INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Improving Transformer-based End-to-End Speech Recognition with Connectionist Temporal Classification and Language Model Integration

Shigeki Karita[1], *Nelson Enrique Yalta Soplin*[2], *Shinji Watanabe*[3], *Marc Delcroix*[1], *Atsunori Ogawa*[1], *Tomohiro Nakatani*[1]

[1]NTT Communication Science Laboratories, Japan
[2]Waseda University, Japan
[3]Center for Language and Speech Processing, Johns Hopkins University, USA

karita@ieee.org

Abstract

The state-of-the-art neural network architecture named Transformer has been used successfully for many sequence-to-sequence transformation tasks. The advantage of this architecture is that it has a fast iteration speed in the training stage because there is no sequential operation as with recurrent neural networks (RNN). However, an RNN is still the best option for end-to-end automatic speech recognition (ASR) tasks in terms of overall training speed (i.e., convergence) and word error rate (WER) because of effective joint training and decoding methods. To realize a faster and more accurate ASR system, we combine Transformer and the advances in RNN-based ASR. In our experiments, we found that the training of Transformer is slower than that of RNN as regards the learning curve and integration with the naive language model (LM) is difficult. To address these problems, we integrate connectionist temporal classification (CTC) with Transformer for joint training and decoding. This approach makes training faster than with RNNs and assists LM integration. Our proposed ASR system realizes significant improvements in various ASR tasks. For example, it reduced the WERs from 11.1% to 4.5% on the Wall Street Journal and from 16.1% to 11.6% on the TED-LIUM by introducing CTC and LM integration into the Transformer baseline.

Index Terms: speech recognition, Transformer, connectionist temporal classification, language model

1. Introduction

End-to-end automatic speech recognition (ASR) systems have received a lot of attention recently because they simplify training and decoding procedures of ASR. Indeed, they directly learn speech-to-text mapping with purely neural network systems, while the hidden Markov model based systems need a hand-crafted pronunciation dictionary and a complex decoding system using a finite state transducer (FST) [1]. End-to-end ASR has been advanced by improving sequence-to-sequence (S2S) architectures, e.g., pyramid networks [2], [3], connectionist temporal classification (CTC) [4], convolutional neural network (CNN) encoders (i.e., VGG) [5], [6] and Transformer [7]. In addition, language model (LM) integration [2], [6] that has been widely used for the HMM based systems [8] is still effective for end-to-end ASR.

The state-of-the-art S2S architecture named Transformer [9] transduces sequential data with its self-attention mechanism, and this can replace the recurrent neural networks (RNN) in previous work [2], [3]. The self-attention mechanism learns time dependency inside the input sequence by employing attention matrices on the input frames. The motivation behind the self-attention mechanism is to achieve sequence transduction in parallel by utilizing global context i.e., all the frames. In contrast, the (bidirectional) RNN utilizes the global context but it requires sequential iterations. Moreover, the CNN transduces the frames in parallel but it utilizes the local context only within its receptive fields [9].

In this paper, we address two problems with Transformer that arose in our preliminary ASR experiments. One is slower convergence, namely its slower increase in validation accuracy over wall clock time, than RNN-based ASR. Transformer takes less time per iteration, but it takes many more epochs to converge. The other is the difficulty of LM integration in joint beam search decoding. In our preliminary experiments, the scores provided by Transformer and LM had drastically different behaviours that make them difficult to combine.

To solve these problems, we introduce CTC joint training and decoding to the Transformer-based ASR system. We implement CTC joint training as a multi-task learning by adding a new branch from Transformer encoder. It can make the convergence of the S2S model faster because CTC learns to align the speech feature and transcription explicitly [10], and thus it can help the S2S model learn monotonic attention for ASR [4]. Moreover, we found that the joint decoding with CTC could perform better with LM integration than the one without CTC in our experiments.

We summarize our contributions in this paper as follows.

- We combine two state-of-the-art ASR techniques; CTC/LM joint decoding [6], [11] and Transformer [7], [9] to achieve better ASR performance.
- We investigate the impact of joint training and decoding in three ASR tasks to evaluate more diverse (e.g., read and spontaneous) and larger scale (e.g., 81, 118 and 581 hours) datasets than existing works [7], [12].
- We implement Transformer on the open ASR toolkit ESPnet [13] as a reproducible source code[1].

2. Transformer architecture

In this section, we describe our baseline implementation of Transformer for ASR. We basically follow the configuration described in previous work [7] to measure the improvements provided by our extensions in the next section.

[1]our implementation is available at the release v.0.4.0 at https://github.com/espnet/espnet

http://dx.doi.org/10.21437/Interspeech.2019-1938

2.1. Overview

In our ASR framework, Transformer predicts an output sequence of character IDs Y from an input sequence of log-Mel filterbank speech features X^{fbank}. We can divide this Transformer architecture into two parts. One is an encoder network that converts X^{fbank} into an intermediate sequence of encoded features X_e. The other is a decoder network that predicts new character $Y[u+1]$ given X_e and prefix characters $Y[1], \ldots, Y[u]$. Both networks consist of attention and feedforward network modules. A major difference between Transformer and RNN is a self attention mechanism that transforms the sequence by employing attention matrices on the input frames instead of the recurrent connection. The remainder of this section describes these attention mechanism, encoder and decoder. Finally, we review training and decoding to emphasize the difference with our proposed method.

2.2. Multi-head attention

Transformer contains a dot-attention layer [14] with scaling:

$$\mathrm{att}(X^{\mathrm{q}}, X^{\mathrm{k}}, X^{\mathrm{v}}) = \mathrm{softmax}\left(\frac{X^{\mathrm{q}} X^{\mathrm{k}\top}}{\sqrt{d^{\mathrm{att}}}}\right) X^{\mathrm{v}}, \quad (1)$$

where $X^{\mathrm{k}}, X^{\mathrm{v}} \in \mathbb{R}^{n^{\mathrm{k}} \times d^{\mathrm{att}}}$ and $X^{\mathrm{q}} \in \mathbb{R}^{n^{\mathrm{q}} \times d^{\mathrm{att}}}$ are input sequences for this attention layer, d^{att} is the number of feature dimensions, n^{q} is the length of X^{q} and n^{k} is the length of X^{k} and X^{v}. We refer to $X^{\mathrm{q}} X^{\mathrm{k}\top}$ as the "attention matrix". Vaswani et al. [9] considered these inputs X^{q}, X^{k} and X^{v} as a query and a set of key-value pair, respectively.

In addition, to allow the model to pay multiple attentions in parallel, Vaswani et al. [9] extended this attention layer in Eq. (1) to multi-head attention (MHA):

$$\mathrm{MHA}(Q, K, V) = [H_1, H_2, \ldots, H_{d^{\mathrm{head}}}] W^{\mathrm{head}}, \quad (2)$$

$$H_h = \mathrm{att}(Q W_h^{\mathrm{q}}, K W_h^{\mathrm{k}}, V W_h^{\mathrm{v}}), \quad (3)$$

where $K, V \in \mathbb{R}^{n^{\mathrm{k}} \times d^{\mathrm{att}}}$ and $Q \in \mathbb{R}^{n^{\mathrm{q}} \times d^{\mathrm{att}}}$ are input sequences for this MHA layer, $H_h \in \mathbb{R}^{n^{\mathrm{q}} \times d^{\mathrm{att}}}$ is the h-th attention layer output ($h = 1, \ldots, d^{\mathrm{head}}$), $W_h^{\mathrm{q}}, W_h^{\mathrm{k}}, W_h^{\mathrm{v}} \in \mathbb{R}^{d^{\mathrm{att}} \times d^{\mathrm{att}}}$ and $W^{\mathrm{head}} \in \mathbb{R}^{d^{\mathrm{att}} d^{\mathrm{head}} \times d^{\mathrm{att}}}$ are learnable weight matrices and d^{head} is the number of attentions in this layer.

2.3. Speech encoder architecture

The input speech is represented as a sequence of 80-dim log-Mel filterbank features $X^{\mathrm{fbank}} \in \mathbb{R}^{n^{\mathrm{fbank}} \times 80}$, where n^{fbank} is the length of the input. First, we subsample X^{fbank} to $X^{\mathrm{sub}} \in \mathbb{R}^{n^{\mathrm{sub}} \times d^{\mathrm{att}}}$ by using two layer time-axis CNN with ReLU activation, d^{att} channels, stride size 2 and kernel size 3 in [7], where n^{sub} is the length of the output sequence of the CNN. Then, the Transformer's encoder network transforms this input into a sequence of encoded features $X_e \in \mathbb{R}^{n^{\mathrm{sub}} \times d^{\mathrm{att}}}$ for the CTC and decoder network as follows:

$$X_0 = [[X^{\mathrm{sub}}[1], \mathrm{PE}[1]]^\top, \ldots, [X^{\mathrm{sub}}[n_{\mathrm{sub}}], \mathrm{PE}[n_{\mathrm{sub}}]]^\top],$$

$$X'_i = X_i + \mathrm{MHA}_i(X_i, X_i, X_i), \quad (4)$$

$$X_{i+1} = X'_i + \mathrm{FF}_i(X'_i), \quad (5)$$

where $i = 0, \ldots, e$ is the index of encoder layers, e is the number of encoder layers, PE is sinusoidal positional encoding [9]:

$$\mathrm{PE}[t] = \begin{cases} \sin \frac{t}{10000^{t/d^{\mathrm{att}}}} & \text{if } t \text{ is even,} \\ \cos \frac{t}{10000^{t/d^{\mathrm{att}}}} & \text{if } t \text{ is odd.} \end{cases} \quad (6)$$

FF_i is the i-th two-layer feedforward network:

$$\mathrm{FF}(X[t]) = \mathrm{ReLU}(X[t] W_1^{\mathrm{ff}} + b_1^{\mathrm{ff}}) W_2^{\mathrm{ff}} + b_2^{\mathrm{ff}}, \quad (7)$$

where $X[t] \in \mathbb{R}^{d^{\mathrm{att}}}$ is the t-th frame of the input sequence X, $W_1^{\mathrm{ff}} \in \mathbb{R}^{d^{\mathrm{att}} \times d^{\mathrm{ff}}}$, $W_2^{\mathrm{ff}} \in \mathbb{R}^{d^{\mathrm{ff}} \times d^{\mathrm{att}}}$ are learnable weight matrices, $b_1^{\mathrm{ff}} \in \mathbb{R}^{d^{\mathrm{ff}}}$, $b_2^{\mathrm{ff}} \in \mathbb{R}^{d^{\mathrm{att}}}$ are learnable bias vectors. We refer to $\mathrm{MHA}(X_i, X_i, X_i)$ in Eq. (4) as the "self attention".

2.4. Character decoder architecture

Transformer's decoder receives the encoded sequence X_e in Eq. (5) and prefix sequence of character (e.g., alphabet, Japanese character, etc) IDs $Y[1:u] = Y[1], \ldots, Y[u]$. Then, it predicts their subsequent IDs $Y[2:u+1]$ as follows:

$$\begin{aligned} E &= \mathrm{Embed}(Y[1:u]), \\ Z_0 &= [[E[1], \mathrm{PE}[1]]^\top, \ldots, [E[u], \mathrm{PE}[u]]^\top], \\ Z'_j &= Z_j + \mathrm{MHA}_j^{\mathrm{self}}(Z_j, Z_j, Z_j), \\ Z''_j &= Z_j + \mathrm{MHA}_j^{\mathrm{src}}(Z'_j, X_e, X_e), \\ Z_{j+1} &= Z''_j + \mathrm{FF}_j(Z''_j), \end{aligned} \quad (8)$$

where $\mathrm{Embed}(\cdot)$ is an embedding layer that transforms the sequence of the character IDs Y into a sequence of learnable vectors indexed by the IDs $E \in \mathbb{R}^{u \times d^{\mathrm{att}}}$, $j = 0, \ldots, d$ is the index of decoder layers and d is the number of decoder layers. Finally, the decoder emits the posterior probabilities of the next character ID $Y[u+1]$ given its prefix IDs $Y[1], \ldots, Y[u]$ and the encoded speech X_e:

$$\begin{aligned} &p_{\mathrm{s2s}}(Y|X_e) = \prod_u p_{\mathrm{s2s}}(Y[u]|Y[1:u-1], X_e)], \\ &[p_{\mathrm{s2s}}(Y[2]|Y[1], X_e), \ldots, p_{\mathrm{s2s}}(Y[u+1]|Y[1:u], X_e)] \\ &= \mathrm{softmax}\left(Z_d W^{\mathrm{att}} + b^{\mathrm{att}}\right), \end{aligned}$$

where $W^{\mathrm{att}} \in \mathbb{R}^{d^{\mathrm{att}} \times d^{\mathrm{char}}}$, $b^{\mathrm{att}} \in \mathbb{R}^{d^{\mathrm{char}}}$ are learnable parameters and d^{char} is the number of characters.

2.5. Training and decoding

In training stage, Transformer's decoder predicts all the character frames as $p_{\mathrm{s2s}}(Y|X_e)$, where Y is a ground truth sequence of the characters, and computes its training loss

$$L_{\mathrm{s2s}} = -\log p_{\mathrm{s2s}}(Y|X_e) \quad (9)$$

at once in parallel unlike RNN-based S2S because there is no sequential operation. $\mathrm{MHA}(\cdot)$ ignores the lower triangular part in the attention matrix $X^{\mathrm{q}} X^{\mathrm{k}\top}$ in Eq. (1) to prevent the output sequence from attending to subsequent positions in the query sequence X^{q}.

In the decoding stage, Transformer's decoder predicts individual characters sequentially using prefix beam search like RNN-based S2S to find the most likely hypothesis:

$$\hat{Y} = \arg\max_{Y \in \mathcal{Y}^*} \log p_{\mathrm{s2s}}(Y|X_e), \quad (10)$$

where $\mathcal{Y}^*$ is a set of output hypotheses.

3. Joint training and decoding

In this section, we explain our extensions to the baseline Transformer described in the previous section.

3.1. Connectionist temporal classification

CTC explicitly learns the monotonic alignment between the speech feature and character transcription. CTC joint training helps the S2S's attention to be monotonic, which is reasonable for an ASR task. Thus it results in faster convergence [4]. Similarly, we introduce a new CTC branch from Transformer encoder output X_e in addition to the decoder branch.

First, the CTC layer receives the output sequence of the encoder X_e. It then computes the probability $p_{\mathrm{ctc}}(Y|X_e)$ over an arbitrary alignment π between X_e and Y, where $\pi[t]$ is a character ID aligned to the t-th frame in X_e as follows:

$$C = \mathrm{softmax}(X_e W^{\mathrm{ctc}} + b^{\mathrm{ctc}}), \tag{11}$$

$$p(\mathcal{B}(\pi) = Y|X_e) = \prod_{t=1}^{n^{\mathrm{sub}}} C[t, \pi[t]], \tag{12}$$

$$p_{\mathrm{ctc}}(Y|X_e) = \sum_{\pi' \in \mathcal{B}^{-1}(Y)} p(\mathcal{B}(\pi) = Y|X_e), \tag{13}$$

where $W^{\mathrm{ctc}} \in \mathbb{R}^{d^{\mathrm{att}} \times d^{\mathrm{char}}}$, $b^{\mathrm{ctc}} \in \mathbb{R}^{d^{\mathrm{char}}}$ are learnable parameters, $C \in \mathbb{R}^{n^{\mathrm{sub}} \times d^{\mathrm{char}}}$ is CTC's output and $C[t, \pi[t]]$ is the probability of the alignment between the output character $\pi[t]$ and the t-th frame in X_E. The many-to-one mapping $\mathcal{B}(\pi)$ removes redundant symbols from the alignment π, for example, $\mathcal{B}(\mathrm{aa}\phi\mathrm{b}) = \mathrm{ab}$, where ϕ is a blank symbol, one-to-many mapping $\mathcal{B}^{-}1$ projects a string into a set of strings with redundant symbols: $\mathcal{B}^{-1}(Y) = \{\pi | Y = \mathcal{B}(\pi)\}$.

In the training stage, we adopt multi-task loss [4], which combines the negative log probability from S2S and CTC:

$$L_{\mathrm{mtl}} = -\alpha L_{\mathrm{s2s}} - (1-\alpha) \log p_{\mathrm{ctc}}(Y|X_e), \tag{14}$$

where α is a hyperparameter. We use efficient dynamic programming in [10] to compute the CTC term $p_{\mathrm{ctc}}(Y|X_e)$.

3.2. Joint decoding with CTC and LM

CTC has an ASR ability in Eq. (13), and so we can exploit its prediction during decoding. We followed the common joint decoding approach [15], [16], which simply takes the sum of log probabilities from the models as follows:

$$\hat{Y} = \arg\max_{Y \in \mathcal{Y}^*} \{\lambda \log p_{\mathrm{s2s}}(Y|X_e) + (1-\lambda) \log p_{\mathrm{ctc}}(Y|X_e) + \gamma \log p_{\mathrm{lm}}(Y)\}, \tag{15}$$

where $p_{\mathrm{lm}}(Y)$ is the LM probability of Y, λ and γ are hyperparameters named "CTC weight" and "LM weight", respectively. Our implementation is based on [16], which computes $\arg\max_{Y \in \mathcal{Y}^*}$ over a prefix tree by employing on-the-fly scoring.

4. Related work

There has been previous work on joint decoding in end-to-end ASR systems. Early systems have already performed joint decoding with the FST based LM [2], [3]. Recent advances in joint decoding include joint decoding methods for CTC and char/word-level RNN-based LM [16]. These state-of-the-art systems have still been implemented on RNN-based S2S. Recently, Dong et al. [7] reported a fast training time of 1.2 days and a low word error rate (WER) of 10.9% obtained by using Transformer on the Wall Street Journal (WSJ) corpus. However, there have already been better reports of a faster training time of 5 hours using RNN-based S2S [13] and a lower WER of 5.1% using RNN-based CTC and LM [6]. In fairness to the Transformer-based systems, on WSJ [7] and HKUST [12], they achieved a lower character error rate (CER) or WER than the RNN-based systems without CTC and LM [6]. Therefore, we focus on the combination of Transformer, CTC and LM to achieve faster training and higher recognition performance.

5. Experiment

5.1. Tasks

Except for the Transformer architecture and its hyperparameters, our experiments are based on existing recipes in the end-to-end speech processing toolkit ESPnet [13][2]. In other words, we set almost all the configurations so that they were the best for the RNN-based model using a VGG-like encoder [6], which is the default architecture in ESPnet. In this section, we refer to that baseline as "RNN".

We chose three different ASR tasks for evaluation. For the first task, we used the WSJ corpus as a read English ASR task, because it has been widely used for the evaluation of previous end-to-end ASR systems. For the second task, we adopted TED-LIUM release 2 [17], which contains 135 hours of TED talks, as a spontaneous English speech task. For the third task, we used the Corpus of Spontaneous Japanese (CSJ) [18], which contains 581 hours of Japanese lectures, because we are also interested in ASR for non-English languages. Unlike the English tasks, the CSJ recipe does not learn word boundary (i.e., space symbol) because Japanese word boundaries are ambiguous. Therefore, we only evaluate CER on the CSJ.

5.2. Training and decoding setup

As suggested in [7], we used the configuration named "big model" for Transformer ($d^{\mathrm{att}} = 256, d^{\mathrm{ff}} = 2048, d^{\mathrm{head}} = 4, e = 12, d = 6$). We initialized Transformer's weights by using the LeCun Uniform method [19] instead of He Uniform [20] used in the original implementation [21]. The training stage of Transformer runs stochastic gradient descent (SGD) over a training set with the Adam update rule [22] using square root learning rate scheduling [9] (25000 warmup steps, 64 minibatch size and 100 epochs). The multi-task loss weight was $\alpha = 0.3$ for the CTC joint training. We also applied three regularization methods; 10% dropout on every attention matrix $X^{\mathrm{q}} X^{\mathrm{k}\top}$ and weight in $\mathrm{FF}(\cdot)$, layer normalization [23] before every $\mathrm{MHA}(\cdot)$ and $\mathrm{FF}(\cdot)$, and label smoothing [24] with a penalty of 0.1 because training without these regularization methods always overfits to the training set. All the hyperparameters are same for all the corpora except for the tunable scalar k in [7] (10.0 for WSJ and 1.0 for others). Our Transformer training always runs 100 epochs over the training set and it does not use an early stopping strategy with a validation set. Finally, the decoding stage runs in the same way as RNN-based S2S using the word-level LM [16] for the WSJ and TED-LIUM corpora, and character-level LM [6] for the CSJ corpus. The LM consists of two long short-term memory layers with 1024 units. We trained the LMs on external text only datasets for WSJ and TED-LIUM, while the LM for CSJ learned the same transcriptions for ASR training. The decoding hyperparameters λ, γ were searched over the development sets for each corpus.

Table 1: *WSJ results*

System	dev93 CER	dev93 WER	eval92 CER	eval92 WER
Transformer [7]	-	-	-	10.9
Self-attention CTC+LM [25]	6.4	8.9	4.7	5.9
RNN+CTC/LM decode [16]	-	8.4	-	5.1
Our RNN	7.7	20.8	5.8	16.7
+CTC/LM decode	4.0	9.0	2.8	6.0
Our Transformer	6.9	14.3	4.1	11.1
+CTC train	5.1	14.2	3.4	10.6
+CTC decode	4.7	13.8	3.2	10.2
+CTC/LM decode	**3.4**	**7.7**	**1.9**	**4.5**

Table 2: *TED-LIUM release 2 results*

System	dev CER	dev WER	test CER	test WER
RNN+CTC/LM decode [26]	10.8	19.8	10.1	18.6
Our Transformer	16.6	24.6	8.7	16.1
+CTC train	13.7	20.9	8.8	15.5
+CTC decode	7.4	15.8	6.5	14.2
+CTC/LM decode	**6.6**	**13.1**	**6.0**	**11.6**

Table 3: *CSJ results. As there is no official development set, we denote the first 4000 utterances in the training set as "dev".*

System	dev CER	eval1 CER	eval2 CER	eval3 CER
RNN [6]	-	11.4	7.9	9.0
+CTC/LM decode [6]	-	7.9	5.8	6.7
Our Transformer	6.0	7.7	5.8	5.9
+CTC train	5.9	7.3	5.0	5.8
+CTC decode	5.8	7.1	4.9	5.6
+CTC/LM decode	**5.5**	**6.5**	**4.5**	**5.2**

5.3. Character and word error rates

Tables 1, 2 and 3 show character and word error rates in the WSJ, TED-LIUM release 2 and CSJ corpora, respectively. Our reimplementation of Transformer results in a similar WER to that in a previous study [7] on the WSJ in Table 1. As seen in Tables 1, 2 and 3, our extensions on Transformer labeled "CTC train", "CTC decode" and "CTC/LM decode" gradually reduce both the CER and WER on all the corpora.

5.4. Training

Figure 1 shows the validation accuracy over time for Transformer with or without CTC, and RNN with CTC labeled "Transformer+CTC", "Transformer" and 'RNN+CTC', respectively. We trained every model on the GTX1080ti. We can see that CTC joint training makes Transformer's convergence faster than the baseline RNN+CTC, while Transformer without CTC appears to converge more slowly in our experiment.

Figure 2 shows attention plots at fifth epoch between the encoded input speech and last decoder's output inside Transformer trained with and without CTC, respectively. According to [4], CTC can encourage monotonic alignment between the speech and transcription. Therefore, the attention at an early epoch with CTC appears more monotonic than that without CTC in Figure 2. This is why our Transformer+CTC converged faster.

5.5. Decoding

Table 4 summarizes WERs on the WSJ dev93 set of the

[2]We developed our baseline systems on ESPnet version 0.3.0. It was the latest release at the time of our experiments.

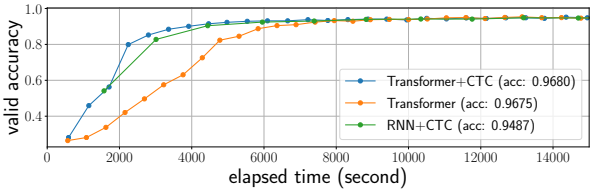

Figure 1: *Validation accuracy on WSJ dev93 over time. Each data point corresponds to the end of each epoch.*

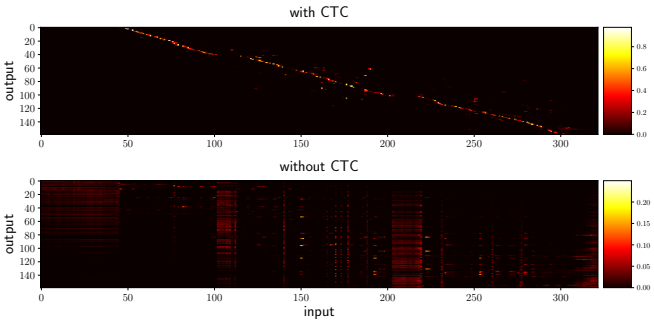

Figure 2: *Attention plots between the encoded input speech and last decoder output for a WSJ 4k0c301 provided by Transformer with CTC (top) and without CTC (bottom) at the fifth epoch.*

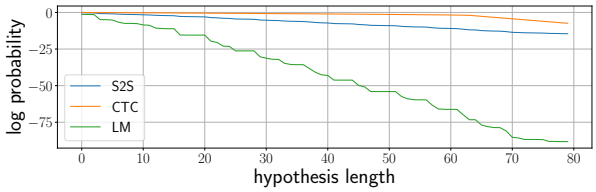

Figure 3: *Relationships between hypothesis log probability and length for each model.*

Table 4: *Transformer WERs on WSJ dev93.*

		LM γ 0.0	0.01	0.1	1.0	10.0
CTC λ	0.0	14.2	69.9	83.4	93.3	95.2
	0.1	13.7	13.6	12.3	7.8	86.3
	0.3	14.4	14.2	12.8	**7.7**	74.0
	0.5	15.7	15.5	13.5	8.5	56.9

Transformer-based system for each CTC and LM joint decoding weight. Figure 3 shows the relationships between log probabilities and the length of the decoded hypothesis for each score in Transformer-based system.

In Table 4, we found that CTC joint decoding greatly reduces the WER in LM integration, while it appears difficult to integrate with LM without CTC. As seen in Figure 3, the log probability of Transformer's two outputs labeled "S2S" and "CTC" are obviously higher than LM's log probability and robust to the change of length. Because of this, the decoded hypotheses with LM without CTC tend to be shorter and result in worse WERs at an CTC weight of 0.0 in Table 4.

6. Conclusions

We proposed a novel ASR system that combines Transformer with CTC and LM. Our experiments resulted in a large reduction in CER/WER on diverse ASR tasks. In addition, it shows that Transformer with CTC joint training could converge faster to a better validation accuracy than an RNN-based system.

7. References

[1] M. Mohri, F. Pereira, and M. Riley, "Speech recognition with weighted finite-state transducers," *Springer Handbook of Speech Processing*, pp. 559–584, 2008.

[2] D. Bahdanau, J. Chorowski, D. Serdyuk, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4945–4949, 2016.

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

[4] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017, pp. 4835–4839.

[5] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4845–4849, 2017.

[6] T. Hori, S. Watanabe, Y. Zhang, and W. Chan, "Advances in joint CTC-attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM," in *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*, 2017, pp. 949–953.

[7] L. Dong, S. Xu, and B. Xu, "Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018, pp. 5884–5888.

[8] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur, "Recurrent neural network based language model," *Proceedings of International Conference on Spoken Language Processing, INTERSPEECH*, pp. 1045–1048, 2010.

[9] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., 2017, pp. 5998–6008.

[10] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *ICML*, ser. ACM International Conference Proceeding Series, vol. 148, 2006, pp. 369–376.

[11] A. Zeyer, K. Irie, R. Schluter, and H. Ney, "Improved training of end-to-end attention models for speech recognition," in *Proceedings of International Conference on Spoken Language Processing, INTERSPEECH*, 2018, pp. 7–11.

[12] S. Zhou, L. Dong, S. Xu, and B. Xu, "Syllable-based sequence-to-sequence speech recognition with the transformer in Mandarin Chinese," in *Proceedings of International Conference on Spoken Language Processing, INTERSPEECH*, 2018, pp. 791–795.

[13] S. Watanabe, T. Hori, S. Karita, *et al.*, "ESPnet: End-to-end speech processing toolkit," in *Proceedings of International Conference on Spoken Language Processing, INTERSPEECH*, 2018, pp. 2207–2211.

[14] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1412–1421.

[15] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *International Conference on Learning Representations*, 2015.

[16] T. Hori, J. Cho, and S. Watanabe, "End-to-end speech recognition with word-based rnn language models," in *2018 IEEE Spoken Language Technology Workshop (SLT)*, 2018, pp. 389–396.

[17] A. Rousseau, P. Deléglise, and Y. Estève, "TED-LIUM: An automatic speech recognition dedicated corpus," in *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, 2012.

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

[19] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, 1998, pp. 9–50.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[21] A. Vaswani, S. Bengio, E. Brevdo, *et al.*, "Tensor2tensor for neural machine translation," in *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, 2018, pp. 193–199.

[22] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, pp. 1–13, 2014.

[23] L. J. Ba, R. Kiros, and G. E. Hinton, "Layer normalization," *CoRR*, vol. abs/1607.06450, 2016.

[24] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826.

[25] J. Salazar, K. Kirchhoff, and Z. Huang, "Self-attention networks for connectionist temporal classification in speech recognition," *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*,

[26] ESPnet, `https://github.com/espnet/espnet/blob/v.0.3.1/egs/tedlium/asr1/RESULTS`.