Self-Distillation for Improving CTC-Transformer-based ASR Systems

Takafumi Moriya, Tsubasa Ochiai, Shigeki Karita, Hiroshi Sato, Tomohiro Tanaka, Takanori Ashihara, Ryo Masumura, Yusuke Shinohara, Marc Delcroix

NTT Corporation, Japan

takafumi.moriya.nd@hco.ntt.co.jp

Abstract

We present a novel training approach for encoder-decoder-based sequence-to-sequence (S2S) models. S2S models have been used successfully by the automatic speech recognition (ASR) community. The important key factor of S2S is the attention mechanism as it captures the relationships between input and output sequences. The attention weights inform which time frames should be attended to for predicting the output labels. In previous work, we proposed distilling S2S knowledge into connectionist temporal classification (CTC) based models by using the attention characteristics to create pseudo-targets for an auxiliary cross-entropy loss term. This approach can significantly improve CTC models. However, it remained unclear whether our proposal could be used to improve S2S models. In this paper, we extend our previous work to create a strong S2S model, i.e. Transformer with CTC (CTC-Transformer). We utilize Transformer outputs and the source attention weights for making pseudo-targets that contain both the posterior and the timing information of each Transformer output. These pseudo-targets are used to train the shared encoder of the CTC-Transformer through the use of direct feedback from the Transformer-decoder and thus obtain more informative representations. Experiments on public and private datasets to perform various tasks demonstrate that our proposal is also effective for enhancing S2S model training. In particular, on a Japanese ASR task, our best system outperforms the previous state-of-the-art alternative.

Index Terms: speech recognition, neural network, Transformer, connectionist temporal classification, attention weight

1. Introduction

Automatic speech recognition (ASR) systems have been dramatically improved with the introduction of deep neural networks (DNN). Conventional ASR systems currently consist of several modules including acoustic, lexicon, and language models (LMs). End-to-end (E2E) ASR systems train these models jointly; they map acoustic feature sequences to token sequences directly. Several E2E ASR frameworks have been proposed, e.g. connectionist temporal classification (CTC) [1,2], recurrent neural network-transducer (RNN-T) [3, 4], and attention-based sequence-to-sequence encoder-decoder (S2S) [5–9]. Among them, the joint CTC-Attention (S2S+CTC) model that consists of CTC- and S2S-branch heads stacked on a shared-encoder has shown promising results [10, 11]. In this paper, we focus on improving S2S+CTC.

An important part of S2S is its attention weights, as they capture the relationships between input and output sequences. In a preceding work, we proposed a knowledge distillation approach that leverages the S2S characteristics for CTC model training [12]. Our proposal utilized the outputs and the attention weights of a trained S2S model for making pseudo-targets, called attention matrix. The attention matrix contains information of both the posterior probability and the spike timing of each S2S output. We showed that the encoder of CTC model could be improved by adding an auxiliary cross-entropy (CE) loss term between the attention matrix and the frame-by-frame outputs of the model.

However, it remained unclear whether our proposal could be benefit for S2S+CTC models. In this paper, we investigate the use of the pseudo-targets based CE loss term in S2S+CTC models. We propose to transfer the knowledge of the decoder into the encoder during the training. This training scheme is called self-distillation (SD). Attention matrices related to decoder knowledge are used to train the encoder via a branch layer for CE loss computation. We expect that the attention matrix acts as direct decoder feedback that can enhance shared-encoder training to obtain more informative representations for improving decoder performance.

We explore our proposed approach using CTC-Transformer [13], which is a strong S2S+CTC model that yields better results than RNN-based model on several ASR tasks [14]. In preceding work, we addressed just an RNN-based S2S with single head attention. Transformer uses a multi-head attention (MHA). In this paper, we also extend our proposal to the MHA case and investigate its effectiveness.

We evaluate our proposal on three English and two Japanese ASR tasks: Wall Street Journal (WSJ) [15], Switchboard [16], Librispeech [17], corpus of spontaneous Japanese (CSJ) [18] and NTT Japanese voice search dataset. The results demonstrate that the proposal yields better trained models than the strong baseline of conventional CTC-Transformer. Moreover, our best system on CSJ achieves a 3.9% character error rate (CER) on the test set, which outperforms the previous state-of-the-art scheme by 18.2%.

2. ASR modules

We explain here Transformer and CTC, which are used as the baseline in this work. $\boldsymbol{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_T)$ and $Y = (y_1, ..., y_L)$ are the input acoustic feature sequence of length-T and token sequence of length-L, respectively, where $y_l \in \{1, ..., K\}$; K is the number of tokens that include two special outputs. Here tokens can be characters, words or subword units. The tokens also include two special characters that represent "end-of-sentence" label for Transformer, and "blank" label for CTC.

2.1. Tranformer

2.1.1. Multi-head attention (MHA)

Transformer is composed of some encoder- and decoder-blocks. Each block has a scaled dot-product attention mechanism. The basic attention mechanism is defined as follows:

$$\text{Attention}\left(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\right) = \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathsf{T}}}{\sqrt{d^k}}\right)\boldsymbol{V} = \boldsymbol{P}\boldsymbol{V}, \tag{1}$$

$\boldsymbol{Q} \in \mathbb{R}^{n^q \times d^q}$, $\boldsymbol{K} \in \mathbb{R}^{n^k \times d^k}$, and $\boldsymbol{V} \in \mathbb{R}^{n^v \times d^v}$ represent query, key and value matrices, respectively. d^* and n^* in each element mean the number of feature dimensions and sequence lengths, respectively. We note that these indexes must satisfy $n^k = n^v$ and $d^q = d^k$. $\boldsymbol{P} \in \mathbb{R}^{n^q \times n^k}$ is a matrix containing the attention weights. Note that the attention mechanism in Transformer consists of a multi-head attention (MHA) mechanism described as follows:

$$\text{MHA}(\boldsymbol{Q'}, \boldsymbol{K'}, \boldsymbol{V'}) = [\text{Head}_1, \cdots, \text{Head}_{d^h}]\, \mathbf{W}^H, \quad (2)$$

$$\text{Head}_h = \text{Attention}\left(\boldsymbol{Q'}\boldsymbol{W}_h^Q, \boldsymbol{K'}\boldsymbol{W}_h^K, \boldsymbol{V'}\boldsymbol{W}_h^V\right), \quad (3)$$

where Head_h is the output of the h-th attention head, and d^h is the total number of heads. d^{model} represents the number of input feature dimensions. $\boldsymbol{W}_h^Q \in \mathbb{R}^{d^{\text{model}} \times d^q}$, $\boldsymbol{W}_h^K \in \mathbb{R}^{d^{\text{model}} \times d^k}$, $\boldsymbol{W}_h^V \in \mathbb{R}^{d^{\text{model}} \times d^v}$, and $\boldsymbol{W}^H \in \mathbb{R}^{d^h d^v \times d^{\text{model}}}$ are the projection matrices that satisfy $d^q = d^k = d^v = d^{\text{model}}/d^h$.

2.1.2. Speech encoder of Transformer

The encoder transforms $\boldsymbol{X}$ into intermediate feature representation $\boldsymbol{E} = (\boldsymbol{e}_1, ..., \boldsymbol{e}_{T'})$. First, $\boldsymbol{X}$ is subsampled and embeded into $\boldsymbol{Z}^{\text{e}} = (\boldsymbol{z}_1^{\text{e}}, ..., \boldsymbol{z}_{T'}^{\text{e}})$ with length-T' via functions, "Conv2D($\cdot$)" and "SpeechEmbed($\cdot$)", which represent 2-dimensional convolution and linear embeddings with learnable parameters. Next, $\boldsymbol{Z}^{\text{e}}$ is encoded with positional encoding into $\boldsymbol{Z}^{(0)}$. Then, $\boldsymbol{Z}^{(0)}$ is transformed into $\boldsymbol{E}$ via a self-attention layer and feedforward (FF) network. The above operations to obtain $\boldsymbol{E}$ are defined as follows:

$$\begin{aligned}
\boldsymbol{Z}^{\text{e}} &= \text{SpeechEmbed}(\text{Conv2D}(\boldsymbol{X})), && (4)\\
\boldsymbol{Z}^{(0)} &= \left(\boldsymbol{z}_1^{\text{e}} + \text{PE}(1), \cdots, \boldsymbol{z}_{T'}^{\text{e}} + \text{PE}(T')\right), && (5)\\
\boldsymbol{Z'}^{(n)} &= \boldsymbol{Z}^{(n)} + \text{MHA}\left(\boldsymbol{Z}^{(n)}, \boldsymbol{Z}^{(n)}, \boldsymbol{Z}^{(n)}\right), && (6)\\
\boldsymbol{Z}^{(n+1)} &= \boldsymbol{Z'}^{(n)} + \text{FF}^{(n)}\left(\boldsymbol{Z'}^{(n)}\right), && (7)\\
\boldsymbol{E} &= \boldsymbol{Z}^{(N)}, && (8)
\end{aligned}$$

where $n = 1, \cdots, N-1$ indicates the index of the N encoder blocks. "PE($\cdot$)" is sinusoidal positional encoding:

$$\text{PE}(i) = \begin{cases} \sin \frac{i}{10000^{i/d^{\text{model}}}} & \text{if } i \text{ is even}, \\ \cos \frac{i}{10000^{i/d^{\text{model}}}} & \text{if } i \text{ is odd}. \end{cases} \quad (9)$$

The value $\text{PE}(i)$ is expanded to a d^{model} dimensional vector that contains the same values, and added to the embedded features. The FF function, "FF($\cdot$)", is two-layer feedforward network with a rectified linear unit (ReLU), and is described as follows:

$$\text{FF}(\boldsymbol{Z}) = \text{ReLU}\left(\boldsymbol{Z}\boldsymbol{W}_1^{\text{ff}} + \boldsymbol{b}_1^{\text{ff}}\right)\boldsymbol{W}_2^{\text{ff}} + \boldsymbol{b}_2^{\text{ff}}, \quad (10)$$

where $\boldsymbol{W}_1^{\text{ff}} \in \mathbb{R}^{d^{\text{model}} \times d^{\text{ff}}}$, $\boldsymbol{W}_2^{\text{ff}} \in \mathbb{R}^{d^{\text{ff}} \times d^{\text{model}}}$ are learnable weight matrices, and $\boldsymbol{b}_1^{\text{ff}} \in \mathbb{R}^{d^{\text{ff}}}$, $\boldsymbol{b}_2^{\text{ff}} \in \mathbb{R}^{d^{\text{model}}}$ are learnable bias vectors. The FF in each block have different learnable parameters. The intermediate representation features $\boldsymbol{E}$ are fed into the decoder blocks.

2.1.3. Token decoder of Transformer

The decoder transforms $Y_{1:l-1} = (y_1, ..., y_{l-1})$ and $\boldsymbol{E}$ into $\hat{\boldsymbol{y}}_l$. First, each token y is embedded into real-valued vector $\boldsymbol{y}^{\text{e}} \in \mathbb{R}^{d^{\text{model}}}$ via an embedding layer, "TokenEmbed($\cdot$)", with positional encoding. Then, the embedded features, $\boldsymbol{Y}^{(0)} = \left(\boldsymbol{y}_1^{(0)}, \cdots, \boldsymbol{y}_{l-1}^{(0)}\right)$, are transformed into the prediction $\hat{\boldsymbol{y}}_l$ via two attention layers, which have self- and source-attention mechanisms, and the FF network. The above operations to obtain $\hat{\boldsymbol{y}}_l$ are defined as follows:

$$\begin{aligned}
\boldsymbol{Y}^{\text{e}} &= \text{TokenEmbed}(Y_{1:l-1}), && (11)\\
\boldsymbol{Y}^{(0)} &= \left(\boldsymbol{y}_1^{\text{e}} + \text{PE}(1), \cdots, \boldsymbol{y}_l^{\text{e}} + \text{PE}(l)\right), && (12)\\
\boldsymbol{Y'}^{(m)} &= \mathbf{Y}^{(m)} + \text{MHA}\left(\boldsymbol{Y}^{(m)}, \boldsymbol{Y}^{(m)}, \boldsymbol{Y}^{(m)}\right), && (13)\\
\boldsymbol{Y''}^{(m)} &= \boldsymbol{Y'}^{(m)} + \text{MHA}\left(\boldsymbol{Y'}^{(m)}, \boldsymbol{E}, \boldsymbol{E}\right), && (14)\\
\boldsymbol{Y}^{(m+1)} &= \mathbf{Y''}^{(m)} + \text{FF}^{(m)}\left(\boldsymbol{Y''}^{(m)}\right), && (15)\\
\hat{\boldsymbol{y}}_l &= \text{Softmax}\left(\boldsymbol{Y}^{(M)}\boldsymbol{W}^O + \boldsymbol{b}^O\right), && (16)
\end{aligned}$$

where $m = 1, \cdots, M-1$ indicates the index of the M decoder blocks. $\boldsymbol{W}^O \in \mathbb{R}^{d^{\text{model}} \times K}$ and $\boldsymbol{b}^O \in \mathbb{R}^K$ are the learnable weight matrix and bias vector, respectively. The prediction $\hat{\boldsymbol{y}}_l$ in Eq. (16) is regarded as $p(y_l|y_{1:l-1}, \boldsymbol{X})$ that corresponds to the probability of a ground-truth token class. The Transformer model is optimized by an objective function as follows:

$$\mathcal{L}_{\text{S2S}} = -\log p(Y|\boldsymbol{X}) = -\sum_{l=1}^{L} \log p(y_l|y_{1:l-1}, \boldsymbol{X}). \quad (17)$$

The CE loss is calculated between the predicted- and the correct token sequence.

2.2. Connectionist temporal classification (CTC)

CTC uses the target labels and the extra "blank" label, ϕ, and learns the mapping between sequences of different lengths. By inserting a blank label between two consecutive labels and allowing each label to be repeated, label sequence Y can be expanded into $\Omega(Y)$, a set of length-T' sequences. Conversely, each CTC path $\boldsymbol{\pi} \in \Omega(Y)$ with redundancy can be reduced to the original label sequence Y by removing all repeating labels and blank labels, where $\boldsymbol{\pi} = (\pi_1, ..., \pi_{T'})$ and $\pi_t \in \{1, ..., K\}$. CTC loss is defined using the probabilities of all CTC paths included in $\Omega(Y)$ as indicated by:

$$\mathcal{L}_{\text{CTC}} = -\log \sum_{\boldsymbol{\pi} \in \Omega(Y)} p(\boldsymbol{\pi}|\boldsymbol{X}) = -\log \sum_{\boldsymbol{\pi} \in \Omega(Y)} \prod_{t=1}^{T'} p(\pi_t|\boldsymbol{x}_t), \quad (18)$$

where the posterior probabilities $p(\pi_t|\boldsymbol{x}_t)$ are calculated by an encoder shared with Transformer-decoder, and a linear layer for CTC branch.

2.3. Joint training and decoding

In this paper, we use Transformer and CTC objectives for training and decoding as in [13]. Our CTC-Transformer architecture and its objectives are summarized in Fig. 1. In the training step, all models are trained so as to minimize both Transformer and CTC losses jointly, see the solid boxes in Fig. 1. The objective function is described as follows:

$$\mathcal{L}_{\text{S2S+CTC}} = (1-\alpha)\,\mathcal{L}_{\text{S2S}} + \alpha \mathcal{L}_{\text{CTC}}, \quad (19)$$

where α represents an interpolation weight that balances the losses. CTC loss helps the attention weights to attain monotonic alignment which results in faster convergence. In the decoding step, we combine the probabilities of Transformer with CTC and LM as in [19–21].

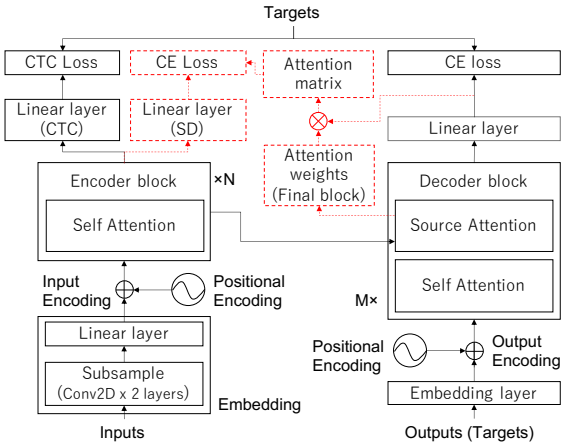

Figure 1: *Schematic diagram of our CTC-Transformer model trained with the SD loss as our proposal (red dashed boxes).*

3. Self-Distillation (SD)

In this paper, we propose a new training objective that can advance joint training so as to improve the performance of CTC-Transformer architecture. Our proposed objective is indicated by the red dashed boxes in Fig. 1. Our proposal creates matrix $\boldsymbol{A} \in \mathbb{R}^{K \times T'}$, called "attention matrix"; it is similar to the posteriorgram in frame-by-frame loss computation. The attention matrix is computed from each head h as follows:

$$\boldsymbol{A}^{(h)} = \sum_{l=1}^{L} \boldsymbol{A}_l^{(h)} = \sum_{l=1}^{L} \boldsymbol{r}_l \boldsymbol{p}_l^{(h)\mathsf{T}}, \tag{20}$$

where $\boldsymbol{r}_l \in \mathbb{R}^K$ is the reference vector of the l-th token. $\boldsymbol{p}_l^{(h)} \in \mathbb{R}^{T'}$ indicates a part of h-th head attention weight $\boldsymbol{P}^{(h)}$, defined in Eq. (1), and corresponds to the attention weight of the l-th token. In this paper, we use the prediction $\hat{\boldsymbol{y}}_l$ in Eq. (16) as the reference vector[1], and use the source-attention weights in the final decoder block. Thus, we obtain as many attention matrices as there are attention heads. Each attention matrix is then normalized along the target class axis k, as follows:

$$a'^{(h)}_{k,t} = \exp(a^{(h)}_{k,t}) / \sum_{k=1}^{K} \exp(a^{(h)}_{k,t}), \tag{21}$$

where $a^{(h)}_{k,t}$ and $a'^{(h)}_{k,t}$ indicate matrix elements of the attention matrix $\boldsymbol{A}^{(h)}$ and its normalized matrix $\boldsymbol{A}'^{(h)}$, respectively. We propose to introduce an additional loss term that consists of the CE loss between the normalized attention matrices A' and the output of the encoder. The loss, called SD, is defined as follows:

$$\mathcal{L}_{\text{SD}} = -\sum_{h=1}^{d^h} \sum_{t=1}^{T'} \sum_{k=1}^{K} a'^{(h)}_{k,t} \log o_{k,t}, \tag{22}$$

where $o_{k,t}$ indicates matrix elements of SD branch outputs. The idea of Eq. (22) is motivated by multitask learning and knowledge distillation using multiple teachers [22–25]. Then, we combine $\mathcal{L}_{\text{SD}}$ with $\mathcal{L}_{\text{S2S}}$ and $\mathcal{L}_{\text{CTC}}$, and the loss term of Eq. (19) is modified as follows:

$$\mathcal{L}_{\text{S2S+CTC+SD}} = (1-\alpha-\beta)\,\mathcal{L}_{\text{S2S}} + \alpha \mathcal{L}_{\text{CTC}} + \beta \mathcal{L}_{\text{SD}}, \tag{23}$$

$$\beta = \gamma \mathcal{A}_{\text{S2S}}\left(\hat{Y}, Y\right), \tag{24}$$

where β is also an interpolation weight that balances the losses. In this paper, β is set as token accuracy with tunable parameter γ. The accuracy of Transformer-decoder is obtained from the function "$\mathcal{A}_{\text{S2S}}(\cdot)$" that treats hypothesis $\hat{Y}$ and ground-truth sequence Y; it is calculated on-the-fly from the training dataset. $\hat{Y}$ is a sequence of the maximum arguments of each prediction, $\hat{\boldsymbol{y}}_{1:L}$ in Eq. (16), without beam search. The range of "$\mathcal{A}_{\text{S2S}}(\cdot)$" is from 0 to 1. At the beginning of training, the attention mechanism may not be very accurate and β is small. This prevents training from being degraded by unreliable attention matrices. As training progresses, the S2S branch becomes more accurate, and the SD loss term becomes more important, and gradually helps to train the shared encoder. Note that the branch output layer for SD loss is thrown away in the decoding step.

Attention matrices $\boldsymbol{A}$ are composed of Transformer outputs and the attention weights of each head; their information includes both posterior and the timing of where to look with regard to the time axis. We expect that a shared-encoder trained with $\mathcal{L}_{\text{SD}}$ will attain more informative representations and better suit Transformer-decoder.

4. Experiments

4.1. Data

We evaluated our proposal on five speech recognition tasks; WSJ, Switchboard, Librispeech, CSJ and NTT Japanese voice search dataset. The datasets had amounts of 81, 260, 960, 581 and 1000 hours, respectively. NTT Japanese voice search task has three test sets and each set consists of 1 hour of speech data. In this paper, we adopt 51 characters, 2000 subwords, 5000 subwords, 3262 characters and 3500 characters as the output tokens for WSJ, Switchboard, Librispeech, CSJ and NTT Japanese voice search dataset, respectively. Speed perturbation (x3) [26] was only applied to CSJ as in the Kaldi chain [27–29] and ESPnet setups [30]. Each subword unit was determined by using the SentencePiece method [31]. The training and evaluation data were preprocessed following the Kaldi and ESPnet toolkit. All models were trained by using the PyTorch toolkit [32].

4.2. System configuration

The input feature was a 80-dimensional log Mel-filterbank with 3 extra features appended for pitch information with first and second-order derivatives. We adopted $N = 12$, $M = 6$, $d^{\text{model}} = 256$, $d^h = 4$, and $d^{\text{ff}} = 2048$ for Transformer with two-layer convolutional neural networks (CNNs). Each CNN layer consisted of a stride of size 2, a kernel size of 3×3, and a ReLU activation function. All network parameters were initialized with random values, and trained for 100 epochs except for WSJ. To further improve performance on all tasks except for WSJ, we used "Large" model setups [14, 33]; $d^{\text{model}} = 512$, $d^h = 8$ and 120 epochs training instead. The parameters of Adam optimizer [34] were set to $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$ with 25000 warmup steps [9]. The initial learning rate was set to 10.0 for WSJ, Switchboard, Librispeech with Large setup, and 5.0 for others. The drop-out rate [35] and threshold of gradient clipping were 0.1 and 5.0, respectively. Moreover, SpecAugment and label smoothing with a penalty of 0.1 were used [36, 37]. Gradient accumulation [38] was also used. For joint training of CTC-Transformer and our proposal, we set α and γ to 0.3 and 0.1, respectively. We used only the early stopping strategy with a validation set and did not apply SpecAugment in the WSJ experiments[2].

[1] Preliminary experiments tried one-hot vector of the ground-truth label, but the results were slightly worse than when using the prediction.

[2] Overall performance on WSJ task was degraded when training full epochs and using SpecAugment. This setup follows that of ESPnet.

Table 1: *Comparisons of the number of heads d^h for SD loss computation (Eq. (22)) on WSJ task. Note that all models have four attention heads.*

Loss	d^h	dev93	eval92
S2S+CTC	-	7.9	4.9
+SD	1	7.3	4.5
	2	7.3	4.7
	3	7.0	4.5
	4	**6.9**	**4.2**

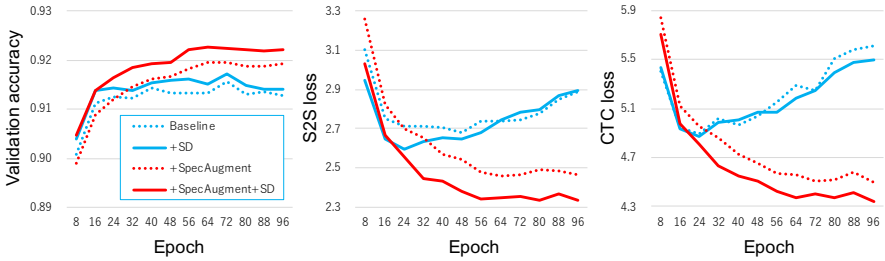

Figure 2: *Accuracy, S2S and CTC losses on NTT voice search development set. Blue and red lines mean the use of SpecAugment or not, respectively. Each plot is for 8 epochs.*

Table 2: *WERs of all systems on English ASR tasks.*

System	Switchboard Hub5'00		Librispeech dev		Librispeech test	
	SWB	CH	clean	other	clean	other
S2S+CTC	11.1	21.1	4.6	13.0	4.9	12.9
+SD	10.5	20.5	4.6	12.4	4.8	12.5
+SpecAug.	9.1	18.0	4.3	10.7	**4.4**	10.8
+SD	**8.9**	**17.3**	**4.1**	**10.0**	**4.4**	**10.0**
+LM	8.3	16.9	2.4	6.0	2.7	6.3
+Large	7.9	15.7	2.1	5.3	2.4	5.6

Table 3: *CERs of all systems on Japanese ASR tasks.*

System	CSJ			NTT voice search		
	E1	E2	E3	E1	E2	E3
HMM+NN-LM [29]	5.7	4.9	4.5	-	-	-
E2E+NN-LM [14]	5.7	4.1	4.5	-	-	-
S2S+CTC	6.3	4.3	4.6	3.2	4.6	5.8
+SD	6.2	4.1	4.4	3.3	4.1	5.4
+SpecAug.	**5.1**	3.8	4.2	2.7	3.6	5.0
+SD	**5.1**	**3.6**	**3.9**	**2.3**	**3.3**	**4.2**
+T-LM	**4.7**	**3.4**	**3.9**	2.0	3.0	3.9
+Large	**4.6**	**3.4**	**3.7**	2.0	3.1	3.7

Neural network-based LM for the shallow fusion [39] was also used. The LMs consisted of one-layer unidirectional long short-term memory (LSTM) with 1000 cells for WSJ, two-layer LSTMs with 650 cells for Switchboard[3], and Transformer-encoder ($N = 16$, $d^{\text{model}} = 512$, $d^h = 8$, $d^{\text{ff}} = 2048$ and 128 dimension linear embedding layer replaced with speech embedding block) and a linear output layer, called T-LM, for others. The numbers of each LM output corresponded to the Transformer-decoder outputs except for WSJ, which used 65k units as word-level entries.

In joint decoding, we set beam width and CTC weight to 20 and 0.3, respectively. The LM weight was set to 1.0 for WSJ, and 0.3 for the others when we used the LMs except for Librispeech, which set beam width, CTC and LM weights to 60, 0.4 and 0.6. We evaluated performance in terms of word error rate (WER) for English tasks, and CER for the Japanese tasks due to the ambiguity of Japanese word boundaries.

4.3. Results

First, we investigated the effectiveness of the number of heads for SD loss computation on the WSJ task; the results are shown in Table 1. LSTM-LM was used for all results. The CTC-Transformer (S2S+CTC) was regarded as the baseline in our experiments. We can see that the WERs of S2S+CTC with SD (our proposal) were better than those of the baseline. The model where we set $d^h = 4$ in Eq. (22) achieved the best performance. The relative WER improvements over the strong baseline system were 12.5%/14.2% on dev/test set, respectively. The greater the number of attention heads, the slightly better the performance. Hereafter we set $d^h = 4$ for SD loss computation.

We also evaluated our proposal SD on public English tasks; the results are shown in Table 2. SpecAugment was also applied to the remaining experiments. We can see that the model trained with SD could achieve better WERs than the baseline. Moreover, this tendency was also observed in the results of the models trained with SpecAugment. The best systems were further improved by using LMs and "Large" setup.

We also evaluated our proposal on public and private Japanese datasets. The results are shown in Table. 3. We can see that the model trained with SD could also achieve better CERs than the baseline on both tasks. Moreover, the trends in the SpecAugment experiments were the same those in the English ASR experiments. These results demonstrate that the models trained with our proposal are superior to the baselines in spite of the strong setups used, i.e. CTC-Transformer and SpecAugment. The best systems were also further improved by using T-LMs and "Large" setup. In particular, the best system in CSJ outperforms the previous state-of-the-art [14] by 18.2%.

Finally, we analyzed the effectiveness of our proposal; the accuracy and S2S- and CTC-loss curves on the validation set of NTT voice search dataset are illustrated in Fig. 2. Each plot is calculated for 8 epochs. We can see that the accuracies were higher than the systems without SD. Moreover, both the losses of the models trained with SD and SpecAugment were lower than the systems without SD. We argue that it means our proposal is also helpful for improving shared-encoder and Transformer-decoder performance.

5. Conclusions

We have proposed a new training approach for S2S+CTC. Our proposal utilizes attention weights and Transformer outputs for making as many pseudo-targets as there are attention heads. The targets, called attention matrices, carry the information of both language and output timing, i.e. where to look with regard to the time axis. They are used in CE loss computation together with the encoder outputs via a branch linear layer. Our loss term, called SD, is combined with S2S and CTC losses, and jointly optimized. We evaluated and compared our proposal using CTC-Transformer as the strong baseline on E2E-ASR experiments. We confirmed using five different datasets that cover various styles, speaking styles and languages, that our proposed loss term yielded consistently improved performance over strong CTC-transformer baselines. Moreover, our best system on CSJ achieved a 3.9% CER on the test set, which outperforms the previous state-of-the-art by 18.2%. Future work includes improving SD loss derived from attention matrices where the alignments are not diagonal and monotonic.

[3]In Switchboard experiment, the performance of our CTC-Transformer with LSTM-LM was slightly better than that with T-LM.

6. References

[1] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification : Labelling unsegmented sequence data with recurrent neural networks," in *Proc. of ICML*, 2006, pp. 369–376.

[2] A. Graves and N. Jaitly, "Towards End-To-End speech recognition with recurrent neural networks," in *Proc. of ICLR*, 2014, pp. 1764–1772.

[3] A. Graves, "Sequence transduction with recurrent neural networks," in *Proc. of ICML*, 2012.

[4] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. of ICASSP*, 2013, pp. 6645–6649.

[5] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio, "End-to-end continuous speech recognition using attention-based recurrent NN: first results," in *Advances in NIPS*, 2014.

[6] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Advances in NIPS*, 2015, pp. 577–585.

[7] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. of ICASSP*, 2016, pp. 4960–4964.

[8] H. Sak, M. Shannon, K. Rao, and F. Beaufays, "Recurrent neural aligner: An encoder-decoder neural network model for sequence to sequence mapping," in *Proc. of INTERSPEECH*, 2017, pp. 1298–1302.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in NIPS*, 2017, pp. 5998–6008.

[10] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-Attention based end-to-end speech recognition using multi-task learning," *Proc. of ICASSP*, pp. 4835–4839, 2017.

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

[12] T. Moriya, H. Sato, T. Tanaka, T. Ashihara, R. Masumura, and Y. Shinohara, "Distilling attention weights for CTC-based ASR systems," in *Proc. of ICASSP*, 2020, pp. 6894 – 6898.

[13] S. Karita, N. Yalta, S. Watanabe, M. Delcroix, A. Ogawa, and T. Nakatani, "Improving transformer-based end-to-end speech recognition with connectionist temporal classification and language model integration," in *Proc. of INTERSPEECH*, 2019, pp. 1408–1412.

[14] S. Karita, X. Wang, S. Watanabe, T. Yoshimura, W. Zhang, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. Yalta, and R. Yamamoto, "A comparative study on transformer vs rnn in speech applications," in *Proc. of ASRU*, 2019, pp. 449–456.

[15] D. B. Paul and J. M. Baker, "The design for the Wall Street Journal-based CSR corpus," in *Proc. of the Workshop on Speech and Natural Language*, 1992, p. 357–362.

[16] J. Godfrey, E. Holliman, and J. McDaniel, "SWITCHBOARD: telephone speech corpus for research and development . acoustics,," in *Proc. of ICASSP*, 1992, pp. 517–520.

[17] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. of ICASSP*, 2015, pp. 5206–5210.

[18] K. Maekawa, H. Koiso, S. Furui, and H. Isahara, "Spontaneous speech corpus of Japanese." in *Proc. of LREC*, 2000, pp. 947–9520.

[19] 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 *Proc. of INTERSPEECH*, 2017, pp. 949–953.

[20] T. Hori, S. Watanabe, and J. R. Hershey, "Multi-level language modeling and decoding for open vocabulary end-to-end speech recognition," in *Proc. of ASRU*, 2017, pp. 287–293.

[21] T. Hori, J. Cho, and S. Watanabe, "End-to-end speech recognition with word-based RNN language models," in *Proc. of SLT*, 2018, pp. 389–396.

[22] J. Li, R. Zhao, J.-T. Huang, and Y. Gong, "Learning small-size dnn with output-distribution-based criteria," *Proc. of INTERSPEECH*, pp. 1910–1914, 2014.

[23] J. Wong and M. Gales, "Sequence student-teacher training of deep neural networks," in *Proc. of INTERSPEECH*, 2016, pp. 2761–2765.

[24] T. Fukuda, M. Suzuki, G. Kurata, S. Thomas, J. Cui, and B. Ramabhadran, "Efficient knowledge distillation from an ensemble of teachers," in *Proc. of INTERSPEECH*, 2017, pp. 3697–3701.

[25] T. Moriya, S. Ueno, Y. Shinohara, M. Delcroix, Y. Yamaguchi, and Y. Aono, "Multi-task learning with augmentation strategy for acoustic-to-word attention-based encoder-decoder speech recognition," in *Proc. of INTERSPEECH*, 2018, pp. 2399–2403.

[26] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition." in *Proc. of INTERSPEECH*, 2015, pp. 3586–3589.

[27] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlıcek, Y. Qian, P. Schwarz, J. Silovskı, G. Stemmer, and K. Veselı, "The Kaldi speech recognition toolkit," in *Proc. ASRU*, 2011.

[28] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for ASR based on lattice-free MMI," in *Proc. of INTERSPEECH*, 2016, pp. 2751–2755.

[29] N. Kanda, Y. Fujita, and K. Nagamatsu, "Lattice-free state-level minimum bayes risk training of acoustic models," in *Proc. of INTERSPEECH*, 2018, pp. 2923–2927.

[30] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "ESPnet: End-to-end speech processing toolkit," in *Proc. of INTERSPEECH*, 2018, pp. 2207–2211.

[31] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proc. of EMNLP: System Demonstrations*, 2018, pp. 66–71.

[32] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS-W*, 2017.

[33] N. Moritz, T. Hori, and J. Le Roux, "Streaming automatic speech recognition with the Transformer model," in *Proc. of ICASSP*, 2020, pp. 6069–6073.

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

[35] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, pp. 1929–1958, 2014.

[36] D. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. Cubuk, and Q. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," 2019, pp. 2613–2617.

[37] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. of CVPR*, 2015, pp. 2818–2826.

[38] M. Ott, S. Edunov, D. Grangier, and M. Auli, "Scaling neural machine translation," in *Proc. of the Third Conference on Machine Translation: Research Papers*, 2018, pp. 1–9.

[39] A. Kannan, Y. Wu, P. Nguyen, T. Sainath, Z. Chen, and R. Prabhavalkar, "An analysis of incorporating an external language model into a sequence-to-sequence model," in *Proc. of ICASSP*, 2018, pp. 1–5828.