Interspeech 2022
18-22 September 2022, Incheon, Korea

Memory-Efficient Training of RNN-Transducer with Sampled Softmax

Jaesong Lee[1*], *Lukas Lee*[1*], *Shinji Watanabe*[2]

[1]Naver Corporation
[2]Carnegie 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 text history is considered in the prediction network. RNN-Transducer has several advantages for production. Its auto-regressive characteristic makes it easier to deal with language model (LM) information of target symbols, compared to Connectionist Temporal Classification (CTC) [18]. Also, RNN-Transducer is streaming-friendly and often outperforms conventional encoder-decoder or monotonic attention architectures for long-form speech [19, 20].

However, the joint network and the loss function of RNN-Transducer consume significantly large memory during training, thus, the development of RNN-Transducer can be computationally expensive compared to other end-to-end architectures. This prevents developing RNN-Transducer without large GPU clusters. Various ways are developed to mitigate the problem [2, 12, 16, 21–26], but most of them require language-specific or network-specific engineering efforts.

In this work, we propose a new way to train RNN-Transducer without large memory consumption. We apply sampled softmax [27], which is originally developed for neural machine translation, to RNN-Transducer. For computing the RNN-Transducer loss, only a small subset of vocabulary is sampled and used for the softmax function, so the memory bottleneck can be reduced by adjusting the size of the subset during training.

To efficiently train RNN-Transducer models, we extend sampled softmax in two ways. First, we break the common assumption of sampled softmax that a sampled subset is shared within minibatch. Instead, we propose to sample the subset independently per each training example, which gives a significant reduction of the subset size and thus saves more memory.

Second, we propose to use a token posterior distribution of joint CTC loss [28, 29] for sampling the subset. We show that a choice of a sampling distribution affects the model accuracy, and that a joint CTC distribution provides good accuracy while requiring only marginal computational costs. We also apply Intermediate CTC [30] and Self-conditioned CTC [31] to RNN-Transducer, which are both used for regularization of a model and for the sampling distribution of sampled softmax. With the two extensions of sampled softmax, the RNN-Transducer model can be trained with much less memory consumption and reaches the accuracy of the original model.

In summary, we show the following contributions:

- We propose to apply sampled softmax to RNN-Transducer for memory-efficient training.
- We extend sampled softmax with the example-wise sampling strategy and with use of a token distribution of joint CTC branch for a sampling distribution.
- We extend Intermediate CTC and Self-conditioned CTC to RNN-Transducer, and also employ them for sampling distributions of sampled softmax.
- We experimentally show that sampled softmax greatly reduces memory consumption while reaching accuracy of the baseline model.

2. RNN-Transducer

RNN-Transducer [1] is an architecture for automatic speech recognition (ASR). It predicts target labels augmented with special symbol $\langle \texttt{blank} \rangle$, which represents an alignment between a network prediction and a target sequence. The architecture consists of three components: encoder, prediction, and joint networks.

The encoder network accepts an input speech sequence $\mathbf{x}$ and produces high-level representations $(\mathbf{h}_1^{\mathsf{enc}}, \cdots, \mathbf{h}_T^{\mathsf{enc}})$, where $\mathbf{h}_t^{\mathsf{enc}} \in \mathbb{R}^H$ is H-dimensional vector of t-th frame and 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 $\mathbf{h}_u^{\mathsf{pre}} \in \mathbb{R}^H$. The joint network fuses encoder represen-

* Two authors contributed equally.

10.21437/Interspeech.2022-787

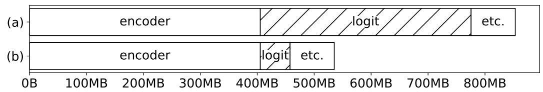

Figure 1: *Memory consumption of (a) an RNN-Transducer model and (b) the same model with sampled softmax.*

tation $\mathbf{h}_t^{\text{enc}}$ and prediction representation $\mathbf{h}_u^{\text{pre}}$ into label logits $\mathbf{s}_{t,u} = (s_{t,u,v} : v \in \mathcal{V})$, where $s_{t,u,v} \in \mathbb{R}$ is an unnormalized logit of a target label v and $\mathcal{V}$ is a vocabulary set augmented with $\langle\texttt{blank}\rangle$.

By applying the softmax function to the logits, a probability distribution over $\mathcal{V}$ is induced:

$$p_{t,u,v} = \mathsf{softmax}(\mathbf{s}_{t,u})_v = \frac{\exp(s_{t,u,v})}{\sum_{v' \in \mathcal{V}} \exp(s_{t,u,v'})}. \quad (1)$$

During decoding, RNN-Transducer maintains a state at (t, u) and generates a sentence based on the state. The model starts with initial state at $(1, 0)$, and for each state, the model draws a label v with probability $p_{t,u,v}$. If $\langle\texttt{blank}\rangle$ is drawn, the state is updated to $(t+1, u)$, and generation is finished when $t > T$. Otherwise, the label is appended to the output and the state is updated to $(t, u+1)$.

During training, a target label sequence $\mathbf{y} = (y_1, \cdots, y_U)$, $y_u \in \mathcal{V} \setminus \{\langle\texttt{blank}\rangle\}$, of length U is given. RNN-Transducer considers all possible "alignment", which is a state transition path compatible to $\mathbf{y}$, i.e., the path generates the target $\mathbf{y}$. Denoting $\mathcal{B}^{-1}(\mathbf{y})$ the set of alignment compatible to $\mathbf{y}$, the probability of the target sequence is:

$$P(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{z} \in \mathcal{B}^{-1}(\mathbf{y})} \prod_{(t,u,y) \in \mathbf{z}} p_{t,u,y}. \quad (2)$$

Then the loss function is negative log-probability of the target sequence:

$$\mathcal{L}_{\mathsf{transducer}} := -\log P(\mathbf{y}|\mathbf{x}). \quad (3)$$

Note that $\mathcal{L}_{\mathsf{transducer}}$ contains $p_{t,u,y}$ for all t and u. Also, due to definition of softmax in Eq. (1), $p_{t,u,y}$ depends on $s_{t,u,v}$ for all $v \in \mathcal{V}$. Thus, to compute $\mathcal{L}_{\mathsf{transducer}}$ and its gradient, $O(T \cdot U \cdot |\mathcal{V}|)$ memory must be computed and stored solely for the logit, resulting in very large memory allocation of the logit tensor. Figure 1 (a) shows estimated memory usage of an RNN-Transducer model, where the whole encoder network and the logit tensor take nearly same memory space.

2.1. Previous works on memory consumption issue

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 $\mathcal{V}$ 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 $\mathcal{V}$. 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], 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 $\mathcal{V}$ 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 $\mathbf{h}_t^{\text{enc}}$ to compute an auxiliary CTC loss $\mathcal{L}_{\mathsf{CTC}} := -\log P_{\mathsf{CTC}}(\mathbf{y}|\mathbf{h}_1^{\text{enc}}, \cdots, \mathbf{h}_T^{\text{enc}})$.

Intermediate CTC [30] uses intermediate output $\mathbf{h}_t^{\text{inter}}$ from a middle layer of an encoder network to compute an additional CTC loss $\mathcal{L}_{\mathsf{InterCTC}} := -\log P_{\mathsf{InterCTC}}(\mathbf{y}|\mathbf{h}_1^{\text{inter}}, \cdots, \mathbf{h}_T^{\text{inter}})$. It is similar to the auxiliary loss of [35] which uses $\mathbf{h}_t^{\text{inter}}$ 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_{InterCTC}. The distribution is combined to the intermediate output $\mathbf{h}_t^{\text{enc}}$ 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 $v' \in \mathcal{V}$, which leads total memory complexity of $O(T \cdot U \cdot |\mathcal{V}|)$ for computing the objective $\mathcal{L}_{\mathsf{transducer}}$ 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 $\mathcal{V}^{\mathsf{pos}} = \{\langle\texttt{blank}\rangle, y_1, \cdots, y_U\}$ a "positive set", which consists of labels used in alignments of Eq. (2). Let $\mathcal{V}^{\mathsf{neg}} \subset \mathcal{V} \setminus \mathcal{V}^{\mathsf{pos}}$ 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 $\mathcal{V}^{\mathsf{sampled}} := \mathcal{V}^{\mathsf{pos}} \bigcup \mathcal{V}^{\mathsf{neg}}$.

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

$$p_{t,u,v}^{\mathsf{sampled}} := \frac{\exp(s_{t,u,v})}{\sum_{v' \in \mathcal{V}^{\mathsf{sampled}}} \exp(s_{t,u,v'})}. \quad (4)$$

Note that the denominator only considers the subset $\mathcal{V}^{\mathsf{sampled}}$, which makes $p_{t,u,v}^{\mathsf{sampled}}$ different from $p_{t,u,v}$. Then the objec-

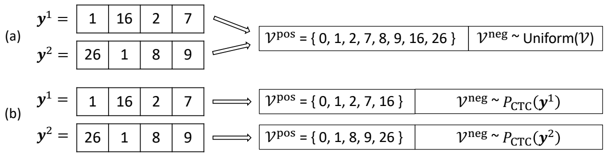

Figure 2: *Comparison of (a) batched sampling and (b) example-wise sampling. Example-wise sampling has smaller $\mathcal{V}^{\text{pos}}$ and can use different sampling distributions for $\mathcal{V}^{\text{neg}}$.*

tive $\mathcal{L}_{\text{transducer}}$ in Eq. (3) is modified by replacing $p_{t,u,y}$ with $p^{\text{sampled}}_{t,u,y}$.

Sampled softmax approximates the original softmax by the subset $\mathcal{V}^{\text{sampled}}$, and the choice of $\mathcal{V}^{\text{neg}}$ affects the quality of approximation. For example, if a label is rarely contained in $\mathcal{V}^{\text{sampled}}$ during training, its probability may not be correctly learned and it may be wrongly emitted at decoding. We sample $\mathcal{V}^{\text{neg}}$ independently for each minibatch; sampling strategies and distributions are discussed at Section 3.1 and Section 3.2.

With sampled softmax, it is only required to compute $s_{t,u,v'}$ for $v' \in \mathcal{V}^{\text{sampled}}$, not the full set $\mathcal{V}$. Thus, the memory size of a logit tensor is reduced to $O(T \cdot U \cdot |\mathcal{V}^{\text{sampled}}|)$.

Figure 1 shows memory reduction of RNN-Transducer due to sampled softmax. While the original model consumes nearly same memory for the encoder network and the logit tensor, the memory of the logit tensor is reduced using sampled softmax if $|\mathcal{V}^{\text{sampled}}|$ is sufficiently small. Sampled softmax also allows training of RNN-Transducer with larger batch size, which may be beneficial for certain networks like Conformer [14] due to large batch dependency of Batch Normalization.

3.1. Interaction with minibatch

For GPU-based training, multiple training examples are packed in one minibatch and computed in parallel. This affects implementation strategy of sampled softmax.

In common implementation of sampled softmax[1], $\mathcal{V}^{\text{sampled}}$ is prepared for each minibatch, i.e., $\mathcal{V}^{\text{pos}}$ consists of all target labels in the minibatch and $\mathcal{V}^{\text{neg}}$ is shared among training examples in the minibatch, as shown in Figure 2 (a). We call this strategy "batched sampling".

Batched sampling has a major drawback that $|\mathcal{V}^{\text{pos}}|$ grows as batch size increases. It can be significantly large (e.g., $|\mathcal{V}^{\text{pos}}| > 500$) for large batch size, and $|\mathcal{V}^{\text{sampled}}|$ cannot be reduced below the value, affecting memory consumption. Also, the positive labels of other examples may not be helpful for training, leading to inefficiency of training.

Alternatively, we propose "example-wise sampling", which is to sample $\mathcal{V}^{\text{sampled}}$ independently for each training example, as shown in Figure 2 (b). This gives much smaller positive sets, and it is possible to assign a different negative set for each training example. Especially, it is possible to use a sampling distribution conditioned on the training example.

3.2. Sampling distributions

Sampled softmax requires a probability distribution over vocabulary for sampling $\mathcal{V}^{\text{neg}}$. The simplest choice is an uniform distribution: all labels in $\mathcal{V} \setminus \mathcal{V}^{\text{pos}}$ 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.

[1]For example, this implementation strategy is used by TensorFlow's `sampled_softmax_loss`.

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_{CTC} of a joint CTC loss for sampling $\mathcal{V}^{\text{neg}}$. 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_{InterCTC} of Intermediate CTC or Self-conditioned CTC.

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

3.3. CTC-constrained decoding

When the negative set $\mathcal{V}^{\text{neg}}$ is drawn from joint CTC distribution as described in at 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 from the joint CTC. As a simple heuristic, we average P_{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 corpora.

4.1. Results on LibriSpeech

For LibriSpeech, we follow the architecture of Conformer-M [14] for an encoder network. SpecAugment [41] and speed

Table 1: *Word error rates (WERs) on LibriSpeech. No external language model is used.*

$\|\mathcal{V}\|$	$\|\mathcal{V}^{\text{sampled}}\|$	AuxLoss	dev clean / other	test clean / other
Baseline				
2000	-		2.69 / 6.91	2.91 / 6.93
2000	-	+InterCTC	2.63 / 6.60	2.84 / 6.76
2000	-	+SC-CTC	**2.51** / 6.57	**2.74** / **6.52**
600	-		2.70 / 7.16	2.80 / 7.17
600	-	+SC-CTC	**2.52** / 6.59	2.81 / 6.61
500	-		2.66 / 7.12	2.96 / 7.25
500	-	+SC-CTC	2.65 / 6.67	2.82 / 6.88
Batched sampling from uniform				
2000	600		2.92 / 7.23	3.13 / 7.30
2000	600	+SC-CTC	2.66 / 6.80	2.97 / 6.88
Example-wise sampling from uniform				
2000	600	+SC-CTC	2.72 / 6.97	2.95 / 6.99
500	250	+SC-CTC	2.70 / 6.81	3.01 / 6.93
Example-wise sampling from SC-CTC				
2000	600	+SC-CTC	2.58 / **6.54**	2.82 / 6.56
2000	300	+SC-CTC	2.58 / **6.53**	**2.75** / 6.63
500	250	+SC-CTC	2.60 / 6.66	2.84 / 6.72
500	160	+SC-CTC	2.59 / 6.70	2.78 / 6.73

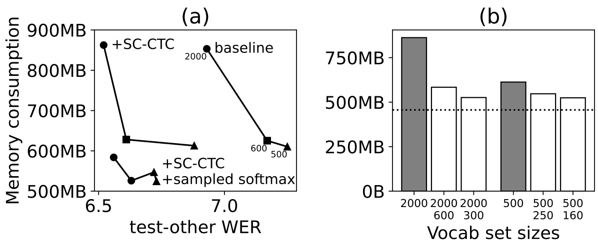

Figure 3: *(a) Memory - test-other WER plot of LibriSpeech for various vocabulary and sample set sizes. Markers indicate vocabulary size ($|\mathcal{V}|$). (b) Memory consumption per training example for various sample set sizes. Gray bars are baseline models without sampled softmax. Dotted line is memory consumption of encoder and predictor networks.*

perturbation are applied for regularization. Beam search [1] is conducted for decoding and a beam size of 10 is used. No external language model is applied. The experimental result is reported in Table 1.

For the choice of a sampling distribution of sampled softmax, we found using the distribution of Self-conditioned CTC yields better accuracy than using the uniform distribution does. Also, we found example-wise sampling gives better accuracy than batched sampling does, and allows smaller $\mathcal{V}^{\text{sampled}}$ for training. The models trained with example-wise sampling from Self-conditioned CTC reach or even outperform the baseline models with Self-conditioned CTC.

Figure 3 (a) shows memory consumption and test-other WER for various configurations of $|\mathcal{V}|$ and $|\mathcal{V}^{\text{sampled}}|$. Self-conditioned CTC improves baseline models with marginal extra memory consumption, and sampled softmax greatly reduces memory consumption with little or no degradation of accuracy.

We found using large vocabulary generally leads to better accuracy, as shown in Figure 3 (a). This shows the advantage of large vocabulary with sampled softmax over small vocabulary. Interestingly, the model with $|\mathcal{V}| = 2000$ and $|\mathcal{V}^{\text{sampled}}| = 600$ consumes less memory than the baseline model of $|\mathcal{V}| = 500$, because larger vocabulary leads to shorter target sequences.

Figure 3 (b) shows memory consumption of various configurations of $\mathcal{V}$ and $\mathcal{V}^{\text{sampled}}$. The baseline model of $|\mathcal{V}| = 2000$ takes nearly 850MB memory per training example, and the sampled softmax with $|\mathcal{V}^{\text{sampled}}| = 300$ only takes 530MB memory, and the amount of a logit tensor is now much smaller than the amount of encoder and predictor networks, displayed as a dotted line in Figure 3 (b).

Table 2: *Character error rates (CERs) on AISHELL-1 and CSJ-APS. No external language model is used.*

$\|\mathcal{V}^{\text{sampled}}\|$	AuxLoss	AISHELL-1 ($\|\mathcal{V}\| = 4231$) dev / test	CSJ-APS ($\|\mathcal{V}\| = 2753$) eval1 / eval2 / eval3
Baseline			
-		4.4 / 4.8	5.3 / 3.9 / 9.5
-	+InterCTC	**4.2** / 4.7	5.3 / 3.9 / **9.0**
Batched sampling from uniform			
500		4.4 / 4.9	5.5 / 4.0 / 9.4
1000		4.4 / 4.9	5.4 / 3.9 / 9.5
Example-wise sampling from uniform			
500		4.5 / 5.0	5.2 / 4.0 / 9.5
1000		4.4 / 4.9	5.4 / 3.9 / 9.4
Example-wise sampling from joint-CTC			
500		4.4 / 4.9	5.3 / 4.0 / 9.5
500	+InterCTC	**4.2** / **4.6**	5.2 / 3.9 / 9.3
500	+large batch	-	**5.1** / **3.7** / 9.1
1000		4.4 / 4.8	5.5 / 3.9 / 9.4

4.2. Results on AISHELL-1 and CSJ-APS

For AISHELL-1 and CSJ-APS, we follow the standard recipe [29] of ESPnet [42], which uses 12-layer Conformer [14] for the encoder network and 1-layer LSTM [43] for the prediction network. SpecAugment [41], speed perturbation, and joint CTC loss are applied for regularization. For decoding, we use CTC-constrained greedy decoding with top-100 labels as described in Section 3.3. Table 2 shows character error rates (CERs) on AISHELL-1 and CSJ-APS.

Similar to Section 4.1, sampling from the joint CTC distribution yields better accuracy than sampling from the uniform distribution, although the difference is small here. We found sampling from joint-CTC with $|\mathcal{V}^{\text{sampled}}| = 500$ is sufficient to match the baseline. The size of sample set is only 12% and 18% of vocabulary sets on AISHELL-1 and CSJ-APS, respectively.

For AISHELL-1, the model trained with sampled softmax and Intermediate CTC achieves the lowest CER in Table 2, and even outperforms the previous result in [29]. Also, for CSJ-APS, it is possible to increase batch size with sampled softmax due to 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 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. 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 subset. 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.*, "EESEN: 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. ICLR*, 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-IJCNLP*, 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. Boyer *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-mmi, ctc and rnn-t criteria for streaming asr," 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.