HYBRID RNN-T/ATTENTION-BASED STREAMING ASR WITH TRIGGERED CHUNKWISE ATTENTION AND DUAL INTERNAL LANGUAGE MODEL INTEGRATION

Takafumi Moriya[1,2], *Takanori Ashihara*[1], *Atsushi Ando*[1], *Hiroshi Sato*[1], *Tomohiro Tanaka*[1], *Kohei Matsuura*[1], *Ryo Masumura*[1], *Marc Delcroix*[1], *Takahiro Shinozaki*[2]

[1]NTT Corporation, Japan; [2]Tokyo Institute of Technology, Japan

ABSTRACT

In this paper we propose improvements to our recently proposed hybrid RNN-T/Attention architecture that includes a shared encoder followed by recurrent neural network-transducer (RNN-T) and triggered attention-based decoders (TAD). The use of triggered attention enables the attention-based decoder (AD) to operate in a streaming manner. When a trigger point is detected by RNN-T, TAD uses the context from the start-of-speech up to that trigger point to compute the attention weights. Consequently, the computation costs and the memory consumptions are quadratically increased with the duration of the utterances because all input features must be stored and used to re-compute the attention weights. In this paper, we use a short context from a few frames prior to each trigger point for attention weight computation resulting in reduced computation and memory costs. We call the proposed framework triggered chunkwise AD (TCAD). We also investigate the effectiveness of internal language model (ILM) estimation approach using both ILMs of RNN-T and TCAD heads for improving RNN-T performance. We confirm in experiments with public and private datasets covering various scenarios that TCAD achieves superior recognition performance while reducing computation costs compared to TAD.

Index Terms— speech recognition, neural network, end-to-end, recurrent neural network-transducer, attention-based decoder

1. INTRODUCTION

End-to-end automatic speech recognition (E2E-ASR), which can directly map acoustic features to output tokens, has recently attracted increased interest from the ASR research community. In particular, the recurrent neural network-transducer (RNN-T) [1] is a promising technology for streaming ASR applications, and has been reported in many recent studies [2–12]. The attention-based encoder-decoder [13] has also been a success in the ASR community, and can be combined with other E2E-ASR systems, e.g. connectionist temporal classification (CTC) [14] and RNN-T, and such hybrid E2E-ASR systems have shown to improve ASR performance over stand-alone systems [2, 12, 15].

Various RNN-T models have been explored to address the streaming nature [5, 6, 8, 9, 11, 12]. In previous work [12], we integrated RNN-T with the attention-based decoder (AD) for improving the RNN-T performance. Since the attention mechanism uses arbitrarily long past and future contexts during decoding, we adopted the triggered attention mechanism [16] to the AD, called TAD, for greater efficiency. TAD can compute the attention weights using the context from start-of-speech to each trigger point derived from the blank probabilities of RNN-T outputs, and thus TAD can decode in streaming manner whenever a trigger is detected. In [12], we reported that TAD improved RNN-T performance. However, the computation costs and the memory consumptions of TAD increase quadratically with utterance duration because all past input features must be used and stored for attention weight re-computation. Moreover, we did not investigate RNN-T/TAD with recently proposed language model integration approaches [3, 7, 8, 10, 17, 18].

In this paper, we propose two modifications to TAD to address the above limitations. First, we modify TAD so that long past input features can be discarded in order to reduce both computation and memory costs. We realize this by computing the attention weights over short windows, an idea that we borrow from monotonic chunkwise attention (MoChA) [19]. More precisely, we propose triggered chunkwise AD (TCAD), which computes the attention weights over a short context window that starts a few frames before each trigger point. We expect that historical context is not essential for attention weight computation and thus TCAD can discard input features before a few past frames while maintaining recognition performance. Moreover, because we use a fixed context window length, the computation complexity and memory of TCAD does not increase with utterance duration.

In addition to the above reduction of computation and memory costs, we investigate the introduction of internal language model (ILM) [7–9] within the TCAD framework to improve recognition performance. ILM estimation is known to be successful in the ASR community [7, 8]; it subtracts the score of a pseudo LM, which mimics the internal LM of RNN-T or AD, from the score of external LM trained on large amount of text data so as to maximize the LM effect. ILM has been investigated for stand-alone RNN-T or attention-based models. Here, we use a hybrid model, which combines two decoder heads: RNN-T and TCAD. We thus extend the conventional ILM scheme with this hybrid model approach and investigate the effectiveness of dual ILM estimation.

We evaluate our proposed TCAD on five datasets for several widely-used encoder configurations such as long short-term memory (LSTM) and Conformer [20]. The results demonstrate that our proposal yields the same ASR performance as TAD while reducing the computation costs significantly. Moreover, we also show that ILM estimation using both ILMs, RNN-T and TCAD, is effective.

2. ASR MODELS

Let us first explain RNN-T and AD, which are the foundations of this work. Let $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_{T'}]$ be the input acoustic feature sequence, and $Y = [y_1, ..., y_U]$ be a token sequence and $y_u \in \{1, ..., K\}$. K is the number of tokens including special symbols, i.e. "blank" label, ϕ, for RNN-T, and "end-of-sentence" label for AD.

2.1. Recurrent neural network-transducer (RNN-T)

RNN-T learns the mapping between sequences of different lengths. By introducing the prediction network, RNN-T can consider the posterior probabilities to be jointly conditioned on not only encoder outputs but also previous predictions as shown in the following steps. First, $\boldsymbol{X}$ is downsampled and encoded into $\boldsymbol{H}^{\text{enc}} = [\boldsymbol{h}_1^{\text{enc}}, ..., \boldsymbol{h}_T^{\text{enc}}]$ with length-T via encoder network $f^{\text{enc}}(\cdot)$. Next, the tokens, Y, are also encoded into $\boldsymbol{H}^{\text{pred}} = \left[\boldsymbol{h}_1^{\text{pred}}, ..., \boldsymbol{h}_U^{\text{pred}}\right]$ via prediction network $f^{\text{pred}}(\cdot)$. Then, these encoded features are fed to a feed-forward network, $f^{\text{joint}}(\cdot)$. The above operations, which yield prediction $\hat{\boldsymbol{y}}_{t,u}$, are defined as follows:

$$\boldsymbol{h}_t^{\text{enc}} = f^{\text{enc}}(\boldsymbol{x}_{t'}; \theta^{\text{enc}}), \tag{1}$$

$$\boldsymbol{h}_u^{\text{pred}} = f^{\text{pred}}(y_{u-1}; \theta^{\text{pred}}), \tag{2}$$

$$\hat{\boldsymbol{y}}_{t,u} = \text{Softmax}\left(f^{\text{joint}}(\boldsymbol{h}_t^{\text{enc}}, \boldsymbol{h}_u^{\text{pred}}; \theta^{\text{joint}})\right), \tag{3}$$

where Softmax$(\cdot)$ means a softmax layer without learnable parameters. All networks with learnable parameters $\theta^{\text{RNN-T}} \triangleq [\theta^{\text{enc}}, \theta^{\text{pred}}, \theta^{\text{joint}}]$ are optimized by using RNN-T loss $\mathcal{L}_{\text{RNN-T}}$ using the forward-backward algorithm [1].

2.2. Triggered attention-based decoder (TAD)

In [12], we introduced TAD using RNN-T outputs to trigger attention, which enables the AD to operate in streaming mode. Fig. 1 is a schematic diagram of AD, and Fig. 1 Ⅱ indicates the actual attention weights of TAD. The model consists of two branches, one for RNN-T and one for AD. TAD predicts the next label distribution $\hat{\boldsymbol{y}}_u$ using the hidden state (memory) activation $\boldsymbol{m}_u$ of the decoder $f^{\text{dec}}(\cdot)$ at u-th step. $\boldsymbol{m}_u$ is computed as follows:

$$\boldsymbol{m}_u = f^{\text{dec}}(\boldsymbol{m}_{u-1}, \boldsymbol{g}_u, y_{u-1}; \theta^{\text{dec}}), \tag{4}$$

where $\boldsymbol{g}_u$ and y_{u-1} denote the context vector at the u-th step and the target token at the previous step, respectively. $\boldsymbol{g}_u$ is the weighted sum of the encoder output sequence from start-of-speech to trigger point t_u:

$$\boldsymbol{g}_u = \sum_{t=1}^{t_u} a_{t,u} \boldsymbol{h}_t^{\text{enc}}, \tag{5}$$

where t_u denotes the trigger point generated from RNN-T, the time index of the u-th non-blank token occurrence on RNN-T paths. In the training step, RNN-T paths are generated from the forward-backward variables [1], and each trigger point t_u is determined by the argument of the maximum of the paths on the time axis. $a_{t,u}$ indicates the triggered attention weight and is calculated as:

$$a_{t,u} = f^{\text{att}}\left(\{a_{t,u-1}\}_{t=1}^{t_u}, \boldsymbol{m}_{u-1}, \boldsymbol{h}_t^{\text{enc}}; \theta^{\text{att}}\right), \tag{6}$$

where $f^{\text{att}}(\cdot)$ computes attention weights; location-aware attention is used in this paper. Note that I in Fig. 1 indicates the case of AD using global context (GAD) when t_u is always T in Eq. (5) and (6). Using $\boldsymbol{m}_u$, TAD predicts $\hat{\boldsymbol{y}}_u$ as:

$$\hat{\boldsymbol{y}}_u = \text{Softmax}\left(f^{\text{out}}(\boldsymbol{m}_u; \theta^{\text{out}})\right), \tag{7}$$

where $f^{\text{out}}(\cdot)$ denotes the output layer. The learnable parameters θ^{dec}, θ^{att} and θ^{out} are optimized by the cross entropy (CE) loss.

2.2.1. Triggered chunkwise attention-based decoder (TCAD)

TAD is an efficient way to enable encoder-decoder models to operate in streaming manner. However, the computation and memory costs of TAD increase quadratically with utterance duration because all past features from 1 to t_u must be cumulatively stored in memory, and used for attention weight computation. To solve this problem, we propose triggered chunkwise AD (TCAD); it uses a fixed short

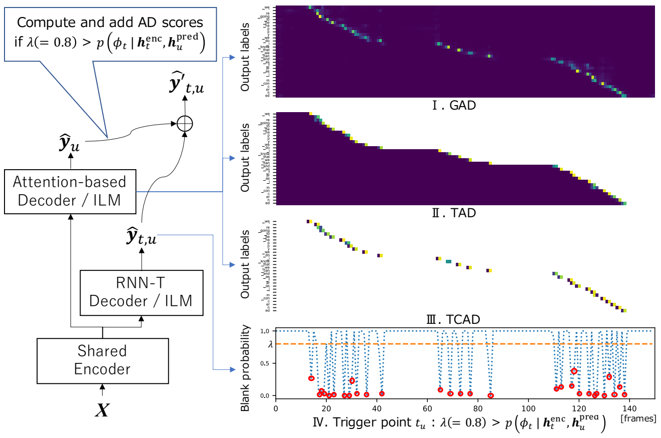
Compute and add AD scores
if $\lambda(=0.8) > p\left(\phi_t \mid \boldsymbol{h}_t^{\text{enc}}, \boldsymbol{h}_u^{\text{pred}}\right)$
$\hat{\boldsymbol{y}}'_{t,u}$
$\hat{\boldsymbol{y}}_u$
Attention-based Decoder / ILM
$\hat{\boldsymbol{y}}_{t,u}$
RNN-T Decoder / ILM
Shared Encoder
$\boldsymbol{X}$
Output labels
I . GAD
Output labels
Ⅱ. TAD
Output labels
Ⅲ. TCAD
Blank probability
λ
Ⅳ. Trigger point t_u : $\lambda(=0.8) > p\left(\phi_t \mid \boldsymbol{h}_t^{\text{enc}}, \boldsymbol{h}_u^{\text{pred}}\right)$
[frames]

Fig. 1. *Schematic diagram of model architecture. Each picture depicts the attention weights of each AD (I - Ⅲ). The colored part of I - Ⅲ represents the valid attention weights. Ⅳ represents trigger points (red circles) in the decoding step for TED-LIUM2 test set.*

window from $t_u - \varepsilon$ to t_u for attention weight computation. Fig. 1 Ⅲ shows an example of attention weight computation with TCAD. We modify Eq. (5) and (6) for TCAD as follows:

$$\boldsymbol{g}_u = \sum_{t=t_u-\varepsilon}^{t_u} a_{t,u} \boldsymbol{h}_t^{\text{enc}}, \tag{8}$$

$$a_{t,u} = f^{\text{att}}\left(\boldsymbol{m}_{u-1}, \boldsymbol{h}_t^{\text{enc}}; \theta^{\text{att}}\right), \tag{9}$$

where ε is the lookback hyperparameter, which we set to a very small value, i.e. 1 in this work. This means that we adopt zero-lookahead and one-frame lookback in this work. Note that we replace location-aware attention with additive attention due to the very short context used. Therefore, we can discard long past features before $t_u - \varepsilon$, which reduces the computation and memory costs. Moreover, trigger points t_u can be regarded as RNN-T knowledge of where to attend the position, and permit efficiently utilization in TAD/TCAD training. We expect that TCAD will decode faster than TAD while keeping the recognition performance.

2.3. Internal language model (ILM)

In this work, we also investigate the effectiveness of ILMs. Each decoder part of RNN-T and AD can be regarded as a pseudo LM module, called ILM. Our ILM implementations modify Eq. (2) and (3) for RNN-T, and Eq. (4) and (7) for AD. RNN-T ILM output $\hat{\boldsymbol{y}}_u^{\text{ILM(RNN-T)}}$ and AD ILM output $\hat{\boldsymbol{y}}_u^{\text{ILM(AD)}}$ are defined as follows:

$$\hat{\boldsymbol{y}}_u^{\text{ILM(RNN-T)}} = \text{Softmax}\left(f^{\text{joint}}(f^{\text{pred}}(y_{u-1}; \theta^{\text{pred}}); \theta^{\text{joint}})\right), \tag{10}$$

$$\hat{\boldsymbol{y}}_u^{\text{ILM(AD)}} = \text{Softmax}\left(f^{\text{out}}(f^{\text{dec}}(\boldsymbol{m}_{u-1}, y_{u-1}; \theta^{\text{dec}}); \theta^{\text{out}})\right), \tag{11}$$

where each parameter θ^* is shared with each decoder. The decoder inputs with acoustic information, i.e. $\boldsymbol{h}_t^{\text{enc}}$ and $\boldsymbol{g}_u$, are zeroing out. ILMs subtract its score from the score of an external LM (extLM) trained on large amount of text data so as to maximize the LM effect [7–9].

2.4. Model Training and Decoding for RNN-T with AD

Training: In this work, we adopt a multi-step training scheme as in previous work [12], i.e. we first train the RNN-T branch (shared encoder and RNN-T decoder), then freeze its parameters and train the AD branch. When we train the RNN-T branch, we combine the RNN-T loss with a CTC [14] and ILM training [7,9] loss as follows:

$$\mathcal{L} = \mathcal{L}_{\text{RNN-T}} + \alpha \mathcal{L}_{\text{CTC}} + \beta \mathcal{L}_{\text{ILM}}, \tag{12}$$

where α and β are the weights of CTC and ILM losses respectively. $\mathcal{L}_{\text{CTC}}$ is a CTC loss calculated between token sequence and extra linear layer outputs $f^{\text{linear}}(f^{\text{enc}}(\boldsymbol{X};\theta^{\text{enc}});\theta^{\text{linear}})$. Note that CTC extra linear layer θ^{linear} is thrown away during decoding. $\mathcal{L}_{\text{ILM}}$ is CE loss between RNN-T ILM outputs from Eq. (10) and reference tokens as in RNN-LM training. In the second step, we freeze the RNN-T branch, and only train the parameters of the AD branch, i.e. θ^{dec}, θ^{att}, and θ^{out} using CE loss as normally used for AD training.

Decoding: We consider time indices where non-blank tokens are emitted as trigger points (red circles in IV of Fig. 1), which we find as the time indices where the blank probability is smaller than a threshold λ as in [12]. Therefore AD is activated when $p(\phi_t|\boldsymbol{h}_t^{\text{enc}}, \boldsymbol{h}_u^{\text{pred}})$ is less than λ. We also use the caches of AD outputs and its decoder states at each time step as in RNN-T decoding [4]. These can greatly reduce the computation costs.

Moreover our proposed architecture has two decoders that are RNN-T and AD. We also combine the benefits of the dual decoders, and so investigate how using dual ILMs, which are both ILMs in RNN-T and AD, can improve performance. The most probable hypothesis at each trigger step is given by:

$$\begin{aligned}\hat{Y} \quad = \quad & \arg\max_{Y}\{(1-\rho)\log p_{\text{RNN-T}}(Y|\boldsymbol{X}) + \rho \log p_{\text{AD}}(Y|\boldsymbol{X}) \\ + \quad & \mu_1 \log p_{\text{extLM}}(Y) - \mu_2 \log p_{\text{srcLM}}(Y) \\ - \quad & \mu_3 \log p_{\text{ILM(RNN-T)}}(Y) - \mu_4 \log p_{\text{ILM(AD)}}(Y)\}, \qquad (13)\end{aligned}$$

where ρ, μ_* are tunable parameters. $p_{\text{RNN-T}}(Y|\boldsymbol{X})$, $p_{\text{AD}}(Y|\boldsymbol{X})$, $p_{\text{ILM(RNN-T)}}(Y)$ and $p_{\text{ILM(AD)}}(Y)$ can be computed from Eq. (3), (7), (10), and (11), respectively. μ_1, μ_2, μ_3, and μ_4 indicate the weights that balance each LM score. $p_{\text{extLM}}(Y)$ and $p_{\text{srcLM}}(Y)$ are the probabilities of extLM and source domain LM (srcLM) that is trained on transcribed texts. Note that Eq. (13) becomes LM shallow fusion (SF) [17] if $\mu_1 > 0$ and $\mu_2 = \mu_3 = \mu_4 = 0$, density ratio fusion [3] if $\mu_1, \mu_2 > 0$ and $\mu_3 = \mu_4 = 0$, and ILM estimation [7,8] if $\mu_1, \mu_3, \mu_4 > 0$ and $\mu_2 = 0$. The AD and all LM scoring are employed on-the-fly during beam search.

3. EXPERIMENTS

3.1. Data

We evaluated our proposal on five ASR tasks; TED-LIUM release-2 (TED-LIUM2) [21], Switchboard [22], Librispeech [23], corpus of spontaneous Japanese (CSJ) [24], and NTT Japanese voice search production dataset (NTTVS). The datasets contain samples totaling 210, 260, 960, 581 and 1000 hours, respectively. The test set of NTTVS task consists of 3 hours of speech data. In this paper, we adopted 500 subwords, 1000 subwords, 1000 subwords, 3262 characters and 3500 characters as the output tokens for TED-LIUM2, Switchboard, Librispeech, CSJ and NTTVS, respectively. Speed perturbation (x3) [25] was applied to all datasets as in the ESPnet setups [26]. The training and evaluation data were preprocessed following the Kaldi and modified ESPnet toolkits [26,27].

3.2. System configuration

The input feature was an 80-dimensional log Mel-filterbank appended with 3 pitch features, and SpecAugment [28] was applied. The minibatch size was set to 64 in all experiments. In this work, we adopted LSTM- and Conformer-based encoders as detailed below.

In LSTM-based experiments, we investigated three different encoder types; unidirectional LSTM (ULSTM), bidirectional LSTM (BLSTM), and latency-controlled BLSTM (LC-BLSTM) [29]. All encoders had four-layer 2D-convolutional neural networks (CNNs) followed by six-layer (LC-) BLSTM with 512 cells per direction.

Table 1. *Comparisons of each AD scoring for error rates and RTFs using full streaming (fs) ULSTM-based encoder setups on TED-LIUM2 (longest) and NTTVS (shortest) test sets. The numbers in parentheses indicate the average length per speech.*

		TED-LIUM2				NTTVS	
		segment (8s)		lecture (17min)		command (2s)	
System	fs	WER	RTF	WER	RTF	CER	RTF
RNN-T	✓	11.4	0.53	14.7	0.53	5.2	0.41
+GAD		11.4	0.66	14.5	9.28	**4.8**	0.55
+TAD	✓	**11.0**	0.62	14.4	6.88	**4.8**	0.53
+TCAD	✓	**11.0**	**0.59**	**14.2**	**0.67**	**4.8**	**0.49**

Table 2. *Comparisons of the effectiveness of TCAD and each LM fusion using various encoder types on TED-LIUM2 test set.*

	Encoder				
System	U	B	LC-B	C	LC-C
RNN-T	11.4	8.9	9.3	7.9	8.6
+extLM	11.1	8.8	9.2	7.6	8.3
++ILM(RNN-T)	11.0	8.5	8.8	7.4	8.1
++srcLM	10.9	8.4	8.8	7.4	8.1
++All LMs	10.8	8.4	8.7	7.3	8.1
+TCAD	11.0	8.7	9.2	7.7	8.4
++extLM	10.9	8.6	9.1	7.5	8.3
+++ILM(RNN-T)	10.8	8.4	8.7	7.3	8.0
+++ILM(TCAD)	10.8	8.3	8.8	7.5	8.2
+++srcLM	10.7	8.3	8.7	7.3	8.0
+++All LMs	**10.5**	**8.2**	**8.5**	**7.2**	**7.9**

U: ULSTM, B: BLSTM, LC-B: LC-BLSTM,
C: Conformer, LC-C: LC-Conformer

The CNNs in the second and last layers had max-pooling layers with two strides. The memory cells were doubled when using the ULSTM encoder. The chunk sizes for LC-BLSTM, N_c and N_r, were set to 60 at the input feature space before CNNs. The average latencies of ULSTM- and LC-BLSTM-based systems were 60ms and 960ms respectively. As the prediction network, we adopted an ULSTM layer with 512 cells. The outputs of the encoder and prediction network were fed to the joint network, which mapped them to 512 dimensions and then classified them to the target tokens of each dataset. ULSTM and BLSTM encoders were first pre-trained [11] using the AdaDelta optimizer [30] with early stopping, and then the parameters were fine-tuned by Eq. (12). In the fine-tuning step, learning rate scheduling was very important, so we used the Adam optimizer [31] with learning rate scheduler [28] that is $(s_r, s_i, s_f) = (4k, 160k, 640k)$.[1] LC-BLSTM models imported BLSTM model parameters for initialization, and then fine-tuned with learning rate scheduler $(s_r, s_i, s_f) = (1k, 80k, 160k)$. α and β were set to 0.1 and 0.4 respectively.

In Conformer-based experiments, we investigated latency-controlled Conformer (LC-Conformer) [6, 32]. The offline and streaming Conformer architecture was the same as Conformer (L) [20] with a kernel size of 15. We also used four-layer 2D-CNNs followed by the Conformer blocks, with the stride sizes of max-pooling layers at second and last layers of 3 and 2, respectively.[2] The prediction network had a 640-dimensional ULSTM layer followed by a 512-dimensional feed-forward network. LC-Conformer models were trained using attention mask, and chunkwise decoding was performed as in [6]. The left and current chunk sizes were set to

[1]We did not use variational weight noise.

[2]In CSJ task, both the stride sizes in max-pooling layers were set to 2, and it was the best configuration. The latency of LC-Conformer was 260ms.

Table 3. *Comparisons of the effectiveness of TCAD and each LM fusion using various encoder types on five datasets. Systems with * correspond to stand-alone RNN-T systems. "Size" means the total number of model parameters.*

		TED-LIUM2		Switchboard			Librispeech			CSJ				NTTVS	
		WER	Size	WER		Size	WER		Size	CER			Size	CER	Size
	System			SWB	CH		clean	other		E1	E2	E3			
Offline	BLSTM*	8.9	39M	8.4	16.8	40M	3.5	9.1	40M	5.4	3.9	4.2	43M	3.2	44M
	+TCAD	8.7	44M	8.2	16.6	45M	3.4	9.0	45M	5.3	3.9	4.2	51M	3.2	51M
	++All LMs	8.2	81M	7.9	15.9	84M	3.1	8.1	84M	5.1	3.9	4.2	91M	3.2	98M
	Conformer*	7.9	115M	7.4	14.8	116M	2.9	7.0	116M	4.5	3.4	3.6	120M	3.2	120M
	+TCAD	7.7	122M	7.2	14.7	123M	2.9	6.8	123M	4.4	3.4	3.6	130M	3.1	130M
	++All LMs	7.2	159M	6.9	14.2	162M	2.6	6.4	162M	4.4	3.3	3.5	170M	2.9	177M
Streaming	ULSTM*	11.4	63M	11.7	22.9	64M	5.2	13.0	64M	7.0	5.2	5.4	68M	5.2	68M
	+TCAD	11.0	70M	11.5	22.5	71M	5.1	12.6	71M	6.9	5.1	5.4	77M	4.8	77M
	++All LMs	10.5	107M	10.8	21.5	109M	4.3	11.3	109M	6.6	5.0	5.4	117M	4.7	124M
	LC-BLSTM*	9.3	39M	8.9	17.4	40M	3.8	10.3	40M	5.7	4.1	4.4	43M	3.4	44M
	+TCAD	9.2	44M	8.8	17.2	45M	3.6	10.2	45M	5.7	4.1	4.4	51M	3.4	51M
	++All LMs	8.5	81M	8.4	16.6	84M	3.4	9.3	84M	5.6	4.0	4.3	91M	3.4	98M
	LC-Conformer*	8.6	115M	8.3	16.6	116M	3.3	8.9	116M	5.5	4.2	4.3	120M	3.3	120M
	+TCAD	8.4	122M	8.1	16.5	123M	3.3	8.7	123M	5.5	4.1	4.2	130M	3.3	130M
	++All LMs	7.9	159M	7.4	15.7	162M	3.0	7.8	162M	5.3	4.2	4.1	170M	3.0	177M

1020ms and 600ms, respectively, and thus the average latency was 380ms ($= 10 \times 60\text{ms}/2 + 80\text{ms}$). All Conformer model parameters were randomly initialized, and trained with Eq. (12) by using the Adam optimizer with 25k warmup for a total of 40 epochs. α and β were set to 0.5 and 0.1, respectively.

In AD training, the LSTM sizes were the same as each LSTM- or Conformer-based transducer decoder. We adopted zero-lookahead (and $\varepsilon = 1$ for TCAD) for triggered attention. The AD parameters were randomly initialized, and trained using the AdaDelta optimizer with early stopping.

LSTM-LMs were also used. The external LMs consisted of a four-layer ULSTM with 1024 cells, and were trained using the same data as the ESPnet recipes.[3] The source LMs, which were trained using only transcribed texts of the speech, had a one-layer ULSTM with 512 cells. The numbers of each LM output corresponded to the transducer-decoder outputs. In the decoding steps, we tuned ρ and μ_* in Eq. (13) by using each development set.[4]

For decoding, we used alignment-length synchronous decoding with beam size of 8 [4]. Threshold λ was set to 0.8 empirically. We evaluated performance in terms of word error rate (WER) for English tasks, as well as character error rate (CER) for Japanese tasks. We measured inference speed in terms of real time factor (RTF: $decoding\ time/data\ time$). The decoding algorithm with Python implementation was executed on an Intel Xeon 2.40GHz CPU.

3.3. Results

First, we investigate the effectiveness of each AD in improving RNN-T performance; the error rates and RTFs are shown in Table 1. We chose the TED-LIUM2 and NTTVS test sets because the average segment lengths were longest (8 seconds) and shortest (2 seconds), respectively, for all datasets. TED-LIUM2 is a lecture type dataset, and we also evaluated each system on an unsegmented version, called "lecture", which has an average length per lecture of 17 minutes. "RNN-T" means the performance using a stand-alone RNN-T decoder. "+GAD", "+TAD", and "+TCAD" indicate the hybrid systems with GAD, TAD, and TCAD. Only model "+GAD" is an offline model. We can see that the error rates of RNN-Ts with TAD/TCAD on segments and command formats are better than RNN-T only. Moreover, TCAD achieved the lowest RTFs among the AD systems, and its effectiveness was prominent on the long-form dataset, i.e. "lecture". Note that GAD did not improve WER much on the TED-LIUM2 task compared to TAD and TCAD. This is because TAD and TCAD could utilize trigger point information as RNN-T knowledge in training and decoding steps, which resulted in faster and better convergence during training. Hereafter we adopt TCAD for combination with each LM.

We investigated the effectiveness of various LM integrations according to Eq. (13) using streaming and offline systems; the WERs on TED-LIUM2 are shown in Table 2. "+extLM" and "+TCAD" mean that only extLM or TCAD were used, respectively. "++*" and "+++*" indicate that each component was incrementally used, and "All LMs" is the case that all the above components were used. We can see that each component is effective in improving RNN-T performance. By using all components, the performance was further improved. Moreover "All LMs" with and without TCAD were compared, and the performance of the system with TCAD enabled was slightly better than the without one. We argue that TCAD is effective for both online and offline models.

Finally, we applied both TCAD and LM combinations to RNN-T on all datasets using various encoders as a comparative study; the error rates and parameter sizes are shown in Table 3. "+TCAD" and "++All LMs" indicate the use of TCAD, and with all LMs, respectively. We can see that both the use of TCAD or the fusion of all LM scores is effective for improving RNN-T performance. By applying all components to each transducer system, all systems achieved the best performance.

4. CONCLUSION

We have proposed a streaming hybrid RNN-T/AD model with TCAD which uses a short context that starts a few frames before each trigger point for computing the attention weight to improve computation efficiency. We also investigated the effectiveness of ILM estimation using the dual ILMs of RNN-T and TCAD to improve ASR performance. These were evaluated on five different datasets, and found to yield consistently superior performance to RNN-T baselines with various encoder types while significantly reducing the computation costs compared to TAD.

[3]As no external language resource of CSJ exists, we trained the four-layer 1024-dimensional ULSTM using only transcribed text so $\mu_2 = 0$.

[4]We heuristically and efficiently found the best combination of ρ and μ_* by grid search under $0.2 \geq \mu_1 - \mu_2 - \mu_3 - \mu_4 \geq 0.0$ when μ_2, μ_3 or μ_4 were valid.

5. REFERENCES

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

[2] T. N. Sainath, R. Pang, D. Rybach, Y. He, R. Prabhavalkar, W. Li, M. Visontai, Q. Liang, T. Strohman, Y. Wu, I. McGraw, and C. Chiu, "Two-pass end-to-end speech recognition," in *Proc. of INTERSPEECH*, 2019, pp. 2773–2777.

[3] E. McDermott, H. Sak, and E. Variani, "A density ratio approach to language model fusion in end-to-end automatic speech recognition," in *Proc. of ASRU*, 2019, pp. 434–441.

[4] G. Saon, Z. Tuske, and K. Audhkhasi, "Alignment-length synchronous decoding for RNN Transducer," in *Proc. of ICASSP*, 2020, pp. 7799–7803.

[5] G. Kurata and G. Saon, "Knowledge distillation from offline to streaming RNN transducer for end-to-end speech recognition," in *Proc. of INERSPEECH*, 2020, pp. 2117–2121.

[6] X. Chen, Y. Wu, Z. Wang, S. Liu, and J. Li, "Developing real-time streaming transformer transducer for speech recognition on large-scale dataset," in *Proc. of ICASSP*, 2021, pp. 5904–5908.

[7] E. Variani, D. Rybach, C. Allauzen, and M. Riley, "Hybrid autoregressive Transducer (HAT)," in *Proc. of ICASSP*, 2020, pp. 6139–6143.

[8] Z. Meng, S. Parthasarathy, E. Sun, Y. Gaur, N. Kanda, L. Lu, X. Chen, R. Zhao, J. Li, and Y. Gong, "Internal language model estimation for domain-adaptive end-to-end speech recognition," in *Proc. of SLT*, 2021, pp. 243–250.

[9] Z. Meng, N. Kanda, Y. Gaur, S. Parthasarathy, E. Sun, L. Lu, X. Chen, J. Li, and Y. Gong, "Internal language model training for domain-adaptive end-to-end speech recognition," in *Proc. of ICASSP*, 2021, pp. 7338–7342.

[10] A. Zeyer, A. Merboldt, W. Michel, R. Schlüter, and H. Ney, "Librispeech Transducer Model with Internal Language Model Prior Correction," in *Proc. of INTERSPEECH*, 2021, pp. 2052–2056.

[11] T. Moriya, T. Ashihara, T. Tanaka, T. Ochiai, H. Sato, A. Ando, Y. Ijima, R. Masumura, and Y. Shinohara, "SimpleFlat: A simple whole-network pre-training approach for RNN transducer-based end-to-end speech recognition," in *Proc. of ICASSP*, 2021, pp. 5664–5668.

[12] T. Moriya, T. Tanaka, T. Ashihara, T. Ochiai, H. Sato, A. Ando, R. Masumura, M. Delcroix, and T. Asami, "Streaming end-to-end speech recognition for Hybrid RNN-T/Attention architecture," in *Proc. of INTERSPEECH*, 2021, pp. 1787–1791.

[13] 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.

[14] 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.

[15] 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.

[16] N. Moritz, T. Hori, and J. L. Roux, "Triggered attention for end-to-end speech recognition," in *Proc. of ICASSP*, 2019, pp. 5666–5670.

[17] 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. 5824–5828.

[18] M. Zeineldeen, A. Glushko, W. Michel, A. Zeyer, R. Schlüter, and H. Ney, "Investigating Methods to Improve Language Model Integration for Attention-Based Encoder-Decoder ASR Models," in *Proc. of INTERSPEECH*, 2021, pp. 2856–2860.

[19] C.-C. Chiu* and C. Raffel*, "Monotonic chunkwise attention," in *Proc. of ICLR*, 2018.

[20] A. Gulati, J. Qin, C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. of INTERSPEECH*, 2020, pp. 5036–5040.

[21] A. Rousseau, P. Deléglise, and Y. Estève, "TED-LIUM: an automatic speech recognition dedicated corpus," in *Proc. of LREC*, 2012, pp. 125–129.

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

[23] 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.

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

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

[26] 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.

[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. of ASRU*, 2011.

[28] 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," in *Proc. of INTERSPEECH*, 2019, pp. 2613–2617.

[29] Y. Zhang, G. Chen, D. Yu, K. Yao, S. Khudanpur, and J. R. Glass, "Highway long short-term memory RNNs for distant speech recognition," in *Proc. of ICASSP*, 2016, pp. 5755–5759.

[30] M. D. Zeiler, "Adadelta: An adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012.

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

[32] H. Inaguma and T. Kawahara, "VAD-Free Streaming Hybrid CTC/Attention ASR for Unsegmented Recording," in *Proc. of INTERSPEECH*, 2021, pp. 4049–4053.