Memory-Efficient Training of RNN-Transducer with Sampled Softmax

Jaesong Lee1∗, Lukas Lee1∗, Shinji Watanabe2

1Naver Corporation
2Carnegie Mellon University
jaesong.lee@navercorp.com, lukas.lee@navercorp.com, shinjiw@ieee.org

Abstract

RNN-Transducer has been one of promising architectures for end-to-end automatic speech recognition. Although RNN-Transducer has many advantages including its strong accuracy and streaming-friendly property, its high memory consumption during training has been a critical problem for development. In this work, we propose to apply sampled softmax to RNN-Transducer, which requires only a small subset of vocabulary during training thus saves its memory consumption. We further extend sampled softmax to optimize memory consumption for a minibatch, and employ distributions of auxiliary CTC losses for sampling vocabulary to improve model accuracy. We present experimental results on LibriSpeech, AISHELL-1, and CSJ-APS, where sampled softmax greatly reduces memory consumption and still maintains the accuracy of the baseline model.

Index Terms: end-to-end speech recognition, RNN-Transducer, sampled softmax, auxiliary CTC loss

1. Introduction

End-to-end automatic speech recognition (ASR) has been considered significant over the past few years [1–11]. There have been large improvements in the performance of end-to-end ASR with advances in deep learning approaches and frameworks. Furthermore, training and inference pipelines of end-to-end ASR are relatively simple compared to those of traditional hidden Markov model (HMM) based methods. As a result, end-to-end ASR has been a promising direction from research to production [12, 13].

RNN-Transducer [1] is one of end-to-end ASR architectures that has been in the spotlight [12–17]. It consists of an encoder network, a prediction network and a joint network in a single model. Acoustic features are processed in the encoder network, and produces high-level representations \( x_t \) for the prediction network and a target sequence. The architecture consists of three components: encoder, prediction, and joint network. The encoder network accepts an input speech sequence \( x \), which is used to predict the next output label, to high-level representation, \( T \) is the number of frames. The prediction network converts previously generated output labels to high-level representation, which is used to predict the next output label. For an output label sequence of length \( u \geq 0 \), the prediction network produces \( h_{u}^{\text{pre}} \in \mathbb{R}^{H} \). The joint network fuses encoder represen-
Huge memory consumption of training RNN-Transducer has been a serious issue, and various methods have been proposed to overcome the issue.

One possible way is to have smaller $V$ by adjusting tokenization level. For alphabet-based languages, character-based tokenization gives smaller vocabulary than subword-level tokenization [32]. However, [17] concludes a subword-level model outperforms a character-based model, implying large vocabulary is inevitable for better accuracy. On the other hand, for Chinese and Japanese, character-level tokenization still gives large vocabulary, due to variety of Chinese characters. For Mandarin, [23] uses syllable-based tokenization to get small $V$. It requires language-dependent engineering for tokenization and an additional network for converting syllables to target characters.

It is also possible to improve network layers of RNN-Transducer for more efficient training. [21] and [22] propose encoder networks with more aggressive downsampling, reducing the number of frames ($T$). [24] proposes new architecture for a joint network using bilinear layer, and achieves better accuracy while using similar amount of memory.

Improving efficiency of training has been also investigated. [16] improves memory usage of RNN-Transducer by reducing padding for the minibatch setting. [33] uses mixed-precision training for memory reduction at the cost of precision loss.

Pre-training of RNN-Transducer networks using other objectives have been investigated, including CTC [2,12], language models [12,16], and cross-entropy [25]. [34] restricts the set of valid alignments via an external alignment, reducing the size of logit tensor. In a teacher-student distillation setting, [26] proposes to simplify distillation loss by merging non-target labels.

At Section 3, we will present a new way to reduce memory consumption by replacing $V$ with much smaller sets at the training time. The proposed method does not need any modification of tokenization, networks, or training pipeline. Also, most of previous works above could be combined with our proposed method.

### 2.2. Auxiliary CTC losses

Auxiliary loss functions have been proposed for regularization of RNN-Transducer models [28, 29, 35] and CTC models [30, 31, 36]. In this work, we consider CTC-based regularization methods for the encoder network, as they are computationally inexpensive and will be also used for sampled softmax of Section 3.

Joint CTC [8, 28, 29] uses encoder output $h_{\text{enc}}$ to compute an auxiliary CTC loss $L_{\text{CTC}} := -\log P_{\text{CTC}}(y|h_{\text{enc}})$.

Intermediate CTC [30] uses intermediate output $h_{\text{inter}}$ from a middle layer of an encoder network to compute an additional CTC loss $L_{\text{interCTC}} := -\log P_{\text{interCTC}}(y|h_{\text{inter}}, h_{\text{enc}})$. It is similar to the auxiliary loss of [35] which uses $h_{\text{pos}}$ to compute an auxiliary RNN-Transducer loss. However, the auxiliary loss requires its own logit tensor, leading to significant memory overhead. From preliminary experiments, we found similar improvement of two intermediate losses and decided to use intermediate CTC to save memory.

Self-conditioned CTC (SC-CTC) [31] extends intermediate CTC by conditioning the encoder using the intermediate distribution $P_{\text{interCTC}}$. The distribution is combined to the intermediate output $h_{\text{inter}}$ then fed to the next layer of the encoder.

### 3. Sampled softmax

As seen in Section 2, RNN-Transducer requires a significant amount of memory for training. In Eq. (1), the denominator of $p_{t,u,v}$ contains $s_{t,u,v}$ for all $v \in V$, which leads total memory complexity of $O(T \cdot |U| \cdot |V|)$ for computing the objective $L_{\text{transducer}}$ in Eq. (3).

To overcome the issue, we propose to apply sampled softmax [27], originally developed for neural machine translation, to RNN-Transducer. Sampled softmax approximates the denominator of softmax operation by choosing only a certain subset.

Let $V_{\text{pos}} = \{\langle \text{blank} \rangle, y_1, \ldots, y_V\}$ a “positive set”, which consists of labels used in alignments of Eq. (2). Let $V_{\text{neg}} \subset V \setminus V_{\text{pos}}$ a “negative set”, which is a subset sampled from some probability distributions. The softmax in Eq. (1) and loss function in Eq. (3) are modified below so that they only depend on $V_{\text{sampled}} := V_{\text{pos}} \cup V_{\text{neg}}$.

For sampled softmax, a probability $p_{t,u,v}^{\text{sampled}}$ is defined for $v \in V_{\text{sampled}}$:

$$
p_{t,u,v}^{\text{sampled}} := \frac{\exp(s_{t,u,v})}{\sum_{v' \in V_{\text{sampled}}} \exp(s_{t,u,v'})}.
$$

Note that the denominator only considers the subset $V_{\text{sampled}}$ which makes $p_{t,u,v}^{\text{sampled}}$ different from $p_{t,u,v}$. Then the objec-
Sampled softmax requires a probability distribution over vocabulary conditioned on the training example. Especially, it is possible to use a sampling distribution conditioned on the training example. The simplest choice is an uniform distribution: all labels in $\mathcal{V} \setminus \mathcal{V}^{pos}$ have equal probabilities. However, if a sampled label is irrelevant to the training example, its probability may be already low and it may not contribute much to model accuracy.

Ideally, if a model predicts a wrong label, it should be corrected during learning. So, it would be beneficial to sample such high-probability labels. However, determining label probabilities would require the computation of the logit $s_{t,u,v}$ for all $v \in \mathcal{V}$, defeating the purpose of sampled softmax.

To this end, we propose to use the token posterior distribution $P_{\text{CTC}}$ of a joint CTC loss for sampling $\mathcal{V}^{neg}$. As joint CTC and RNN-Transducer share same encoder network, the joint CTC distribution would be similar to the RNN-Transducer distribution and it would be a good approximation for the RNN-Transducer model. Similarly, we may also use the distribution $P_{\text{interCTC}}$ of Intermediate CTC or Self-conditioned CTC.

$P_{\text{CTC}}$ consists of $T$ independent distributions over $\mathcal{V}$, and it needs to be transformed into a single distribution over $\mathcal{V} \setminus \mathcal{V}^{pos}$. We simply average the distribution over $T$ frames and assign zero probability to $\mathcal{V}^{pos}$ to produce the desired distribution.

### 3.3. CTC-constrained decoding

When the negative set $\mathcal{V}^{neg}$ is drawn from joint CTC distribution as described in Section 3.2, a label may be rarely sampled during training if its probability is very low in joint CTC. If the label has a high probability from a RNN-Transducer but a low probability from a joint CTC, it may not be correctly learned and may be erroneously emitted during decoding.

To prevent such issue, we propose to constrain decoder to emit labels only in a vocabulary subset, which is determined from the distribution of the joint CTC. As a simple heuristic, we average $P_{\text{CTC}}$ over frames to get a single distribution over $\mathcal{V}$, and take top-$K$ labels based on their probabilities. This can be applied to both greedy decoding and beam-search decoding.

The proposed method can be viewed as an approximation of joint decoding of CTC and RNN-Transducer. While the implementation of joint decoding requires complex handling of blank labels of both models [28], the proposed method can be easily applied to ordinary decoding algorithms of RNN-Transducer.

### 4. Experiments

We use three corpora: LibriSpeech [37], AISHELL-1 [38], and Corpus of Spontaneous Japanese (CSJ) [39]. LibriSpeech consists of read English speech; we use the full set (960h hours) for training. AISHELL-1 consists of Mandarin speech; we use the full set (170 hours) for training. CSJ consists of various Japanese speech including academic presentations and public speech; we use the 271-hour subset of academic presentation speech (CSJ-APS) for training, following [40].

For LibriSpeech, SentencePiece [32] tokenization is used. SentencePiece allows user to determine vocabulary size, affecting a level of tokenization. Small vocabulary size would be memory-efficient for RNN-Transducer, but its corresponding target sentence becomes long and may affect accuracy of the model. To compare the effect of vocabulary size, we use various vocabulary sizes for SentencePiece from 500 to 2000. We report word error rates (WERs) for the corpus.

For AISHELL-1 and CSJ-APS, character-level tokenization is used. Since their transcriptions contain various Chinese and Kanji characters, their vocabulary sizes are inevitably large: 4231 on AISHELL and 2753 on CSJ-APS. We report character error rates (CERs) for the corpusa.
We propose to apply sampled softmax to RNN-Transducer for memory-efficient training. During training, a subset of vocabulary is sampled for each iteration, which leads to great memory reduction. With large batch size, accuracy of the model is improved and the CERs are comparable to the state-of-the-art result in [40]. This illustrates the importance of memory reduction for training RNN-Transducer models.

5. Conclusions

We propose two extensions of sampled softmax: the example-wise sampling strategy for efficient implementation for minibatch setting, and the employment of joint CTC and Self-conditioned CTC for a sampling distribution of the sub-set. We experimentally show that sampled softmax gives huge memory reduction while achieving the accuracy of the baseline model.
6. References

[1] A. Graves, “Sequence transduction with recurrent neural networks,” 2012.
[2] A. Graves et al., “Speech recognition with deep recurrent neural networks,” in Proc. ICASSP, 2013.
[3] A. Hannun et al., “Deep speech: Scaling up end-to-end speech recognition,” 2014.
[4] J. K. Chorowski et al., “Attention-based models for speech recognition,” in Proc. NeurIPS, 2015.
[5] Y. Miao et al., “ESEN: End-to-end speech recognition using deep rnn models and WFST-based decoding,” in Proc. ASRU, 2015.
[6] D. Amodei et al., “Deep speech 2: End-to-end speech recognition in english and mandarin,” in Proc. ICML, 2016.
[7] W. Chan et al., “Listen, attend and spell: A neural network for large vocabulary conversational speech recognition,” in Proc. ICASSP, 2016.
[8] S. Kim et al., “Joint CTC-attention based end-to-end speech recognition using multi-task learning,” in Proc. ICASSP, 2017.
[9] C.-C. Chiu et al., “State-of-the-art speech recognition with sequence-to-sequence models,” in Proc. ICASSP, 2018.
[10] C.-C. Chiu and C. Raffel, “Monotonic chunkwise attention,” in Proc. ICML, 2018.
[11] A. Baevski et al., “wav2vec 2.0: A framework for self-supervised learning of speech representations,” in Proc. NeurIPS, 2020.
[12] K. Rao et al., “Exploring architectures, data and units for streaming end-to-end speech recognition with RNN-transducer,” in Proc. ASRU, 2017.
[13] J. Guo et al., “Efficient minimum word error rate training of RNN-transducer for end-to-end speech recognition,” in Proc. Interspeech, 2020.
[14] A. Gulati et al., “Conformer: Convolution-augmented transformer for speech recognition,” in Proc. Interspeech, 2020.
[15] E. Battenberg et al., “Exploring neural transducers for end-to-end speech recognition,” in Proc. ASRU, 2017.
[16] J. Li et al., “Improving RNN transducer modeling for end-to-end speech recognition,” in Proc. ASRU, 2019.
[17] Y. He et al., “Streaming end-to-end speech recognition for mobile devices,” in Proc. ICASSP, 2019.
[18] A. Graves et al., “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in Proc. ICML, 2006.
[19] C.-C. Chiu et al., “A comparison of end-to-end models for long-form speech recognition,” in Proc. ASRU, 2019.
[20] J. Kim et al., “A comparison of streaming models and data augmentation methods for robust speech recognition,” in Proc. ASRU, 2021.
[21] S. Wang et al., “Exploring RNN-transducer for Chinese speech recognition,” in Proc. APSIPA, 2019.
[22] M. Burchi and V. Vielzeuf, “Efficient conformer: Progressive downsampling and grouped attention for automatic speech recognition,” in Proc. ASRU, 2021.
[23] X. Wang et al., “Cascade RNN-transducer: Syllable based streaming on-device Mandarin speech recognition with a syllable-to-character converter,” in Proc. SLT, 2021.
[24] G. Saon et al., “Advancing RNN transducer technology for speech recognition,” in Proc. ICASSP, 2021.
[25] H. Hu et al., “Exploring pre-training with alignments for RNN transducer based end-to-end speech recognition,” in Proc. ICASSP, 2020.
[26] S. Panchapagesan et al., “Efficient knowledge distillation for RNN-transducer models,” in Proc. ICASSP, 2021.

[27] S. Jean et al., “On using very large target vocabulary for neural machine translation,” in Proc. ACL-UCLNLP, 2015.
[28] J.-J. Jeon and E. Kim, “Multitask learning and joint optimization for transformer-RNN-transducer speech recognition,” in Proc. ICASSP, 2021.
[29] F. Boyet et al., “A study of transducer based end-to-end ASR with ESPNet: Architecture, auxiliary loss and decoding strategies,” in Proc. ASRU, 2021.
[30] J. Lee and S. Watanabe, “Intermediate loss regularization for CTC-based speech recognition,” in Proc. ICASSP, 2021.
[31] J. Nozaki and T. Komatsu, “Relaxing the conditional independence assumption of CTC-based ASR by conditioning on intermediate predictions,” in Proc. Interspeech, 2021.
[32] T. Kudo and J. Richardson, “SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing,” in Proc. EMNLP: System Demonstrations, 2018.
[33] X. Zhang et al., “Benchmarking lf-nmi, ctc and nn-t criteria for streaming ast,” in Proc. SLT, 2021.
[34] J. Mahadeokar et al., “Alignment restricted streaming recurrent neural network transducer,” in Proc. SLT, 2021.
[35] C. Liu et al., “Improving RNN transducer based ASR with auxiliary tasks,” in Proc. SLT, 2020.
[36] A. Tjandra et al., “Deja-vu: Double feature presentation and iterated loss in deep transformer networks,” in Proc. ICASSP, 2020.
[37] V. Panayotov et al., “Librispeech: An ASR corpus based on public domain audio books,” in Proc. ICASSP, 2015.
[38] H. Bu et al., “AISHELL-1: An open-source mandarin speech corpus and a speech recognition baseline,” in Proc. O-COCOSDA, 2017.
[39] K. Maekawa et al., “Spontaneous speech corpus of japanese,” in Proc. LREC, 2000.
[40] Y. Higuchi et al., “A comparative study on non-autoregressive modelings for speech-to-text generation,” in Proc. ASRU, 2021.
[41] D. S. Park et al., “SpecAugment: A simple data augmentation method for automatic speech recognition,” in Proc. Interspeech, 2019.
[42] S. Watanabe et al., “ESPNet: End-to-end speech processing toolkit,” in Proc. Interspeech, 2018.
[43] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural computation, 1997.