV.

For the speech of the patient, CER was slightly reduced when reference transcripts were used, but CER for the speech of the doctor was not reduced. Therefore, we conclude that effect of the ASR error is small for the proposed model.

### D. Analysis of Self-Attention in Large-Context Encoder

We analyzed self-attention in the large-context encoder. Fig. 3 shows an example of self-attention during inference. The number of concatenated hypotheses was 64. From this figure, we can see a column-based pattern. We can see that specific inputs play a special role regardless of the position in inputs. These results suggest that the large-context encoder can capture long-range context.

## V. CONCLUSION

In this paper, we proposed large-context end-to-end ASR based on RNN–T for conversational ASR. To utilize hypotheses generated in previous utterances as a large context, we introduced a large-context encoder, which consists of Transformer for RNN–T. The large-context encoder obtains concatenated hypotheses and converts them to hidden vectors. The hidden vectors are summarized as the available context based on an attention mechanism. During training, the large-context encoder and RNN–T components are jointly optimized using RNN–T loss. In our experiments, we compared the proposed method with the original RNN–T using medical conversations. As a result of the experiment, we obtained 17.2% and 6.0% relative reductions in CER compared with those of the original RNN–T on utterances spoken by the doctor and patient, respectively.

## ACKNOWLEDGMENT

We would like to thank Rei Oguro for providing computational resources.

## REFERENCES

[1] W. Chan, N. Jaitly, Q. Le and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. ICASSP*, 2016.
[2] L. Dong, S. Xu and B. Xu "Speech-Transformer: A no-recurrence sequence-to-sequence model for speech recognition," in *Proc. ICASSP*, 2018.
[3] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.
[4] C. F. Yeh, J. Mahadeokar, K. Kalgaonkar, Y. Wang, D. Le, M. Jain, K. Schubert, C. Fuegen and M. L. Seltzer, "Transformer–Transducer: End-to-end speech recognition with self-attention," in *Proc. INTERSPEECH*, 2020.
[5] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo and S. Kumar, "Transformer Transducer: A streamable speech recognition model with Transformer encoders and RNN–T loss," in *Proc. INTERSPEECH*, 2020.
[6] A. Graves, S. Fernández, F. Gomez and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006.
[7] T. Hori, N. Moritz, C. Hori and J. L. Roux, "Transformer-based long-context end-to-end speech recognition," in *Proc. INTERSPEECH*, 2020.
[8] R. Masumura, T, Tanaka, T. Moriya, Y. Shinohara, T. Oba and T. Aono, "Large context end-to-end automatic speech recognition via extension of hierarchical recurrent encoder-decoder models," in *Proc. ICASSP*, 2019.
[9] R. Masumura, N. Makishima, M. Ihori, A. Takashima, T. Tanaka and S. Orihashi, "Hierarchical Transformer-based large-context end-to-end ASR with large-context knowledge distillation," in *Proc. ICASSP*, 2021.
[10] S. Kim and F. Metze, "Dialog-context aware end-to-end speech recognition," in *Proc. SLT*, 2018.

[11] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. C. Sim, T. Bagby, S. Chang, K. Rao and A. Gruenstein, "Streaming end-to-end speech recognition for mobile devices," in *Proc. ICASSP*, 2019.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention is all you need," in *Proc. NeurIPS*, 2017.

[13] D. Bahdanau, K. Cho and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. ICLR*, 2015.

[14] H. Sak, A. Senior, K. Rao and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," *arXiv preprint arXiv:1507.06947*, 2015.

[15] D. S. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E. D. Cubuk and Q. V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Proc. INTERSPEECH*, 2019.

[16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. CVPR*, 2016.

[17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. NeurIPS*, 2019.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.