INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

4D ASR: Joint modeling of CTC, Attention, Transducer, and Mask-Predict decoders

Yui Sudo[1], *Muhammad Shakeel*[1], *Brian Yan*[2], *Jiatong Shi*[2], *Shinji Watanabe*[2]

[1]Honda Research Institute Japan Co., Ltd., Saitama, Japan
[2]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA

{yui.sudo, shakeel.muhammad}@jp.honda-ri.com,
{byan, jiatongs}@andrew.cmu.edu, shinjiw@ieee.org

Abstract

End-to-end (E2E) automatic speech recognition (ASR) can be classified into several models, including connectionist temporal classification (CTC), recurrent neural network transducer (RNN-T), attention mechanism, and mask-predict models. There are pros and cons to each of these architectures, and thus practitioners may switch between these different models depending on application requirements. Instead of building separate models, we propose a joint modeling scheme where four different decoders (CTC, attention, RNN-T, mask-predict) share an encoder – we refer to this as 4D modeling. Additionally, we propose to 1) train 4D models using a two-stage strategy which stabilizes multitask learning and 2) decode 4D models using a novel time-synchronous one-pass beam search. We demonstrate that jointly trained 4D models improve the performances of each individual decoder. Further, we show that our joint CTC/RNN-T/attention decoding surpasses the previously proposed CTC/attention decoding.

Index Terms: speech recognition, CTC, attention, RNN-T, non-autoregressive

1. Introduction

End-to-end (E2E) automatic speech recognition (ASR) has been actively studied. E2E ASR systems include four main network architectures, such as connectionist temporal classification (CTC) [1–3], recurrent neural network transducer (RNN-T) [4–8], attention mechanism [9–12], and non-autoregressive (NAR) methods [13–15]. These networks align speech signals and token sequences in various ways, each with its own strengths and weaknesses, as follows:

- CTC assumes conditional independence and predicts monotonic alignment of output tokens with input frames. It is fast and suitable for real-time applications, but its performance may suffer from the conditional independence assumption. CTC can also be used to segment long recordings [16].
- RNN-T also has a monotonic alignment assumption, but unlike CTC, it relaxes the conditional independence assumption. It generally outperforms CTC and is suitable for streaming ASR [7]. However, the modeling space is larger than that of the CTC, making it more difficult to train.
- The attention model includes a source-target attention mechanism that aligns speech signals with token sequences. This mechanism is extremely useful in tasks requiring flexible alignments between input and output sequences (e.g., translation tasks) [17, 18]. However, it is prone to alignment errors in ASR tasks because it lacks the monotonicity constraint.
- A typical example of NAR models is Mask-CTC [13]. Mask-CTC estimates the token sequence using the entire sequence.

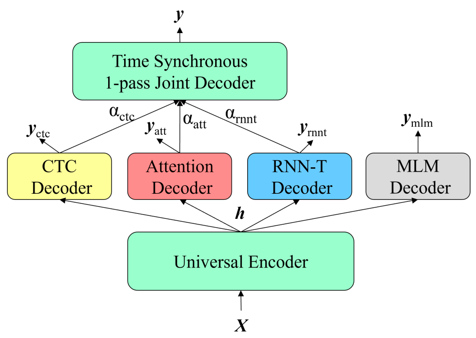


Figure 1: *4D joint model of CTC, attention, RNN-T, and Mask-CTC: the four decoders share a universal encoder and joint decoding is performed using CTC, attention, and RNN-T.*

The mask-based approach [19] can consider label dependency and still retain fast latency. It can be also used for two-pass rescoring approaches as ASR error correction [20, 21].

Since each network architecture has the above different properties, separate models for each application scenario are usually required. CTC, for example, is appropriate for on-device systems requiring less computation, whereas attention is suitable for offline systems with less stringent latency requirements. However, having multiple models for each application scenario increases overheads. Several attempts have been made to integrate these models, such as CTC/attention, to reduce their respective shortcomings [22–24]. Multitask learning is performed by sharing an encoder to regularize the overflexibility of the attention mechanism and the conditional independence assumption of the CTC model. A one-pass beam search is performed during decoding using both CTC and attention decoders to further improve performance [25]. Other integrated models based on two-pass decoding of RNN-T/attention, NAR/attention, and NAR/RNN-T models have also been proposed [21, 26–29]. Given the success of these joint models with two types of decoders, a natural question is *how many decoders can be successfully integrated in a single model?*

In this work, we seek to jointly model four decoders (4D) with a shared encoder (Figure 1): CTC, attention, RNN-T, and Mask-CTC. To accommodate the increased modeling complexity under this 4D scheme, we adapt both our training and decoding strategies. In particular, we:

- Employ a two-stage optimization strategy to select multitasking hyper-parameters in an efficient, data-driven manner.
- Demonstrate experimentally that each single-decoder branch of 4D models are improved over counterparts without joint training across three benchmark ASR tasks.
- Further introduce novel time-synchronous beam search algorithms for joint CTC/RNN-T/attention decoding, which outperforms CTC/attention decoding on average.

10.21437/Interspeech.2023-1303

2. Preliminary

This section describes our Conformer-based encoder, which is commonly used in the four models, CTC, attention, RNN-T, and Mask-CTC. Then, we describe each decoder in detail.

2.1. Universal encoder

Conformer [12, 30] is used as a universal encoder in our study, which consists of two convolutional layers, a linear projection layer, and a positional encoding layer, followed by Conformer blocks. The convolutional layers subsample an audio feature sequence, $\boldsymbol{X}$, into a subsampled feature sequence. Then, the Conformer blocks transform the subsampled feature sequence to a T-length hidden state sequence, $\boldsymbol{H} = [\boldsymbol{h}_1, ..., \boldsymbol{h}_T]$, described as,

$$\boldsymbol{H} = \text{ConEncoder}(\boldsymbol{X}). \tag{1}$$

2.2. CTC

The CTC decoder estimates the output token sequence, $\boldsymbol{y}$, given $\boldsymbol{H}$ generated by the encoder in Eq. (1). CTC introduces the alignment sequence $\boldsymbol{z} = \{\boldsymbol{z}_t \in V \cup \{\phi\}\}$, where t, V, and ϕ denote the time index, the vocabulary, and a blank token, respectively. CTC estimates the alignment posteriors, $P(\boldsymbol{z}|\boldsymbol{X})$. Each alignment sequence, $\boldsymbol{z}$, is deterministically mapped to a corresponding S-length output sequence, $\boldsymbol{y} = [y_1, ..., y_S]$. CTC assumes conditional independence, yielding:

$$P_{\text{ctc}}(\boldsymbol{y} \mid \boldsymbol{X}) = \sum_{\boldsymbol{z} \in \mathcal{Z}(\boldsymbol{y})} P(\boldsymbol{z} \mid \boldsymbol{X}) = \sum_{\boldsymbol{z} \in \mathcal{Z}(\boldsymbol{y})} \left[\prod_{t=1}^{T} P\left(\boldsymbol{z}_t \mid \boldsymbol{h}_t\right) \right]. \tag{2}$$

During training, CTC optimizes model parameters by minimizing the following negative log-likelihood as follows:

$$L_{\text{ctc}} = -\log P_{\text{ctc}}(\boldsymbol{y} \mid \boldsymbol{X}), \tag{3}$$

where $\mathcal{Z}(\boldsymbol{y})$ is a set of all possible alignment sequences of $\boldsymbol{y}$.

2.3. RNN-T

The RNN-T decoder comprises a prediction network and a joint network. The prediction network generates a high-level representation g_s by conditioning on the previous non-blank token sequence $\boldsymbol{g}_{s-1}$, where s denotes a non-blank token index. The joint network is a feed-forward network that combines $\boldsymbol{h}_t$ and g_s. While CTC assumes the conditional independence in Eq. (2), the RNN-T model marginalizes the potential alignments $\boldsymbol{u}$ that output $\boldsymbol{y}$ as follows:

$$P_{\text{rnnt}}(\boldsymbol{y} \mid \boldsymbol{X}) = \sum_{\boldsymbol{z} \in \mathcal{Z}(\boldsymbol{y})} P(\boldsymbol{z}|\boldsymbol{X}) = \sum_{\boldsymbol{z} \in \mathcal{Z}(\boldsymbol{y})} \left[\prod_{i=1}^{T+S} P\left(\boldsymbol{z}_i \mid \boldsymbol{h}_{t_i}, g_{s_i}\right) \right], \tag{4}$$

where i represents a position in $(T+S)$-length alignment path specified by t_i-th decoder state and s_i-th token, respectively. RNN-T optimizes model parameters by minimizing the following negative log-likelihood described as,

$$L_{\text{rnnt}} = -\log P_{\text{rnnt}}(\boldsymbol{y} \mid \boldsymbol{X}). \tag{5}$$

2.4. Attention

Given $\boldsymbol{H}$ generated by the encoder in Eq. (1) and the previously estimated token sequence $\boldsymbol{y}_{s-1}$, the attention decoder recursively estimates the next token y_s. The token history $\boldsymbol{y}_{s-1}$ is converted to token embeddings and fed into decoder layers with hidden states $\boldsymbol{H}$, unlike $\boldsymbol{h}_t$ in CTC/RNN-T. While CTC and RNN-T uses $\boldsymbol{h}_t$ in Eq. (2) and (4), the likelihood of an attention model is described as follows:

$$P_{\text{att}}(\boldsymbol{y} \mid \boldsymbol{X}) = \prod_{s=1}^{S} P\left(y_s \mid \boldsymbol{y_{s-1}}, \boldsymbol{H}\right). \tag{6}$$

Attention optimizes model parameters by minimizing the following negative log-likelihood described as,

$$L_{\text{att}} = -\log P_{\text{att}}(\boldsymbol{y} \mid \boldsymbol{X}). \tag{7}$$

2.5. Mask-CTC

The Mask-CTC model uses a masked language model (MLM) [19] decoder, which estimates the token sequence using the entire sequence given $\boldsymbol{H}$ in Eq. (1), smilar to the attention case. However, unlike attention, randomly sampled tokens y_{mask} are masked with a special token during training. Then, y_{mask} is predicted conditioning on the remaining unmasked tokens y_{obs} as $P_{\text{mlm}}(y_{\text{mask}}|y_{\text{obs}}, \boldsymbol{X})$. Mask-CTC optimizes model parameters by minimizing the following negative log-likelihood:

$$L_{\text{mlm}} = -\log P_{\text{mlm}}(y_{\text{mask}}|y_{\text{obs}}, \boldsymbol{X}). \tag{8}$$

3. Proposed 4D ASR model

This section describes the joint training using the two-stage optimization strategy and the joint decoding used in the proposed 4D model as shown in Figure 1.

3.1. Joint training with a two stage strategy

Multitask learning is performed using the weighted sum of losses shown in Eqs. (3), (5), (7), and (8) described as follows:

$$L = \lambda_{\text{ctc}} L_{\text{ctc}} + \lambda_{\text{rnnt}} L_{\text{rnnt}} + \lambda_{\text{att}} L_{\text{att}} + \lambda_{\text{mlm}} L_{\text{mlm}}, \tag{9}$$

where λ represents training weights. Training weights are usually determined experimentally [23] or based on meta-learning [31]. In this work, with four weights, experimenting with all possible combinations would be overly time-consuming. To address this issue, we used a two-stage optimization strategy to determine the multitask weights $(\lambda_{\text{ctc}}, \lambda_{\text{rnnt}}, \lambda_{\text{att}}, \lambda_{\text{mlm}})$ in Eq. (9). In the first stage, all four training weights were set to be equal, i.e., $(0.25, 0.25, 0.25, 0.25)$. Then, in the second stage, the training weights were determined to be roughly proportional to the number of epochs in the first stage at which each validation loss reached its minimum value. For example, if the validation losses $(L_{\text{ctc}}, L_{\text{rnnt}}, L_{\text{att}}, L_{\text{mlm}})$ takes their minimum values at the 10th, 10th, 10th, and 70th epochs in the first stage, the training weights in the second stage were set to $(0.1, 0.1, 0.1, 0.7)$. This strategy is based on the proposition that losses requiring more epochs to convergence should be given higher weights.

3.2. Time synchronous one-pass joint decoding

Another key contribution of this paper is to propose two one-pass joint CTC/RNN-T/attention decoding algorithms using time-synchrony: Algorithms 1 and 2 present CTC-driven and RNN-T-driven algorithms, respectively. While adding an RNN-T decoder to label-synchronous one-pass beam search proposed in [22] would require computing all RNN-T alignment paths,

Algorithm 1 CTC-driven one-pass joint decoding

```
 1: $hyps$ = {<sos>:1.0}; $ext\_hyps$ = {<sos>:1.0}
 2: for $t = 1$ to $T$ do
 3:     for $\boldsymbol{l} \in hyps$ do
 4:         for $z_t \in$ top-k($P(z_t \mid \boldsymbol{h}_t)$, $k_{\mathrm{pre}}$) do
 5:             $\tilde{\boldsymbol{l}} = \boldsymbol{l} + z_t$ (if $z_t$ is not $\emptyset$ else $\tilde{\boldsymbol{l}} = \boldsymbol{l}$)
 6:             Add $\tilde{\boldsymbol{l}}$ and score to $ext\_hyps$
 7:         end for
 8:     end for
 9:     for $\tilde{\boldsymbol{l}} \in ext\_hyps$ do
10:         $\alpha_{\mathrm{ctc}} = ext\_hyps[\tilde{\boldsymbol{l}}]$
11:         $\alpha_{\mathrm{att}}$ = AttScore($\tilde{\boldsymbol{l}}, \boldsymbol{H}$)
12:         $ext\_hyps[\tilde{\boldsymbol{l}}] = \mu_{\mathrm{ctc}}\,\alpha_{\mathrm{ctc}} + \mu_{\mathrm{att}}\,\alpha_{\mathrm{att}}$
13:     end for
14:     $ext\_hyps$ = top-k($ext\_hyps$, $k_{\mathrm{pre}}$)
15:     for $\tilde{\boldsymbol{l}} \in ext\_hyps$ do
16:     $\alpha_{\mathrm{rnnt}}$ = RNNTScore($\tilde{\boldsymbol{l}}, \boldsymbol{h}_{1:t}$)
17:         $ext\_hyps[\tilde{\boldsymbol{l}}] = \mu_{\mathrm{ctc}}\,\alpha_{\mathrm{ctc}} + \mu_{\mathrm{att}}\,\alpha_{\mathrm{att}} + \mu_{\mathrm{rnnt}}\,\alpha_{\mathrm{rnnt}}$
18:     end for
19:     $hyps$ = top-k($ext\_hyps$, $k_{\mathrm{beam}}$)
20: end for
21: return $hyps$
```

Hypothesis generation by a primary decoder

Joint scoring

Algorithm 2 RNN-T-driven one-pass joint decoding

```
 1: $hyps$ = {<blank>:1.0}
 2: for $t = 1$ to $T$ do
 3:     $A = hyps$; $ext\_hyps$ = {}
 4:     while $ext\_hyps$ contains less than $k_{\mathrm{pre}}$ elements more
        probable than the most probable in $A$ do
 5:         $\tilde{\boldsymbol{l}}$ = most probable in $A$; remove $\tilde{\boldsymbol{l}}$ from $A$
 6:         for $z_t$ in top-k($P(z_t \mid \boldsymbol{h}_t, \boldsymbol{g}_s)$, $k_{\mathrm{pre}}$) do
 7:             Add $\tilde{\boldsymbol{l}}$ and score to $ext\_hyps$ (if $z_t$ is $\emptyset$ else Add $\tilde{\boldsymbol{l}} \oplus$
                $z_t$ to $A$)
 8:         end for
 9:     end while
10:     for $\tilde{\boldsymbol{l}} \in ext\_hyps$ do
11:         $\alpha_{\mathrm{rnnt}} = ext\_hyps[\tilde{\boldsymbol{l}}]$
12:         $\alpha_{\mathrm{ctc}}$ = CTCPrefixScore($\tilde{\boldsymbol{l}}, \boldsymbol{H}$)
13:         $\alpha_{\mathrm{att}}$ = AttScore($\tilde{\boldsymbol{l}}, \boldsymbol{H}$)
14:         $ext\_hyps[\tilde{\boldsymbol{l}}] = \mu_{\mathrm{ctc}}\,\alpha_{\mathrm{ctc}} + \mu_{\mathrm{att}}\,\alpha_{\mathrm{att}} + \mu_{\mathrm{rnnt}}\,\alpha_{\mathrm{rnnt}}$
15:     end for
16:     $hyps$ = top-k($ext\_hyps$, $k_{\mathrm{beam}}$)
17: end for
18: return $hyps$
```

the proposed method avoids this inefficient computation by using time synchrony. Note that we excluded the Mask-CTC from the proposed joint decoding because, unlike the other three decoders, it predicts the entire output sequence in parallel.

The CTC-driven method uses CTC as a primary decoder, which generates initial hypotheses, *ext_hyps* (lines 3-8 in Algorithm 1); this is similar to CTC-driven joint decoding methods with attention [32,33] or a language model [34], but additionally accounts for RNN-T likelihoods. The RNN-T-driven method uses RNN-T as a primary (lines 4-9 in Algorithm 2); this time-synchronous method is performed autoregressively, unlike iterative CTC refinement of RNN-T [21], plus we additionally account for attention likelihoods.

Then, the generated hypotheses, *ext_hyps*, are scored combining CTC, attention, and RNN-T decoders (lines 9-18 in Algorithm 1, lines 10-15 in Algorithm 2). The attention score is calculated using the forward computation as in Eq. (6) for both joint decoding methods. The CTC score is calculated using dynamic programming as in [1] for the CTC-driven method, and CTC prefix scoring proposed in [22] is used for the RNN-T-driven method. As for RNN-T scoring, the probabilities of all possible paths from the previous hypotheses at $t-1$ to the current hypothesis at t are added for the CTC-driven method, whereas the RNN-T score is calculated as in the conventional RNN-T [4]. Each decoder score α is added with the decoder weights (μ_{ctc}, μ_{att}, μ_{rnnt}) (line 17 in Algorithm 1, line 14 in Algorithm 2).

Top k_{beam} hypotheses, *hyps*, are retained for the next time frame based on the obtained joint score (line 19 in Algorithm 1, line 16 in Algorithm 2), where k_{beam} denotes the main beam size.

Note that, RNN-T-driven joint decoding generates only k_{pre} hypotheses (line 4 in Algorithm 2), whereas CTC-driven joint decoding generates $k_{\mathrm{pre}} \times k_{\mathrm{beam}}$ hypotheses (line 3-8 in Algorithm 1). In other words, CTC-driven joint decoding requires roughly k_{beam} times more computation for scoring. Therefore, we pruned the hypotheses in CTC-driven joint decoding to reduce the computational cost. The top k_{pre} hypotheses were chosen specifically based on the CTC and attention scores before RNN-T scoring (line 14 in Algorithm 1).

4. Experiments

4.1. Experimental setup

The input features were 80-dimensional Mel-scale filter-bank features with a window size of 512 samples and a hop length of 160 samples. The sampling frequency was 16 kHz. SpecAugment [35] was then used. The encoder consisted of two convolutional layers and a 512-dimensional linear projection layer followed by 12 Conformer layers with 2048 linear units. The CTC decoder had a 1-layer linear layer. The attention and MLM decoders had six Transformer layers with 2048 linear units each. We set the attention dimension size to 512 with 8-multi-head attention. The RNN-T decoder used a 1-layer long short-term memory (LSTM) with a 512 hidden size and a linear layer of 640 joint sizes for the prediction and joint networks, respectively. The proposed model was trained 150 epochs using the Adam optimizer at a learning rate of 0.0015. The training weights (λ_{ctc}, λ_{rnnt}, λ_{att}, λ_{mlm}) of the second stage were (0.15, 0.10, 0.30, 0.45) based on the two-stage strategy described in Section 3.1. The decoder weights (μ_{ctc}, μ_{rnnt}, μ_{att}) of CTC-driven and RNN-T-driven joint decoding in Algorithms 1 and 2 were (0.2, 0.2, 0.6) and (0.1, 0.4, 0.5), respectively.

The proposed method was tested using the LibriSpeech (960 h, 100 h) [36] and our in-house dataset (855 h). Our in-house dataset[1] consists of 93 hours of Japanese speech data, including meeting and morning assembly scenarios, plus the Corpus of Spontaneous Japanese (581 h) [37] and the 181 hours of Japanese speech database developed by the Advanced Telecommunications Research Institute International (ATR-APP) [38]. The word/character error rate (WER/CER) were calculated for LibriSpeech and our in-house dataset, respectively. We used the ESPnet [39] toolkit.

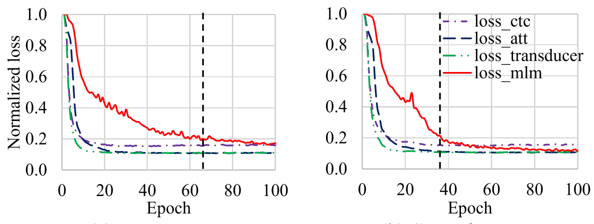
4.2. Effect of the joint training

Table 1 presents two forms of results, the effect of the joint training and decoding. First, we show that the single-decoder branches of our 4D models outperform their counterparts without joint training (`A1-4` vs. `B1-4`) – 4D joint encoders

[1]Our in-house dataset is not released for privacy and confidentiality reasons.

Table 1: *Our 4D models with jointly trained encoders (`B1-8`) compared to respective baselines (`A1-5`). Best WER/CER result in each comparison is* ***bolded*** *and best results overall are further* ***underlined****. The average absolute improvements (Δ) are also shown.*

		JOINT	LIBRISPEECH 960 H		LIBRISPEECH 100 H		IN-HOUSE		
ID	MODEL NAME	ENCODER	test-clean	test-other	test-clean	test-other	assembly	meeting	avg Δ
`A1`	Attention *(baseline)*	-	2.88	**5.62**	8.59	19.43	3.81	5.82	-
`B1`	4D Attention	✓	**2.66**	**5.62**	**7.83**	**17.91**	**3.56**	**5.31**	-0.54
`A2`	CTC *(baseline)*	-	3.10	6.90	8.20	20.66	3.91	6.30	-
`B2`	4D CTC	✓	**2.84**	**6.39**	**7.30**	**18.96**	**3.74**	**5.53**	-0.72
`A3`	Mask-CTC *(baseline)*	-	**2.97**	6.92	8.72	20.78	4.37	6.56	-
`B3`	4D Mask-CTC	✓	3.11	**6.82**	**7.47**	**18.97**	**3.86**	**6.24**	-0.64
`A4`	RNN-T *(baseline)*	-	2.66	5.82	7.25	18.30	4.00	5.91	-
`B4`	4D RNN-T	✓	**2.56**	**5.74**	**7.10**	**17.61**	**3.94**	**5.30**	-0.28
`A5`	CTC/Attention *(baseline)*	-	2.45	**5.18**	7.08	17.80	3.75	5.72	-
`B5`	4D CTC/Attention	✓	**2.42**	5.31	**6.49**	**17.03**	**3.67**	**5.17**	-0.32
`A5`	CTC/Attention *(baseline)*	-	2.45	<u>**5.18**</u>	7.08	17.80	3.75	5.72	-
`B6`	4D RNN-T/Attention (RNN-T-driven)	✓	<u>**2.37**</u>	5.25	**6.35**	**16.47**	3.81	**5.17**	-0.43
`B7`	4D CTC/RNN-T/Attn (CTC-driven)	✓	**2.42**	5.25	<u>**6.32**</u>	**16.48**	**3.69**	<u>**5.12**</u>	-0.45
`B8`	4D CTC/RNN-T/Attn (RNN-T-driven)	✓	**2.38**	5.21	**6.33**	<u>**16.43**</u>	<u>**3.65**</u>	**5.16**	-0.47



(a) *First stage* (b) *Second stage*

Figure 2: *Validation curves of the first vs. second stage.*

Table 2: *Effect of the two stage strategy with Librispeech 100 h.*

Decoder	w/o multitasking	1st stage	2nd stage
Attention	19.43	18.69	**17.91**
CTC	20.66	19.52	**18.96**
Mask-CTC	20.78	20.38	**18.97**
RNN-T	18.30	18.10	**17.61**

improve CTC, attention, RNN-T, and Mask-CTC models by 0.28-0.72 WER/CER overall. This improvement persists for CTC/attention models as well (`A5` vs. `B5`). The effect of the joint decoding (`B6-8`) will be discussed in Section 4.4

4.3. Analysis of the two-stage training strategy

Figure 2 shows the normalized validation losses in the first and second stages. The MLM loss took more epochs to converge than the other three decoders in the first training (Figure 2a), indicating that the trained model did not converge sufficiently with the MLM decoder, or the other decoders were overfitted. The difference in the convergence speed of the four losses, on the other hand, was smaller in the second training, indicating that the four losses converged relatively adequately (Figure 2b). Table 2 shows the performance of each decoder on the test-other set of Librispeech 100 h without and with joint training in the first/second stage. Even the first stage outperformed the model without multitask learning, the performance of all four decoders improved in the second stage. Using the proposed two-stage approach, the four weights were efficiently determined with only two experimental trials.

4.4. Effect of the joint decoding

Table 1 also presents the results of joint decoding (`A5` and `B6-8`). We show that 4D models also offer RNN-T/attention and CTC/RNN-T/attention decoding, which out-

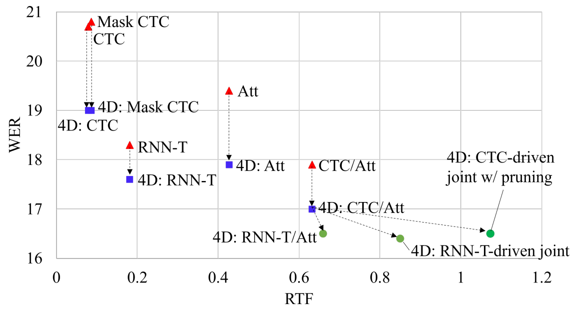


Figure 3: *Relationship between RTFs and WERs. The red, blue, and green dots denote the baselines, 4D model without joint decoding, and 4D model with joint decoding, respectively.*

perform CTC/attention on average (`A5` vs. `B6-8`) – RNN-T-driven CTC/RNN-T/attention decoding is 0.47 WER/CER better than CTC/attention overall. Note that these improvements are more pronounced on Librispeech 100 h, suggesting that the 4D method offers a regularization effect which is important for smaller amounts of training data.

Figure 3 shows the relationship between real-time factor (RTF) using a GPU (NVIDIA RTX3090) and WER on the Librispeech 100 h test-other set. The red dots denote the baselines, the blue dots denote 4D joint training but *without* joint decoding, and the green dots denote 4D joint training *with* joint decoding. Comparing the red and blue dots, the proposed 4D model reduced WER for all decoders without increasing RTF as long as a single decoder is used. The proposed two joint decoding methods had larger RTFs than the other decoders due to increased complexity, but the WERs were the smallest. CTC-driven joint decoding had a larger RTF than RNN-T joint decoding, even with the pruning as described in Section 3.2.

5. Conclusion

This paper proposed a 4D joint model of CTC, attention, RNN-T, and Mask-CTC by sharing an encoder trained in a multi-task fashion. We demonstrated that jointly trained 4D models with the proposed two-stage training strategy improved the performance of each individual decoder. Furthermore, the proposed joint CTC/RNN-T/attention decoding improved the performance and outperformed the previously proposed CTC/attention decoding.

6. References

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

[2] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. ICML*, 2014, pp. 1764–1772.

[3] S. Kriman, S. Beliaev, B. Ginsburg, J. Huang, O. Kuchaiev, V. Lavrukhin, R. Leary, J. Li, and Y. Zhang, "Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions," in *Proc. ICASSP*, 2020, pp. 6124–6128.

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

[5] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, and S. Kumar, "Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss," in *Proc. ICASSP*, 2020, pp. 7829–7833.

[6] W. Han, Z. Zhang, Y. Zhang, J. Yu, C.-C. Chiu, J. Qin, A. Gulati, R. Pang, and Y. Wu, "Contextnet: Improving convolutional neural networks for automatic speech recognition with global context," in *Proc. Interspeech*, 2020, pp. 3610–3614.

[7] K. Rao, H. Sak, and R. Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with RNN-transducer," in *Proc. ASRU*, 2017, pp. 193–199.

[8] J. Lee and S. Watanabe, "Intermediate loss regularization for ctc-based speech recognition," in *Proc. ICASSP*, 2021, pp. 6224–6228.

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

[10] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Proc. NeurIPS*, vol. 28, 2015, pp. 577–585.

[11] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang *et al.*, "A comparative study on transformer vs rnn in speech applications," in *Proc. ASRU*, 2019, pp. 449–456.

[12] P. Guo, F. Boyer, X. Chang, T. Hayashi, Y. Higuchi, H. Inaguma, N. Kamo, C. Li, D. Garcia-Romero, J. Shi *et al.*, "Recent developments on espnet toolkit boosted by conformer," in *Proc. ICASSP*, 2021, pp. 5874–5878.

[13] Y. Higuchi, S. Watanabe, N. Chen, T. Ogawa, and T. Kobayashi, "Mask ctc: Non-autoregressive end-to-end asr with ctc and mask predict," in *Proc. Interspeech*, 2020, pp. 3655–3659.

[14] N. Chen, S. Watanabe, J. Villalba, P. Żelasko, and N. Dehak, "Non-autoregressive transformer for speech recognition," *IEEE Signal Process. Lett.*, vol. 28, pp. 121–125, 2020.

[15] X. Song, Z. Wu, Y. Huang, C. Weng, D. Su, and H. Meng, "Non-autoregressive transformer asr with ctc-enhanced decoder input," in *Proc. ICASSP*, 2021, pp. 5894–5898.

[16] L. Kürzinger, D. Winkelbauer, L. Li, T. Watzel, and G. Rigoll, "Ctc-segmentation of large corpora for german end-to-end speech recognition," in *Speech and Computer*, A. Karpov and R. Potapova, Eds. Cham: Springer International Publishing, 2020, pp. 267–278.

[17] M. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. EMNLP*, 2015, pp. 1412–1421.

[18] M. Sperber, G. Neubig, J. Niehues, and A. Waibel, "Attention-passing models for robust and data-efficient end-to-end speech translation," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 313–325, 2019.

[19] M. Ghazvininejad, O. Levy, Y. Liu, and L. Zettlemoyer, "Mask-predict: Parallel decoding of conditional masked language models," in *Proc. EMNLP-IJCNLP*, 2019.

[20] H. Futami, H. Inaguma, S. Ueno, M. Mimura, S. Sakai, and T. Kawahara, "Non-autoregressive error correction for ctc-based asr with phone-conditioned masked lm," in *Proc. Interspeech*, 2022, pp. 3889–3893.

[21] W. Wang, K. Hu, and T. N. Sainath, "Deliberation of streaming rnn-transducer by non-autoregressive decoding," in *Proc. ICASSP*, 2022, pp. 7452–7456.

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

[23] S. Kim, T. Hori, and S. Watanabe, "Joint ctc-attention based end-to-end speech recognition using multi-task learning," in *Proc. ICASSP*, 2017, pp. 4835–4839.

[24] S. Ueno, H. Inaguma, M. Mimura, and T. Kawahara, "Acoustic-to-word attention-based model complemented with character-level ctc-based model," in *Proc. ICASSP*, 2018, pp. 5804–5808.

[25] T. Hori, S. Watanabe, and J. R. Hershey, "Joint ctc/attention decoding for end-to-end speech recognition," in *Proc. ACL*, 2017, pp. 518–529.

[26] T. N. Sainath, R. Pang, D. Rybach, Y. He, R. Prabhavalkar, W. Li, M. Visontai, Q. Liang, T. Strohman, Y. Wu *et al.*, "Two-pass end-to-end speech recognition," in *Proc. Interspeech*, 2019, pp. 2713–2777.

[27] K. Hu, R. Pang, T. N. Sainath, and T. Strohman, "Transformer based deliberation for two-pass speech recognition," in *Proc. SLT*, 2021, pp. 68–74.

[28] Z. Tian, J. Yi, J. Tao, S. Zhang, and Z. Wen, "Hybrid autoregressive and non-autoregressive transformer models for speech recognition," *IEEE Signal Process. Lett.*, vol. 29, pp. 762–766, 2022.

[29] Z. Yao, D. Wu, X. Wang, B. Zhang, F. Yu, C. Yang, Z. Peng, X. Chen, L. Xie, and X. Lei, "Wenet: Production oriented streaming and non-streaming end-to-end speech recognition toolkit," in *Proc. Interspeech*, 2021, pp. 4045–4058.

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

[31] X. Lin, H. S. Baweja, G. A. Kantor, and D. Held, "Adaptive auxiliary task weighting for reinforcement learning," in *Proc. NeurIPS*, 2019.

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

[33] B. Yan, S. Dalmia, Y. Higuchi, G. Neubig, F. Metze, A. W. Black, and S. Watanabe, "Ctc alignments improve autoregressive translation," *arXiv preprint arXiv:2210.05200*, 2022.

[34] A. Y. Hannun, A. L. Maas, D. Jurafsky, and A. Y. Ng, "First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns," *arXiv preprint arXiv:1408.2873*, 2014.

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

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

[37] K. Maekawa, "Corpus of spontaneous Japanese: Its design and evaluation," in *ISCA & IEEE Workshop on SSPR*, 2003.

[38] A. Kurematsu, K. Takeda, Y. Sagisaka, S. Katagiri, H. Kuwabara, and K. Shikano, "Atr japanese speech database as a tool of speech recognition and synthesis," *Speech Communication*, vol. 9, no. 4, pp. 357–363, 1990.

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