BERT Meets CTC: New Formulation of End-to-End Speech Recognition with Pre-trained Masked Language Model

Yosuke Higuchi[1,2], **Brian Yan**[1], **Siddhant Arora**[1], **Tetsuji Ogawa**[2], **Tetsunori Kobayashi**[2], **Shinji Watanabe**[1]
[1]Carnegie Mellon Univeristy, [2]Waseda University
higuchi@pcl.cs.waseda.ac.jp

Abstract

This paper presents BERT-CTC, a novel formulation of end-to-end speech recognition that adapts BERT for connectionist temporal classification (CTC). Our formulation relaxes the conditional independence assumptions used in conventional CTC and incorporates linguistic knowledge through the explicit output dependency obtained by BERT contextual embedding. BERT-CTC attends to the full contexts of the input and hypothesized output sequences via the self-attention mechanism. This mechanism encourages a model to learn inner/inter-dependencies between the audio and token representations while maintaining CTC's training efficiency. During inference, BERT-CTC combines a mask-predict algorithm with CTC decoding, which iteratively refines an output sequence. The experimental results reveal that BERT-CTC improves over conventional approaches across variations in speaking styles and languages. Finally, we show that the semantic representations in BERT-CTC are beneficial towards downstream spoken language understanding tasks.

1 Introduction

The field of natural language processing (NLP) has witnessed remarkable improvements in performance thanks to the advances in deep learning-based techniques (Collobert et al., 2011; Bahdanau et al., 2015; Sutskever et al., 2014; Vaswani et al., 2017; Young et al., 2018). Much of the recent progress in NLP lies in large-scale language models (LMs) (Devlin et al., 2019; Brown et al., 2020), which are pre-trained on a vast amount of text data to learn versatile linguistic knowledge (Tenney et al., 2019). Such pre-trained models have been shown to improve diverse NLP tasks, alleviating the heavy requirement of supervised training data. Inspired by the great success in NLP, pre-trained LMs have been actively adopted for speech processing tasks, including automatic speech recognition (ASR) (Shin et al., 2019; Huang et al., 2021), spoken language understanding (SLU) (Chuang et al., 2020; Chung et al., 2021), and text-to-speech synthesis (Hayashi et al., 2019; Kenter et al., 2020).

This paper focuses on leveraging pre-trained LMs for end-to-end ASR (E2E-ASR), which aims to model direct speech-to-text conversion (Graves and Jaitly, 2014; Chorowski et al., 2015; Chan et al., 2016). One of the challenges in E2E-ASR is a huge discrepancy between input and output sequences; the input is a continuous acoustic signal with fine-grained patterns, while the output is discrete linguistic symbols (e.g., words) with long-range dependencies. Such an input-output gap makes it difficult for an E2E-ASR model to extract semantic/morphosyntax information from speech, which is essential for generating proper text. We believe this limitation can be mitigated by taking advantage of the rich linguistic representations obtained from pre-trained LMs.

Several attempts have been made to use pre-trained LMs indirectly for improving E2E-ASR, such as N-best hypothesis rescoring (Shin et al., 2019; Salazar et al., 2020; Chiu and Chen, 2021; Futami et al., 2021; Udagawa et al., 2022) and knowledge distillation (Futami et al., 2020; Bai et al., 2021; Kubo et al., 2022). Others have investigated directly unifying an E2E-ASR model with a pre-trained LM, where the LM is fine-tuned to optimize ASR in an end-to-end trainable framework (Huang et al., 2021; Yi et al., 2021; Zheng et al., 2021; Deng et al., 2021; Yu et al., 2022).

We explore a novel direction for adopting a pre-trained masked language model (MLM) for E2E-ASR, based on connectionist temporal classification (CTC) (Graves et al., 2006). Compared to other autoregressive approaches, such as RNN-Transducer (RNN-T) (Graves, 2012) and attention-based sequence-to-sequence (Chorowski et al., 2015), CTC's non-autoregressive formulation allows simple training and inference processes

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 5486–5503
December 7-11, 2022 ©2022 Association for Computational Linguistics

for realizing E2E-ASR. However, the performance of CTC is often limited due to a *conditional independence assumption* between output tokens (Chiu et al., 2018). In this work, we propose **BERT-CTC** that adapts BERT (Devlin et al., 2019) for CTC to mitigate the conditional independence assumption. BERT-CTC conditions CTC outputs on context-aware BERT embeddings, thereby incorporating explicit linguistic information into training/inference. The BERT-conditional formulation enables a model to attend to the full contexts of the input and hypothesized output sequences via the self-attention mechanism, while maintaining the benefits of a simple training algorithm in CTC. During inference, BERT-CTC combines a mask-predict algorithm with CTC decoding, which iteratively refines outputs with flexible length adjustment.

The key contributions of this work are summarized as follows:

- We propose BERT-CTC, which efficiently adapts a pre-trained MLM for CTC-based E2E-ASR without fine-tuning. We provide a probabilistic formulation of our BERT-CTC and its close relation to conventional approaches, i.e., CTC and RNN-T.
- We evaluate BERT-CTC in various ASR tasks, which demonstrates its effectiveness regardless of variations in speaking styles and languages. We also show its potential application to end-to-end SLU.
- The codes and recipes are open-sourced on ESPnet (Watanabe et al., 2018), the widely used toolkit for end-to-end speech processing.[1] We hope our work encourages further research on combining ASR with pre-trained LMs, helping to bridge ASR and NLP fields.

2 Background

To understand how BERT-CTC exploits BERT for relaxing the conditional independence assumption in CTC, we start with a brief review of probabilistic formulations of conventional E2E-ASR approaches, including CTC (Graves et al., 2006; Graves and Jaitly, 2014) and RNN-T (Graves, 2012; Graves et al., 2013).

Definition of End-to-End ASR Let $O = (\mathbf{o}_t \in \mathbb{R}^D | t = 1, \cdots, T)$ be an input sequence of length T, and $W = (w_n \in \mathcal{V} | n = 1, \cdots, N)$ be the corresponding output sequence of length N. Here, $\mathbf{o}_t$ is a D-dimensional acoustic feature at frame t, w_n is an output token at position n, and $\mathcal{V}$ is a vocabulary.[2] In general, the output length is much shorter than the input length (i.e., $N \ll T$). The objective of ASR is to find the most probable output sequence $\hat{W}$ that corresponds to a given input sequence O:

$$\hat{W} = \operatorname*{argmax}_{W \in \mathcal{V}^*} p(W|O), \tag{1}$$

where $\mathcal{V}^*$ denotes all possible token sequences. E2E-ASR aims to realize the direct mapping from O to W by modeling the posterior distribution $p(W|O)$ with a single deep neural network.

2.1 Connectionist Temporal Classification

CTC formulates E2E-ASR by considering all possible alignments between an input sequence O and output sequence W. To align the sequences at the frame level, CTC augments an output sequence by allowing repetitions of the same token and inserting a blank symbol ϵ for representing "no output token" (e.g., silence). Let A denote an augmented output sequence defined as $A = (a_t \in \mathcal{V} \cup \{\epsilon\} | t = 1, \cdots, T)$, which we refer to as an *alignment* between O and W.

With the introduction of the frame-level alignment, CTC factorizes $p(W|O)$ as follows:

$$p_{\text{ctc}}(W|O) = \sum_{A \in \mathcal{B}_{\text{ctc}}^{-1}(W)} p(W|A, \cancel{O}) p(A|O) \tag{2}$$

$$\approx \sum_{A \in \mathcal{B}_{\text{ctc}}^{-1}(W)} p(A|O), \tag{3}$$

where $\mathcal{B}_{\text{ctc}}$ is the collapsing function (Graves et al., 2006) that maps A to W by suppressing repeated tokens and removing blank symbols, and $\mathcal{B}_{\text{ctc}}^{-1}(W)$ is a set of all possible CTC alignments that are compatible with W. To obtain Eq. (3), CTC makes a conditional independence assumption of O in Eq. (2), and we assume $p(W|A) = 1$, as W can be determined uniquely by the collapsing function.

The joint probability $p(A|O)$ is further factorized using the probabilistic chain rule as

$$p(A|O) \approx \prod_{t=1}^{T} p(a_t | \cancel{a_1, \cdots, a_{t-1}}, O). \tag{4}$$

[1] https://github.com/YosukeHiguchi/espnet/tree/bert-ctc

[2] We consider $\mathcal{V}$ as a vocabulary constructed for pre-training a large-scale MLM, i.e., BERT.

In Eq. (4), CTC makes a conditional independence assumption between output tokens, where $p(A|O)$ is approximated as the product of token emission probabilities at each time frame. The conditional probability $p(a_t|O)$ in Eq. (4) is computed as

$$p(a_t|O) = \text{Softmax}(\text{Linear}(\mathbf{h}_t^{\text{ae}})), \tag{5}$$
$$\mathbf{h}_t^{\text{ae}} \sim \text{AudioEnc}(O). \tag{6}$$

In Eq. (5), $\text{Softmax}(\cdot)$ is a softmax function, and $\text{Linear}(\cdot)$ is a linear projection layer. $\text{AudioEnc}(\cdot)$ in Eq. (6) is an audio encoder network that embeds speech input into a sequence of d^{ae}-dimensional hidden vectors $H^{\text{ae}} = (\mathbf{h}_t^{\text{ae}} \in \mathbb{R}^{d^{\text{ae}}} | t = 1, \cdots, T)$.

Training The objective function of CTC is defined by the negative log-likelihood of Eq. (4) over all possible alignments:

$$\mathcal{L}_{\text{ctc}}(O, W) = -\log \sum_{A \in \mathcal{B}_{\text{ctc}}^{-1}(W)} \prod_{t=1}^{T} p(a_t|O). \tag{7}$$

The summation in Eq. (7) is efficiently computed via dynamic programming (Graves et al., 2006).

Inference Eq. (1) is solved using the best path decoding algorithm (Graves et al., 2006). The algorithm first obtains the most probable alignment $\hat{A}$ in a greedy manner, concatenating the most active tokens at each frame: $\hat{a}_t = \text{argmax}_{a_t} p(a_t|O)$. The most probable token sequence $\hat{W}$ is then obtained by applying the collapsing function to $\hat{A}$ as $\hat{W} = \mathcal{B}_{\text{ctc}}(\hat{A})$.

2.2 RNN-Transducer

CTC estimates the distribution over alignments only depending on speech input (Eq. (4)). Thus, by definition, CTC cannot consider output dependencies, preventing a model from properly capturing the multimodal distribution of target token sequences (Gu et al., 2018).

RNN-T overcomes this problem by making each token prediction explicitly conditioned on the previous non-blank output tokens $(w_1, \cdots, w_{n-1})$. Let $Z = (z_u \in \mathcal{V} \cup \{\epsilon\} | u = 1, \cdots, T + N)$ be an alignment used in RNN-T, and RNN-T factorizes $p(W|O)$ similarly to Eq. (3) as

$$p_{\text{rnnt}}(W|O) \approx \sum_{Z \in \mathcal{B}_{\text{rnnt}}^{-1}(W)} p(Z|O), \tag{8}$$

where $\mathcal{B}_{\text{rnnt}}$ is the collapsing function of RNN-T (Graves, 2012) that map Z to W. The joint probability $p(Z|O)$ is factorized using the probabilistic chain rule *without* the conditional independence assumption (cf. Eq. (4)) as

$$p(Z|O) = \prod_{u=1}^{T+N} p(z_u | z_1, \cdots, z_{u-1}, O) \tag{9}$$
$$\approx \prod_{u=1}^{T+N} p(z_u | \underbrace{w_1, \cdots, w_{n_u - 1}}_{= \mathcal{B}_{\text{rnnt}}(z_1, \cdots, z_{u-1})}, O), \tag{10}$$

where n_u is the number of tokens predicted up to an index of u. From Eq (9) to Eq. (10), RNN-T assumes $(z_1, \cdots, z_{u-1}) \approx (w_1, \cdots, w_{n_u - 1})$, which is reasonable since W can be determined uniquely by the collapsing function. The conditional probability $p(z_u | w_1, \cdots, w_{n_u - 1}, O)$ is computed as

$$p(z_u | w_1, \cdots, w_{n_u-1}, O) = \text{Softmax}(\text{JointNet}(\mathbf{h}_t^{\text{ae}}, \mathbf{h}_{n_u}^{\text{pn}})), \tag{11}$$
$$\mathbf{h}_{n_u}^{\text{pn}} = \text{PredictionNet}(w_1, \cdots, w_{n_u-1}). \tag{12}$$

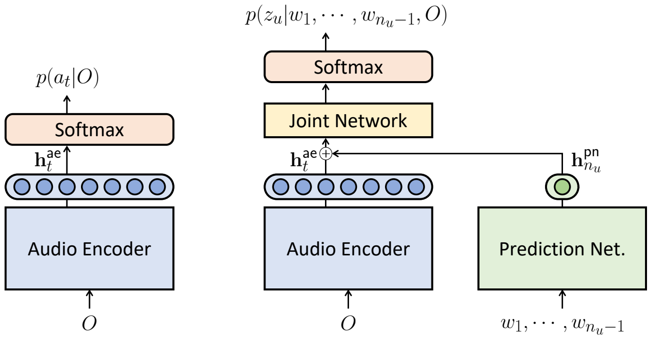
In Eq. (11), $\mathbf{h}_t^{\text{ae}}$ is obtained from the audio encoder (Eq. (6)), and $\text{JointNet}(\cdot)$ is a joint network that combines the audio and token representations, $\mathbf{h}_t^{\text{ae}}$ and $\mathbf{h}_{n_u}^{\text{pn}}$, using a linear projection layer. In Eq. (12), $\text{PredictionNet}(\cdot)$ is a prediction network that encodes the previous non-blank output tokens to a hidden vector $\mathbf{h}_{n_u}^{\text{pn}}$. The adoption of the prediction network is the main difference from CTC, which explicitly captures causal dependency in outputs.

Training The RNN-T loss $\mathcal{L}_{\text{rnnt}}(O, W)$ is defined by the negative log-likelihood of Eq. (10). Similar to the CTC objective in Eq. (7), the summation over alignments is efficiently computed using dynamic programming (Graves, 2012).

Inference RNN-T estimates the most probable token sequence $\hat{W}$ using the beam search algorithm proposed in (Graves, 2012).

3 BERT-CTC

Overview In Fig. 1, we compare our proposed E2E-ASR model, **BERT-CTC**, to CTC and RNN-T. BERT-CTC leverages powerful representations from BERT (Devlin et al., 2019) to make CTC training/inference explicitly conditioned on linguistic information (Fig. 1(a) vs. Fig. 1(c)). We use BERT as a feature extractor for a (masked) token sequence, whose parameters are frozen during training. BERT-CTC can be similar to RNN-T in that audio and token representations are fused to estimate the distribution over alignments (Fig. 1(b)



(a) CTC (b) RNN-T (c) BERT-CTC (ours)

Figure 1: Comparisons between different model architectures for end-to-end ASR.

vs. Fig. 1(c)). However, BERT-CTC attends to the full contexts of the input and output sequences via the self-attention mechanism (Vaswani et al., 2017), which facilitates a model to learn inner/inter-dependencies within/between the sequences.

BERT-CTC is formulated by introducing a partially masked (or partially observed) sequence $\tilde{W} = (\tilde{w}_n \in \mathcal{V} \cup \{\texttt{[MASK]}\} | n = 1, \cdots, N)$, which is obtained by replacing some tokens in an output sequence W with a special mask token [MASK]. Note that during inference, we apply masks to a hypothesized sequence $\hat{W}$ to obtain a masked sequence. Considering all possible $\tilde{W}$, the conditional probability $p(W|O)$ is factorized as follows:

$$p_{\text{bc}}(W|O) = \sum_{\tilde{W} \in \mathcal{A}(W)} p(W|\tilde{W}, O) p(\tilde{W}|O), \quad (13)$$

where $\mathcal{A}(W)$ covers W with all possible masking patterns. Here, we interpret $p(\tilde{W}|O)$ as a prior distribution of sequences consisting of observed tokens that are easily recognized only from speech input; the other masked tokens are difficult and require contextual information to be determined (e.g., homophones), which is modeled by $p(W|\tilde{W}, O)$. We further describe the above interpretation in the training (§3.1) and inference (§3.2) sections.

The conditional probability $p(W|\tilde{W}, O)$ is further factorized by using the CTC alignment as

$$p(W|\tilde{W}, O) = \sum_{A \in \mathcal{B}_{\text{ctc}}^{-1}(W)} p(W, A|\tilde{W}, O) \quad (14)$$

$$\approx \sum_{A \in \mathcal{B}_{\text{ctc}}^{-1}(W)} p(A|W, \cancel{\tilde{W}}, O) p(W|\tilde{W}, \cancel{O}). \quad (15)$$

In Eq. (15), we make two conditional independence assumptions. The first is that given W and O, $\tilde{W}$ is not required to determine A. This is reasonable because W already contains observed tokens in $\tilde{W}$ and is helpful in avoiding the combination of all possible masked sequences and alignments (i.e., $\mathcal{A} \times \mathcal{B}_{\text{ctc}}^{-1}$). The second is that given $\tilde{W}$, O is not required to determine W. We consider $p(W|\tilde{W})$ as a strong prior modeled by a pre-trained MLM (i.e., BERT), which can be achieved without the observation from O. We empirically show that this assumption holds in §7.3.

Similar to CTC, the joint probability $p(A|W, O)$ is factorized using the probabilistic chain rule as

$$p(A|W, O) \approx \prod_{t=1}^{T} p(a_t|\cancel{a_1, \cdots, a_{t-1}}, W, O). \quad (16)$$

To obtain Eq. (16), we make the same conditional independence assumption as in CTC. However, compared to Eq. (4), Eq. (16) is conditioned on an output sequence W, enabling a model to explicitly use linguistic information to estimate the distribution over alignments. This is somewhat similar to RNN-T (Eq. (10)), but is different in that BERT-CTC attends to the whole context $(w_1, \cdots, w_N)$. We discuss this advantage in §7.1.

Substituting Eq. (16) into Eq. (15), we model the product of $p(a_t|W, O)$ and $p(W|\tilde{W})$ as

$$\text{Eq. (15)} \triangleq \sum_{A \in \mathcal{B}_{\text{ctc}}^{-1}(W)} \prod_{t=1}^{T} p(a_t|\text{BERT}(\tilde{W}), O), \quad (17)$$

where $\text{BERT}(\cdot)$ is the output of BERT representing the distribution of target sequences.[3] This enables Eq. (17) to be realized with a single differentiable model, enabling the whole network to be

[3]Note that $\text{BERT}(\cdot)$ can be any pre-trained MLM.

trained end-to-end. The conditional probability $p(a_t|\text{BERT}(\tilde{W}), O)$ is computed as

$$p(a_t|\text{BERT}(\tilde{W}), O) = \text{Softmax}(\text{SelfAttn}_t(H^{\text{ae}}, H^{\text{bert}})), \quad (18)$$

$$H^{\text{bert}} = \text{BERT}(\tilde{W}). \quad (19)$$

In Eq. (18), $\text{SelfAttn}_t(\cdot)$ indicates the t-th output of stacked Transformer self-attention layers (Vaswani et al., 2017), which consume the concatenated H^{ae} (from Eq. (6)) and H^{bert}.[4] In Eq. (19), $\text{BERT}(\cdot)$ embeds a masked sequence $\tilde{W}$ into a sequence of d^{bert}-dimensional hidden vectors $H^{\text{bert}} = (\mathbf{h}_n^{\text{bert}} \in \mathbb{R}^{d^{\text{bert}}}|n = 1, \cdots, N)$.

3.1 Training

The BERT-CTC objective is defined by the negative log-likelihood of Eq. (13) expanded with Eq. (15):

$$-\log \sum_{\tilde{W}} \sum_{A} p(A|W, O)p(W|\tilde{W})p(\tilde{W}|O). \quad (20)$$

To deal with the intractable marginalization over $\tilde{W}$ in Eq. (20), we rewrite it under expectation with respect to the sampling distribution $\mathcal{A}(W)$:

$$\approx -\log \mathbb{E}_{\tilde{W}\sim\mathcal{A}(W)}\left[\sum_{A} p(A|W, O)p(W|\tilde{W})p(\tilde{W}|O)\right],$$

whose upper bound can be derived by using the Jensen's inequality as

$$\leq -\mathbb{E}_{\tilde{W}\sim\mathcal{A}(W)}\left[\log \sum_{A} p(A|W, O)p(W|\tilde{W})p(\tilde{W}|O)\right]$$

$$\leq \underbrace{-\mathbb{E}_{\tilde{W}\sim\mathcal{A}(W)}\left[\log \sum_{A} \prod_{t} p(a_t|\text{BERT}(\tilde{W}), O)\right]}_{\triangleq \mathcal{L}_{\text{bc}}(O,W)}, \quad (21)$$

where $\mathcal{L}_{\text{bc}}$ is the loss for BERT-CTC training. Compared with the CTC objective (Eq. (7)), each token prediction in Eq. (21) is explicitly conditioned on contextual embedding from BERT. This relaxes the conditional independence assumption between outputs while retaining the same optimization strategy as in CTC. For sampling $\tilde{W}$ from $\mathcal{A}(W)$ in Eq. (21), we first obtain the random number of tokens from a uniform distribution as $M \sim \text{Uniform}(1, N)$. Then, M tokens in a ground-truth sequence W are randomly selected to be replaced with [MASK], similar to (Ghazvininejad et al., 2019).

[4]We apply simple embedding layers to H^{ae} and H^{bert} so that the dimensions of hidden vectors match, but we omit it for simplicity. See Appendix D.2 for detailed implementation.

Hierarchical Loss We apply an auxiliary CTC loss to the audio encoder output in a hierarchical multi-tasking manner (Fernández et al., 2007; Sanabria and Metze, 2018). As the vocabulary size of BERT is often too large for ASR training, we train the audio encoder to predict a sequence $W' = (w'_l \in \mathcal{V}'|l = 1, \cdots, L)$ tokenized with a smaller vocabulary $\mathcal{V}'$ (i.e., $|\mathcal{V}'| \ll |\mathcal{V}|$). This has been shown effective for training sparse word-level ASR (Higuchi et al., 2022). The BERT-CTC loss is combined with the hierarchical CTC loss as

$$(1 - \lambda_{\text{ctc}})\mathcal{L}_{\text{bc}}(O, W) + \lambda_{\text{ctc}}\mathcal{L}_{\text{ctc}}(O, W'), \quad (22)$$

where λ_{ctc} is a tunable parameter. We investigate the importance of the hierarchical loss in §7.1.

3.2 Inference

The most probable token sequence $\hat{W}$ is estimated by solving Eq. (1) for Eq. (13) as

$$\hat{W} = \underset{W}{\text{argmax}} \sum_{\tilde{W}} p(W|\tilde{W}, O)p(\tilde{W}|O) \quad (23)$$

$$\approx \underset{W}{\text{argmax}}\, p(W|\bar{W}, O), \quad (24)$$

$$\text{where } \bar{W} = \underset{\tilde{W}}{\text{argmax}}\, p(\tilde{W}|O). \quad (25)$$

From Eq. (23) to Eq. (24), we make the Viterbi approximation to deal with the intractable summation over all possible masked sequences.

To solve Eq. (24), we design a mask-predict algorithm (Ghazvininejad et al., 2019) assisted by CTC inference, inspired by (Chan et al., 2020; Higuchi et al., 2020). See Table 4 for an example decoding and Appendix A for pseudocode. The algorithm first initializes a target sequence with an estimated length, which is then followed by $k = \{1, \cdots, K\}$ iterations of token masking and prediction steps.

Initialization ($k = 1$) BERT-CTC is non-autoregressive, and the length of a target sequence $\hat{N}$ needs to be given in advance to start decoding (Gu et al., 2018). We determine the target length based on the auxiliary sequence $\hat{W}'$ predicted from the audio encoder output H^{ae} as $\hat{N} \sim |\hat{W}'|$. Given the estimated length, we initialize an initial masked sequence $\bar{W}^{(k=1)}$ by filling all $\hat{N}$ positions with the mask token [MASK]. By feeding H^{ae} and H^{bert} $(= \text{BERT}(\bar{W}^{(k=1)}))$ to the self-attention module, a hypothesized sequence $\hat{W}^{(k=1)}$ is obtained via CTC inference. Here, $\hat{W}^{(k=1)}$ is predicted only from speech without any observations from output tokens, as they are all masked.

Token Masking Step (Eq. (25)) Given a current prediction $\hat{W}^{(k)}$, we replace $m(k)$ tokens having the lowest probability scores with [MASK], which results in the next masked sequence $\bar{W}^{(k+1)}$. Here, $m(k)$ is a linear decay function $m(k) = \lfloor |\hat{W}^{(k)}| \cdot \frac{K-k}{K} \rfloor$, similar to (Ghazvininejad et al., 2019).

Token Prediction Step (Eq. (24)) H^{ae} and H^{bert} ($= \text{BERT}(\bar{W}^{(k+1)})$) are fed to the self-attention module to generate the next hypothesis $\hat{W}^{(k+1)}$. Here, the prediction of $\hat{W}^{(k+1)}$ is conditioned on the contextual embedding obtained from BERT.

Similar to (Chan et al., 2020; Chi et al., 2021), BERT-CTC inference repeatedly predicts a target sequence at the alignment level, which does not require an additional mechanism (Gu et al., 2019; Higuchi et al., 2021b) for adjusting the target length over iterations. Moreover, BERT-CTC considers the output dependencies at the token level, making it more suitable for a model to capture linguistic information.

3.3 BERT-CTC for End-to-End SLU

In addition to E2E-ASR, BERT-CTC can model end-to-end SLU jointly by extending Eq. (18) as

$$\begin{aligned} &p(y|\text{BERT}(\tilde{W}), O) \\ &\quad = \text{Softmax}(\text{SelfAttn}_{T+1}(H^{\mathsf{ae}}, H^{\mathsf{bert}})), \end{aligned} \tag{26}$$

where we assume $y \in \mathcal{Y}$ as an intent label in a set of intents $\mathcal{Y}$. Note that $\text{SelfAttn}_{T+1}(\cdot)$ indicates the $T+1$-th output of the self-attention module, which corresponds to the [CLS] token of BERT.

Training The loss is defined by adding Eq. (22) and the negative log-likelihood of Eq. (26) as

$$\text{Eq. (22)} - \lambda_{\mathsf{slu}} \log p(y|\text{BERT}(\tilde{W}), O), \tag{27}$$

where λ_{slu} is a tunable parameter.

Inference The most probable label $\hat{y}$ can be estimated at any timing of BERT-CTC inference by $\hat{y} = \text{argmax}_y\, p(y|\bar{W}, O)$. When $k = 1$, the label is predicted only from audio information, and when $k = K$, the label is predicted with full access to audio and linguistic information.

4 Additional Related Work

End-to-End ASR with MLM Inspired by the great success in non-autoregressive neural machine translation, conditional masked language model (CMLM) (Ghazvininejad et al., 2019) has been adopted for E2E-ASR. Audio-CMLM (A-CMLM) (Chen et al., 2020) has trained an E2E-ASR model with an MLM objective (Devlin et al., 2019), making token predictions conditioned on both the speech input and a partially masked target sequence. Imputer (Chan et al., 2020) and Mask-CTC (Higuchi et al., 2020, 2021b) have introduced CTC to the CMLM-based modeling, where the mask-predict algorithm is used to refine a frame-level or token-level sequence predicted by CTC.

Our method of combining CTC and MLM is related to the above studies, but conceptually different in that BERT-CTC aims to relax the conditional independence assumption used in CTC by leveraging an external pre-trained MLM (i.e., BERT) as contextual embedding.

LM Integration for End-to-End ASR. There is a line of prior studies seeking to integrate an external LM into E2E-ASR. Shallow fusion has been the most widely used approach (Hannun et al., 2014; Gulcehre et al., 2015; Chorowski and Jaitly, 2017; Kannan et al., 2018), which linearly interpolates the output probabilities from an E2E-ASR model and external LM. Deep fusion (Gulcehre et al., 2015) is a more structured approach, where an E2E-ASR model is jointly trained with an external LM to learn the optimal combination of the audio and linguistic information in a latent space. Cold fusion (Sriram et al., 2018) and component fusion (Shan et al., 2019) have further improved deep fusion by a gating mechanism that learns a more sophisticated combination of the two models.

Our approach can be seen as a variant of cold fusion in that an external pre-trained MLM is fused to a CTC-based E2E-ASR model, selectively combining audio and linguistic representations via the self-attention mechanism. However, BERT-CTC is a novel direction in which we seek to integrate BERT into a CTC-based model in a theoretically-sound manner.

5 Experiments

We used the ESPnet toolkit (Watanabe et al., 2018) for all the experiments. All the implementations and recipes are made publicly available (see §1).

5.1 Tasks and Datasets

Speech Recognition We evaluated models on the LibriSpeech (Panayotov et al., 2015), TED-LIUM2 (Rousseau et al., 2014) and AISHELL-1 (Bu et al., 2017) datasets. LibriSpeech consists of read English speech from audiobooks, and we used *train-clean-100* for training. TED-LIUM2 contains spontaneous English speech from Ted Talks. AISHELL-1 consists of read Mandarin speech.

Spoken Language Understanding We also evaluated our model on the SLURP dataset (Bastianelli et al., 2020). SLURP consists of English prompts of an in-home personal robot assistant, and we focused on the intent classification task.

We used the standard development and test sets for tuning hyper-parameters and evaluating performance for each dataset. Full dataset descriptions are in Appendix D.1.

5.2 End-to-End ASR Models

CTC (baseline): A model trained based on the CTC loss $\mathcal{L}_{\mathsf{ctc}}$ (see §2.1). Given the recent advances in CTC-based modeling (Higuchi et al., 2021a), we built a strong baseline using the intermediate CTC technique (Tjandra et al., 2020; Lee and Watanabe, 2021), which applies an auxiliary CTC loss to intermediate outputs of the audio encoder. We used the intermediate loss in a hierarchical manner (Sanabria and Metze, 2018), where the loss is calculated using a target sequence tokenized with a smaller vocabulary (i.e., $\mathcal{V}'$ in §3.1).
RNN-T (baseline): A model trained based on the RNN-T loss $\mathcal{L}_{\mathsf{rnnt}}$ (see §2.2). Considering the recent techniques developed upon multi-task learning (Boyer et al., 2021), we trained a strong model using an auxiliary CTC loss applied to the audio encoder output (Jeon and Kim, 2021). Same as CTC, we enhanced the audio encoder with intermediate CTC (Lee et al., 2022). All the CTC losses were calculated using the smaller-vocabulary sequence.
BERT-CTC (ours): The proposed model trained based on the BERT-CTC loss (Eq. (22)). As in the other models, we adopted intermediate CTC for the audio encoder. All the CTC losses were calculated using the smaller-vocabulary sequence.

See Appendices B and C for intermediate CTC and detailed model descriptions, respectively.

5.3 Experimental Settings

Model Configuration For the audio encoder, we adopted the Conformer architecture (Gulati et al., 2020), which consisted of 12 encoder blocks. The prediction network in RNN-T was a single long short-term memory (LSTM) layer. The self-attention module in BERT-CTC had 6 Transformer encoder blocks, and we used a $\text{BERT}_{\text{BASE}}$ model

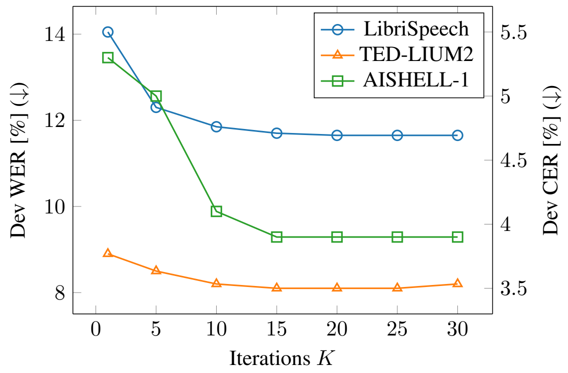


Figure 2: BERT-CTC results on development sets, using different number of decoding iterations.

provided by HuggingFace (Wolf et al., 2020).

Tokenization For each language, we used the same vocabulary as BERT for tokenizing target texts. We also constructed a smaller-sized vocabulary $\mathcal{V}'$ for the hierarchical losses, which is obtained by applying the byte pair encoding-based algorithm (Sennrich et al., 2016) to the transcription of each dataset.

Training We mostly followed ESPnet recipes provided for each dataset. For BERT-CTC, we set λ_{ctc} (in Eq. (22)) to 0.3 for all the ASR tasks and λ_{slu} (in Eq. (27)) to 1.0 for the SLU task.

Inference For CTC, we performed the best path decoding (§2.1). For RNN-T, we used the beam search decoding (§2.2) with a beam size of 20. For BERT-CTC, unless otherwise indicated, the number of iterations K was always set to 20 (§3.2).

Detailed experimental settings for reproducibility are in Appendix D.

6 Results

Speech Recognition Table 1 shows results on LibriSpeech-100h and TED-LIUM2 in word error rate (WER), and AISHELL-1 in character error rate (CER). While RNN-T slightly outperformed CTC on several evaluation sets in LibriSpeech-100h and AISHELL-1, CTC resulted in better performance on TED-LIUM2. RNN-T was ineffective at training ASR with the BERT vocabulary, particularly when a severe mismatch exists against the target ASR domain (i.e., Wikipedia vs. lecture). BERT-CTC significantly outperformed the baselines, consistently achieving the best results on all datasets. BERT-CTC improved over RNN-T, and we attribute this to not only considering the whole context of the target sequence but also using the

Model	LibriSpeech-100h				TED-LIUM2		AISHELL-1	
	Dev WER (↓)		Test WER (↓)		Dev WER (↓)	Test WER (↓)	Dev CER (↓)	Test CER (↓)
	clean	other	clean	other				
CTC†	11.2	21.4	11.4	22.0	9.9	9.3	5.1	5.6
RNN-T†	9.7	21.5	9.8	22.2	10.2	9.6	5.2	5.5
BERT-CTC	**7.0**	**16.3**	**7.2**	**16.6**	**8.1**	**7.6**	**3.9**	**3.9**

Table 1: WER [%] on LibriSpeech-100h and TED-LIUM2, and CER [%] on AISHELL-1. † indicates that the models are slightly different from the original CTC or RNN-T in that they are trained with hierarchical CTC loss.

Model	WER (↓)	Acc. (↑)
ESPnet-SLU (Arora et al., 2022)	–	86.3
ASR + BERT (Arora et al., 2022)	–	85.7
BERT-CTC ($K = 1$)	19.1	87.0
BERT-CTC ($K = 20$)	**18.2**	**87.8**

Table 2: WER [%] and classification accuracy [%] on SLURP intent classification task.

Model	Dev WER (↓)	Test WER (↓)
CTC†	**11.2 / 21.4**	**11.4 / 22.0**
w/o hierarchical loss	11.8 / 23.2	12.2 / 24.1
RNN-T†	**9.7 / 21.5**	**9.8 / 22.2**
w/o hierarchical loss	11.4 / 24.6	11.5 / 25.8
BERT-CTC	**7.0 / 16.3**	**7.2 / 16.6**
w/o hierarchical loss	8.6 / 19.1	8.9 / 19.5
w/o BERT	7.4 / 17.2	7.4 / 17.7

Table 3: Ablation studies on LibriSpeech-100h.

powerful representations from BERT, which we further analyze later. In Appendix E, we compare our AISHELL-1 results to those from recent works and show that our approach is on par with the state-of-the-art (Zheng et al., 2021) with fewer parameters. Figure 2 illustrates the correlation between BERT-CTC results and the number of decoding iterations. When decoded with $K = 1$, the model only uses speech input to predict a token sequence. By increasing K, the model beneficially exploited the BERT knowledge for refining the output tokens.

Spoken Language Understanding Table 2 lists the results of the SLURP intent classification task, evaluated in accuracy. We refer to the ESPnet-SLU (Arora et al., 2022) result as a baseline, which performs SLU along with ASR by prepending an intent label to the corresponding output sequence. We also refer to the ESPnet-SLU result obtained by stacking BERT on top of an ASR model, which was found to be less effective. BERT-CTC outperformed the baselines by effectively incorporating acoustic and linguistic information. By decoding in a single iteration ($K = 1$), BERT-CTC predicted an intent only from speech, and the accuracy was already higher than those of baselines. We observed a slight but clear gain by increasing K, which improved both ASR and SLU performance thanks to BERT. We note that our result outperforms the state-of-the-art 86.9% reported in (Seo et al., 2022).

7 Analyses

7.1 Ablation Studies

To validate the effectiveness of our model design for BERT-CTC, we conduct ablation studies (Table 3) on the usage of hierarchical loss and BERT.

Hierarchical Loss We observed that hierarchical CTC helped all the models improve their performance by a large margin. As the vocabulary of BERT is generally too large for E2E-ASR, the hierarchical modeling was crucial for predicting the sparse word-level tokens. Moreover, the result indicates that the hierarchical loss is effective for training an ASR model with a vocabulary from a different domain, as there is a non-negligible domain-mismatch between the BERT training text and ASR transcription.

BERT To ablate BERT-CTC with BERT, we replaced $\text{BERT}(\cdot)$ in Eq. (19) with a simple embedding layer with positional encoding. We found that removing BERT led to degradation in BERT-CTC performance, which supports the importance of using BERT. However, interestingly, the result was still better than the baselines, indicating the advantage over RNN-T in that BERT-CTC is capable of considering the bi-directional context.

7.2 Error Analysis with Decoding Example

Table 4 shows a process of BERT-CTC inference, decoding an utterance in the LibriSpeech test set.

$k=1$	... thou a gave meet any one afterter these hour recite aught of courtry whether he be ne'er ...
$k=10$	... thou a again meet any one afterter these hour reciteiting aught of poetryry whether he be near'er ...
$k=15$	... thou again meet any one after this hour reciteiting aught of poetryry whether he be near'or ...
$k=20$	... thou again meet any one after this hour reciteiting aught of poetry whether he be near ...
w/o BERT	... thou a gag meet any one after this hour residing aught of boy whether he be near ...
Reference	... thou again meet any one after this hour reciting aught of poetry whether he be near ...

Table 4: Decoding example from LibriSpeech test-other set (2033-164914-0016). At each iteration, the highlighted tokens are masked and repredicted in the next iteration. Blue indicates refined tokens, and red indicates ones not.

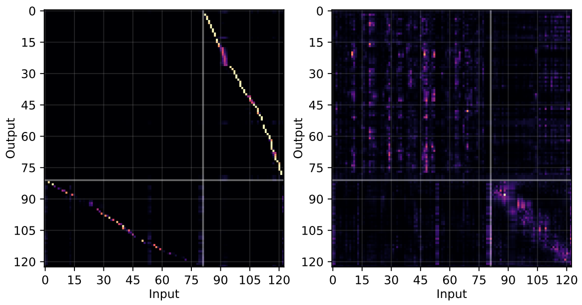


Figure 3: Attention visualization for BERT-CTC. White lines indicate the boundaries of audio and token seqs.

In the output sequence at $k = 1$, the model mistakenly predicted phonetically similar tokens (e.g., "again"→"a gave", "near"→"ne'er"). At the first iteration, the model was only conditioned on acoustic information, making it challenging to determine target tokens accurately. As the iteration proceeded, the model corrected the most errors by considering the output dependency. Unlike the original mask-predict algorithm (Ghazvininejad et al., 2019), our approach permits for flexibly adjusting the target length, enabling the model to resolve insertion and deletion errors (e.g., "afterer"→"after"). We also show an example obtained w/o BERT (from Table 3), which failed to recover tokens that were correctly recognized by BERT-CTC with BERT.

7.3 Conditional Independence of $p(W|\tilde{W}, \emptyset)$

We empirically validate the conditional independence assumption made in Eq. (15), where the output sequence W depends only on its masked sequence $\tilde{W}$ without audio information O. To this end, we augmented the BERT module by inserting adaptive cross-attention layers, which is similar to Adapter-BERT Networks (Guo et al., 2020). These additional layers are trained to infuse the audio encoder output H^{ae} into each BERT layer, thereby allowing BERT-CTC to realize $p(W|\tilde{W}, O)$. When evaluated on LibriSpeech, the modified BERT-CTC resulted in 7.2%/17.9% on the dev. set and 7.3%/18.0% on the test set, which are worse than the results in Table 1. This indicates that BERT already captures sophisticated linguistic information and does not require extra parameters for adapting BERT to audio input.

7.4 Attention Visualization

Figure 3 depicts example attention weight matrices, produced by the second self-attention layer of BERT-CTC. We observed two major attention patterns: weights aligning audio and token sequences by capturing inter-dependencies (Fig. 3 left) and weights attending inter-dependencies within each sequence (Fig. 3 right). These patterns support our motivation for the BERT-CTC design in learning inner/inter dependencies within/between the audio and token representations.

7.5 Inference Speed Comparison

To see how the iterative decoding with BERT affects the inference speed of BERT-CTC, we evaluated each model on the real-time factor (RTF). RTF was measured on the LibriSpeech test-other set using a single GPU with a batchsize of 1 or a single CPU. RTFs for GPU / CPU inference resulted in 7.91e-3 / 4.18e-2 for CTC, 4.81e-1 / 4.55 for RNN-T, and 9.72e-2 / 7.22e-1 for BERT-CTC. The semi-autoregressive characteristic in BERT-CTC enabled faster inference than autoregressive RNN-T and provided further speedup with the parallel computing using GPU.

8 Conclusion

We proposed BERT-CTC that leverages BERT for relaxing the conditional independence assumption in CTC. BERT-CTC uses BERT as contextual embedding to explicitly condition CTC training/inference on linguistic information. Experimental results showed that BERT-CTC improved over conventional approaches. Moreover, we confirmed that BERT-CTC is applicable to end-to-end SLU.

$\mathcal{V}$	**Model**	**Dev WER** (↓) clean / other	**Test WER** (↓) clean / other
$\mathcal{V}^{\text{asr}}$	CTC†	6.9 / 20.1	7.0 / 20.2
$\mathcal{V}^{\text{asr}}$	RNN-T†	**5.7** / 17.0	**6.0** / 17.2
$\mathcal{V}^{\text{bert}}$	CTC†	11.2 / 21.4	11.4 / 22.0
$\mathcal{V}^{\text{bert}}$	RNN-T†	9.7 / 21.5	9.8 / 22.3
$\mathcal{V}^{\text{bert}}$	BERT-CTC	7.0 / **16.3**	7.2 / **16.6**

Table 5: WER [%] on LibriSpeech-100h. $\mathcal{V}^{\text{asr}}$ indicates a subword vocabulary constructed from ASR transcriptions, where $|\mathcal{V}^{\text{asr}}| = 300$. $\mathcal{V}^{\text{bert}}$ indicates the BERT vocabulary, where $|\mathcal{V}^{\text{bert}}| = 30522$.

Limitations

Vocabulary Constraint The output unit of BERT-CTC is constrained to the vocabulary of BERT, which is likely to be not generalized to an ASR domain and too sparse for ASR training. Table 5 shows results on LibriSpeech-100h with different vocabularies, where $\mathcal{V}^{\text{asr}}$ is an ASR vocabulary with a vocabulary size of 300 constructed from LibriSpeech transcriptions, and $\mathcal{V}^{\text{bert}}$ is the BERT vocabulary with a vocabulary size of 30522. We observed that, by using $\mathcal{V}^{\text{asr}}$, the performance of CTC and RNN-T improved over the results using $\mathcal{V}^{\text{bert}}$ and closed the gap with the BERT-CTC results. We believe that using a BERT variant with a smaller vocabulary, e.g., CharacterBERT (El Boukkouri et al., 2020) improves BERT-CTC further.

Computational Cost BERT-CTC requires a high computational cost, especially during inference, due to the iterative forward calculations of BERT (i.e., $K = 20$ times) with the $\mathcal{O}(N^2)$ computational and memory complexities in the self-attention layers. Still, GPUs can greatly accelerate the inference speed, and BERT-CTC can alternatively use other pre-trained MLMs with lighter weights, e.g., ALBERT (Lan et al., 2019) and DistilBERT (Sanh et al., 2019).

Non-streaming BERT-CTC is not suited for online streaming scenarios, where output tokens are predicted synchronously to sequential speech input. It is not a significant problem when we consider applying BERT-CTC to utterance-level ASR tasks, such as end-to-end SLU as we demonstrated the capability of BERT-CTC (Table 2). Otherwise, we can adopt existing techniques for making BERT-CTC streaming, e.g., causal masking (Vaswani et al., 2017), time-restricted attention (Povey et al., 2018), and block-wise processing (Tsunoo et al., 2019). Another solution can be to apply the two-pass algorithm (Sainath et al., 2019), where BERT-CTC first performs streaming recognition at $k = 1$ and then refines the outputs using the full context information at $k > 1$.

Acknowledgements

This work was supported in part by JST ACT-X (JPMJAX210J) and JSPS KAKENHI (JP21J23495). This work used the Extreme Science and Engineering Discovery Environment (XSEDE) (Towns et al., 2014) supported by National Science Foundation grant number ACI-1548562. It uses the Bridges system (Nystrom et al., 2015) supported by NSF award number ACI-1445606, at the Pittsburgh Supercomputing Center (PSC).

References

Siddhant Arora, Siddharth Dalmia, Pavel Denisov, Xuankai Chang, Yushi Ueda, Yifan Peng, Yuekai Zhang, Sujay Kumar, Karthik Ganesan, Brian Yan, Ngoc Thang Vu, Alan W Black, and Shinji Watanabe. 2022. ESPnet-SLU: Advancing spoken language understanding through ESPnet. In *Proceedings of the 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7167–7171.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICML)*.

Ye Bai, Jiangyan Yi, Jianhua Tao, Zhengkun Tian, Zhengqi Wen, and Shuai Zhang. 2021. Fast end-to-end speech recognition via non-autoregressive models and cross-modal knowledge transferring from BERT. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:1897–1911.

Emanuele Bastianelli, Andrea Vanzo, Pawel Swietojanski, and Verena Rieser. 2020. SLURP: A spoken language understanding resource package. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7252–7262.

Florian Boyer, Yusuke Shinohara, Takaaki Ishii, Hirofumi Inaguma, and Shinji Watanabe. 2021. A study of Transducer based end-to-end ASR with ESPnet: Architecture, auxiliary loss and decoding strategies. In *Proceedings of the 2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 16–23.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda

Askell, et al. 2020. Language models are few-shot learners. In *Proceedings of Advances in Neural Information Processing Systems 33 (NeurIPS)*, pages 1877–1901.

Hui Bu, Jiayu Du, Xingyu Na, Bengu Wu, and Hao Zheng. 2017. AISHELL-1: An open-source Mandarin speech corpus and a speech recognition baseline. In *Proceedings of the 20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA)*, pages 1–5.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964.

William Chan, Chitwan Saharia, Geoffrey Hinton, Mohammad Norouzi, and Navdeep Jaitly. 2020. Imputer: Sequence modelling via imputation and dynamic programming. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 1403–1413.

Nanxin Chen, Shinji Watanabe, Jesus Antonio Villalba, Piotr Zelasko, and Najim Dehak. 2020. Non-autoregressive Transformer for speech recognition. *IEEE Signal Processing Letter*.

Nanxin Chen, Piotr Żelasko, Laureano Moro-Velázquez, Jesús Villalba, and Najim Dehak. 2021. Align-Denoise: Single-pass non-autoregressive speech recognition. In *Proceedings of Interspeech 2021*, pages 3770–3774.

Ethan A Chi, Julian Salazar, and Katrin Kirchhoff. 2021. Align-Refine: Non-autoregressive speech recognition via iterative realignment. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1920–1927.

Chung-Cheng Chiu, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Ekaterina Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, and Michiel Bacchiani. 2018. State-of-the-art speech recognition with sequence-to-sequence models. In *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4774–4778.

Shih-Hsuan Chiu and Berlin Chen. 2021. Innovative BERT-based reranking language models for speech recognition. In *Proceedings of the 2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 266–271.

Jan Chorowski and Navdeep Jaitly. 2017. Towards better decoding and language model integration in sequence to sequence models. In *Proceedings of Interspeech 2017*, pages 523–527.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Proceedings of Advances in Neural Information Processing Systems 28 (NeurIPS)*, pages 577–585.

Yung-Sung Chuang, Chi-Liang Liu, Hung yi Lee, and Lin shan Lee. 2020. SpeechBERT: An audio-and-text jointly learned language model for end-to-end spoken question answering. In *Proceedings of Interspeech 2020*, pages 4168–4172.

Yu-An Chung, Chenguang Zhu, and Michael Zeng. 2021. SPLAT: Speech-language joint pre-training for spoken language understanding. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1897–1907.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(76):2493–2537.

Keqi Deng, Songjun Cao, Yike Zhang, and Long Ma. 2021. Improving hybrid CTC/attention end-to-end speech recognition with pretrained acoustic and language models. In *Proceedings of the 2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 76–82.

Keqi Deng, Zehui Yang, Shinji Watanabe, Yosuke Higuchi, Gaofeng Cheng, and Pengyuan Zhang. 2022. Improving non-autoregressive end-to-end speech recognition with pre-trained acoustic and language models. In *Proceedings of the 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8522–8526.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional Transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186.

Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun'ichi Tsujii. 2020. CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6903–6915.

Santiago Fernández, Alex Graves, and Jürgen Schmidhuber. 2007. Sequence labelling in structured domains with hierarchical recurrent neural networks. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 774–779.

Yuya Fujita, Shinji Watanabe, Motoi Omachi, and Xuankai Chang. 2020. Insertion-based modeling for end-to-end automatic speech recognition. In *Proceedings of Interspeech 2020*, pages 3660–3664.

Hayato Futami, Hirofumi Inaguma, Masato Mimura, Shinsuke Sakai, and Tatsuya Kawahara. 2021. ASR rescoring and confidence estimation with ELECTRA. In *Proceedings of the 2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 380–387.

Hayato Futami, Hirofumi Inaguma, Sei Ueno, Masato Mimura, Shinsuke Sakai, and Tatsuya Kawahara. 2020. Distilling the knowledge of BERT for sequence-to-sequence ASR. In *Proceedings of Interspeech 2020*, pages 3635–3639.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6114–6123.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 369–376.

Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning (ICML)*, pages 1764–1772.

Alex Graves, Abdelrahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein Transformer. In *Proceedings of Advances in Neural Information Processing Systems 32 (NeurIPS)*.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. 2020. Conformer: Convolution-augmented Transformer for speech recognition. In *Proceedings of Interspeech 2020*, pages 5036–5040.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Junliang Guo, Zhirui Zhang, Linli Xu, Hao-Ran Wei, Boxing Chen, and Enhong Chen. 2020. Incorporating BERT into parallel sequence decoding with adapters. In *Proceedings of Advances in Neural Information Processing Systems 33 (NeurIPS)*, pages 10843–10854.

Pengcheng Guo, Florian Boyer, Xuankai Chang, Tomoki Hayashi, Yosuke Higuchi, Hirofumi Inaguma, Naoyuki Kamo, Chenda Li, Daniel Garcia-Romero, Jiatong Shi, Jing Shi, Shinji Watanabe, Kun Wei, Wangyou Zhang, and Yuekai Zhang. 2021. Recent developments on ESPnet toolkit boosted by Conformer. In *Proceedings of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5874–5878.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. 2014. Deep Speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

Tomoki Hayashi, Shinji Watanabe, Tomoki Toda, Kazuya Takeda, Shubham Toshniwal, and Karen Livescu. 2019. Pre-trained text embeddings for enhanced text-to-speech synthesis. In *Proceedings of Interspeech 2019*, pages 4430–4434.

Yosuke Higuchi, Nanxin Chen, Yuya Fujita, Hirofumi Inaguma, Tatsuya Komatsu, Jaesong Lee, Jumon Nozaki, Tianzi Wang, and Shinji Watanabe. 2021a. A comparative study on non-autoregressive modelings for speech-to-text generation. In *Proceedings of the 2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 47–54.

Yosuke Higuchi, Hirofumi Inaguma, Shinji Watanabe, Tetsuji Ogawa, and Tetsunori Kobayashi. 2021b. Improved mask-CTC for non-autoregressive end-to-end ASR. In *Proceedings of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8363–8367.

Yosuke Higuchi, Keita Karube, Tetsuji Ogawa, and Tetsunori Kobayashi. 2022. Hierarchical conditional end-to-end ASR with CTC and multi-granular subword units. In *Proceedings of the 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7797–7801.

Yosuke Higuchi, Shinji Watanabe, Nanxin Chen, Tetsuji Ogawa, and Tetsunori Kobayashi. 2020. Mask CTC: Non-autoregressive end-to-end ASR with CTC and mask predict. In *Proceedings of Interspeech 2020*, pages 3655–3659.

Wen-Chin Huang, Chia-Hua Wu, Shang-Bao Luo, Kuan-Yu Chen, Hsin-Min Wang, and Tomoki Toda. 2021.

Speech recognition by simply fine-tuning BERT. In *Proceedings of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7343–7347.

Jae-Jin Jeon and Eesung Kim. 2021. Multitask learning and joint optimization for Transformer-RNN-Transducer speech recognition. In *Proceedings of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6793–6797.

Anjuli Kannan, Yonghui Wu, Patrick Nguyen, Tara N Sainath, Zhijeng Chen, and Rohit Prabhavalkar. 2018. An analysis of incorporating an external language model into a sequence-to-sequence model. In *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5828.

Tom Kenter, Manish Sharma, and Rob Clark. 2020. Improving the prosody of RNN-based english text-to-speech synthesis by incorporating a BERT model. In *Proceedings of Interspeech 2020*, pages 4412–4416.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. Audio augmentation for speech recognition. In *Proceedings of Interspeech 2015*, pages 3586–3589.

Yotaro Kubo, Shigeki Karita, and Michiel Bacchiani. 2022. Knowledge transfer from large-scale pretrained language models to end-to-end speech recognizers. In *Proceedings of the 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8512–8516.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 66–75.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Jaesong Lee, Jingu Kang, and Shinji Watanabe. 2021. Layer pruning on demand with intermediate CTC. In *Proceedings of Interspeech 2021*, pages 3745–3749.

Jaesong Lee, Lukas Lee, and Shinji Watanabe. 2022. Memory-efficient training of RNN-Transducer with sampled softmax. In *Proceedings of Interspeech 2022*.

Jaesong Lee and Shinji Watanabe. 2021. Intermediate loss regularization for CTC-based speech recognition. In *Proceedings of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6224–6228.

Jumon Nozaki and Tatsuya Komatsu. 2021. Relaxing the conditional independence assumption of CTC-based ASR by conditioning on intermediate predictions. In *Proceedings of Interspeech 2021*, pages 3735–3739.

Nicholas A Nystrom, Michael J Levine, Ralph Z Roskies, and J Ray Scott. 2015. Bridges: A uniquely flexible HPC resource for new communities and data analytics. In *Proceedings of XSEDE*, pages 1–8.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An ASR corpus based on public domain audio books. In *Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210.

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. 2019. SpecAugment: A simple data augmentation method for automatic speech recognition. In *Proceedings of Interspeech 2019*, pages 2613–2617.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Proceedings of Advances in Neural Information Processing Systems 32 (NeurIPS)*.

Daniel Povey, Hossein Hadian, Pegah Ghahremani, Ke Li, and Sanjeev Khudanpur. 2018. A time-restricted self-attention layer for ASR. In *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5874–5878.

Anthony Rousseau, Paul Deléglise, and Yannick Estève. 2014. Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*, pages 3935–3939.

Tara N. Sainath, Ruoming Pang, David Rybach, Yanzhang He, Rohit Prabhavalkar, Wei Li, Mirkó Visontai, Qiao Liang, Trevor Strohman, Yonghui Wu, Ian McGraw, and Chung-Cheng Chiu. 2019. Two-pass end-to-end speech recognition. In *Proceedings of Interspeech 2019*, pages 2773–2777.

Julian Salazar, Davis Liang, Toan Q Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2699–2712.

Ramon Sanabria and Florian Metze. 2018. Hierarchical multitask learning with CTC. In *Proceedings of the 2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 485–490.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1715–1725.

Seunghyun Seo, Donghyun Kwak, and Bowon Lee. 2022. Integration of pre-trained networks with continuous token interface for end-to-end spoken language understanding. In *Proceedings of the 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7152–7156.

Changhao Shan, Chao Weng, Guangsen Wang, Dan Su, Min Luo, Dong Yu, and Lei Xie. 2019. Component fusion: Learning replaceable language model component for end-to-end speech recognition system. In *Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5361–5635.

Joonbo Shin, Yoonhyung Lee, and Kyomin Jung. 2019. Effective sentence scoring method using BERT for speech recognition. In *Proceedings of Asian Conference on Machine Learning (ACML)*, pages 1081–1093.

Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. 2018. Cold fusion: Training seq2seq models together with language models. In *Proceedings of Interspeech 2018*, pages 387–391.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of Advances in Neural Information Processing Systems 27 (NeurIPS)*, pages 3104–3112.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4593–4601.

Andros Tjandra, Chunxi Liu, Frank Zhang, Xiaohui Zhang, Yongqiang Wang, Gabriel Synnaeve, Satoshi Nakamura, and Geoffrey Zweig. 2020. DEJA-VU: Double feature presentation and iterated loss in deep Transformer networks. In *Proceedings of the 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6899–6903.

J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. R. Scott, and N. Wilkins-Diehr. 2014. XSEDE: Accelerating scientific discovery. *Computing in Science & Engineering*, 16(5):62–74.

Emiru Tsunoo, Yosuke Kashiwagi, Toshiyuki Kumakura, and Shinji Watanabe. 2019. Transformer ASR with contextual block processing. In *Proceedings of the 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 427–433.

Takuma Udagawa, Masayuki Suzuki, Gakuto Kurata, Nobuyasu Itoh, and George Saon. 2022. Effect and analysis of large-scale language model rescoring on competitive ASR systems. In *Proceedings of Interspeech 2022*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 5998–6008.

Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai. 2018. ESPnet: End-to-end speech processing toolkit. In *Proceedings of Interspeech 2018*, pages 2207–2211.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*, pages 38–45.

Cheng Yi, Shiyu Zhou, and Bo Xu. 2021. Efficiently fusing pretrained acoustic and linguistic encoders for low-resource speech recognition. *IEEE Signal Processing Letters*, 28:788–792.

Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2018. Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine*, 13(3):55–75.

Fu-Hao Yu, Kuan-Yu Chen, and Ke-Han Lu. 2022. Non-autoregressive ASR modeling using pre-trained language models for Chinese speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:1474–1482.

Guolin Zheng, Yubei Xiao, Ke Gong, Pan Zhou, Xiaodan Liang, and Liang Lin. 2021. Wav-BERT: Cooperative acoustic and linguistic representation learning for low-resource speech recognition. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2765–2777.

Algorithm 1 BERT-CTC Inference

Input: The number of iterations K, audio encoder output H^{ae}

1: $\hat{A}' = \mathrm{argmax}_{A'}\, p(A'|O)$ ▷ Obtain the most probable alignment from the audio encoder
2: $\hat{W}' = \mathcal{B}_{\mathsf{ctc}}(\hat{A}')$
3: $\hat{N} \sim |\hat{W}'|$ ▷ Obtain the target length from the intermediate prediction
4: $\bar{W} = (w_n = \texttt{[MASK]} | n = 1, \cdots, \hat{N})$ ▷ Initialize a masked sequence
5: **for** $k = 1, \ldots, K$ **do**
6: # Token prediction
7: $H^{\mathsf{bert}} = \mathrm{BERT}(\bar{W})$ ▷ Forward BERT
8: $p(A|\cdot) = \mathrm{Softmax}(\mathrm{SelfAttn}(H^{\mathsf{ae}}, H^{\mathsf{bert}}))$ ▷ Forward self-attention module
9: $\hat{A} = \mathrm{argmax}_{A}\, p(A|\cdot)$ ▷ Obtain the most probable alignment
10: $\hat{W} = \mathcal{B}_{\mathsf{ctc}}(\hat{A})$
11: # Token-level probability calculation
12: $\hat{P} = (\hat{p}_n = 0 | n = 1, \cdots, |\hat{W}|)$ ▷ Initialize *token-level* probabilities
13: $n = 1$ ▷ Initialize an index for token position
14: $a_0 = \epsilon$
15: **for** $t = 1, \ldots, T$ **do**
16: **if** $\hat{a}_t = \epsilon$ **then**
17: **if** $\hat{a}_{t-1} \neq \epsilon$ **then**
18: $n = n + 1$
19: **end if**
20: **else**
21: $\hat{p}_n = \max(p(a_t = \hat{w}_n|\cdot), \hat{p}_n)$ ▷ Keep the maximum probability for each token
22: **end if**
23: **end for**
24: # Token masking
25: $M = \lfloor |\hat{W}| \cdot \frac{K-k}{K} \rfloor$ ▷ Calculate the number of masked tokens
26: $\bar{W} = \mathrm{MaskLowestProb}(\hat{W}, \hat{P}, M)$ ▷ Mask tokens with the M lowest probability scores
27: **end for**
28: **return** $\hat{W}$

A Inference Algorithm

Algorithm 1 describes the overall process of BERT-CTC inference. For estimating the target length in line 3, at the implementation level, we first decode $\hat{W}'$ into a sentence, which is then tokenized using the BERT vocabulary, and the length of the resulting sequence is used as the target length. In lines 12–25, before the token masking step, we calculate a probability score $\hat{p}_n$ for each token $\hat{w}_n$ in the estimated output sequence $\hat{W}$. This score calculation simply takes the maximum value in frame-level token probabilities that correspond to a predicted token $\hat{w}_n$ after the collapsing operation. In line 28, given the probability scores, $\mathrm{MaskLowestProb}(\cdot)$ masks tokens in $\hat{W}$ with the M lowest scores.

B Intermediate CTC

Intermediate CTC (Tjandra et al., 2020; Lee and Watanabe, 2021) applies additional CTC losses to intermediate layers of the audio encoder network. Let $H^{(e)} = (\boldsymbol{h}_t^{(e)} \in \mathbb{R}^{d^{\mathsf{ae}}} | t = 1, \cdots, T)$ be an intermediate output of the e-th layer of the audio encoder, which is computed as in Eq. (6) as

$$\mathbf{h}_t^{(e)} \sim \mathrm{AudioEnc}^{(e)}(O), \tag{28}$$

where $\mathrm{AudioEnc}^{(e)}(\cdot)$ indicates the e-th layer output of the audio encoder. Similar to Eq. (5), token emission probabilities at each time frame is computed based on Eq. (28) as

$$p^{(e)}(a_t|O) = \mathrm{Softmax}(\mathrm{Linear}(\mathbf{h}_t^{(e)})). \tag{29}$$

Finally, an intermediate CTC loss $\mathcal{L}_{\text{ctc}}^{(e)}$ is defined similarly to Eq. (7) as

$$\mathcal{L}_{\text{ctc}}^{(e)}(O, W) = -\log \sum_{A \in \mathcal{B}_{\text{ctc}}^{-1}(W)} \prod_{t=1}^{T} p^{(e)}(a_t|O). \tag{30}$$

C Model Details

C.1 CTC (baseline)

We applied the intermediate CTC loss to the 6-th layer of the audio encoder, which is calculated using the smaller-vocabulary sequence W' in a hierarchical multi-tasking manner. Using the intermediate loss, the CTC loss $\mathcal{L}_{\text{ctc}}$ is extended as

$$(1 - \lambda_{\text{ic}})\mathcal{L}_{\text{ctc}}(O, W) + \lambda_{\text{ic}}\mathcal{L}_{\text{ctc}}^{(e=6)}(O, W'), \tag{31}$$

where λ_{ic} is a tunable weight for the intermediate loss, and we equally weighted each loss (i.e., $\lambda_{\text{ic}} = 0.5$) as in (Lee et al., 2021; Higuchi et al., 2022).

C.2 RNN-T (baseline)

We applied auxiliary CTC losses to the final and intermediate layer of the audio encoder. As in Eq. (31), the intermediate loss was applied to the 6-th layer. With the additional CTC losses, the RNN-T loss $\mathcal{L}_{\text{rnnt}}$ is extended as

$$\begin{aligned}&(1 - \lambda_{\text{ctc}})\mathcal{L}_{\text{rnnt}}(O, W)\\ &+\lambda_{\text{ctc}}\{(1-\lambda_{\text{ic}})\mathcal{L}_{\text{ic}}(O, W')+\lambda_{\text{ic}}\mathcal{L}_{\text{ctc}}^{(e=6)}(O, W')\},\end{aligned} \tag{32}$$

where λ_{ctc} is a tunable weight for CTC losses, and we set $\lambda_{\text{ctc}} = 0.3$ as in (Boyer et al., 2021). Note that all the CTC losses were calculated using the smaller-vocabulary sequence W' in a hierarchical multi-tasking manner.

C.3 BERT-CTC (ours)

We applied auxiliary CTC losses to the final and intermediate layer of the audio encoder. As in Eq. (31), the intermediate loss was applied to the 6-th layer. With the additional CTC losses, the BERT-CTC loss $\mathcal{L}_{\text{bc}}$ is extended as

$$\begin{aligned}&(1 - \lambda_{\text{ctc}})\mathcal{L}_{\text{bc}}(O, W)\\ &+\lambda_{\text{ctc}}\{(1-\lambda_{\text{ic}})\mathcal{L}_{\text{ic}}(O, W')+\lambda_{\text{ic}}\mathcal{L}_{\text{ctc}}^{(e=6)}(O, W')\},\end{aligned} \tag{33}$$

where λ_{ctc} is a tunable weight for CTC losses, and we set $\lambda_{\text{ctc}} = 0.3$. Note that all the CTC losses were calculated using the smaller-vocabulary sequence W' in a hierarchical multi-tasking manner (as explained in §3.1).

D Experimental Details

D.1 Dataset

Tables 6 and 7 list descriptions of ASR and SLU datasets, respectively. Data preparation was done using the ESPnet2 recipe provided for each dataset: LibriSpeech-100h[5], TED-LIUM2[6], AISHELL-1[7], SLURP[8].

D.2 Model Configuration

For the audio encoder network, we used the Conformer (Gulati et al., 2020)-based encoder architecture implemented in ESPnet (Guo et al., 2021). The audio encoder consisted of 2 or 3 convolutional neural network (CNN) layers followed by a stack of 12 encoder blocks. The dimensions of the self-attention layer d^{ae} and feed-forward network d^{ff} were set to 256 and 1024, respectively, and the number of heads d^{head} was set to 4. The kernel size of depthwise separable convolution was set to 31. For RNN-T, the prediction network was a single LSTM layer with 512 units, and the joint network was a single linear layer with 640 units. For BERT-CTC, we built the Transformer (Vaswani et al., 2017)-based architecture for the self-attention module, which consisted of a stack of 6 encoder blocks with $d^{\text{model}} = 256$, $d^{\text{ff}} = 2048$, and $d^{\text{head}} = 4$. Before feeding into the self-attention module, the hidden vectors, H^{ae} and H^{bert}, were emebbeded using 2 CNN layers and a single linear layer, respectively, which mapped each vector to the dimension size of d^{model}. For the BERT module in BERT-CTC, we downloaded pre-trained models from the HuggingFace Transformers library (Wolf et al., 2020)[9]. We used a $\text{BERT}_{\text{BASE}}$ model provided for each language: English[10], Mandarin[11]. Note that the dimension of the BERT output d^{bert} was 768. The number of total/trainable parameters in the CTC, RNN-T, and BERT-CTC models was about 30M/30M, 60M/60M, and 150M/40M, respectively.

[5] https://github.com/espnet/espnet/tree/master/egs2/librispeech_100/asr1
[6] https://github.com/espnet/espnet/tree/master/egs2/tedlium2/asr1
[7] https://github.com/espnet/espnet/tree/master/egs2/aishell/asr1
[8] https://github.com/espnet/espnet/tree/master/egs2/slurp/asr1
[9] https://github.com/huggingface/transformers
[10] https://huggingface.co/bert-base-uncased
[11] https://huggingface.co/bert-base-chinese

Dataset	Language	Speech Style	# Train Hours	# Valid. Hours	# Test Hours
LibriSpeech-100h (Panayotov et al., 2015)	English	Read	100h	5.4h / 5.3h	5.4h / 5.1h
TED-LIUM2 (Rousseau et al., 2014)	English	Spontaneous	210h	1.6h	2.6h
AISHELL-1 (Bu et al., 2017)	Mandarin	Read	170h	10h	5h

Table 6: ASR dataset descriptions. The validation and test sets of LibriSpeech are split into "clean" / "other" sets based on the quality of the recordec utterances.

Dataset	Language	# Intents	# Train Hours	# Valid. Hours	# Test Hours
SLURP (Bastianelli et al., 2020)	English	69	40h + 43h	6.9h	10.3h

Table 7: Dataset description for SLURP intent classification. We bootstrap the train set with the synthetic data.

Hyperparameter	Value
Dropout rate	0.1
LR schedule	Noam (Vaswani et al., 2017)
Max learing rate	best of [1e-3, 2e-3]
Warmup steps	15k
Epochs	best of [50, 70, 100]
Adam betas	(0.9, 0.98)
Weight decay	1e-6

Table 8: Training configuration for CTC model.

Hyperparameter	Value
Dropout rate	0.1
LR schedule	Noam (Vaswani et al., 2017)
Max learing rate	best of [1e-3, 2e-3]
Warmup steps	15k
Epochs	best of [50, 70, 100]
Adam betas	(0.9, 0.98)
Weight decay	1e-6

Table 10: Training configuration for BERT-CTC model.

Hyperparameter	Value
Dropout rate	0.1
LR schedule	Noam (Vaswani et al., 2017)
Max learing rate	2e-3
Warmup steps	15k
Epochs	best of [50, 70]
Adam betas	(0.9, 0.98)
Weight decay	1e-6

Table 9: Training configuration for RNN-T model.

D.3 Tokenization

We used the same subword vocabulary as BERT for tokenizing target texts, where the vocabulary size $|\mathcal{V}|$ was 30522 for English and 21128 for Mandarin. For the smaller-sized vocabulary $\mathcal{V}'$ used in hierarchical CTC, we used SentencePiece (Kudo, 2018)[12] to construct subword vocabularies from transcription data in each training set. Following the ESPnet recipes, the vocabulary size was set to 300 for LibriSpeech-100h, and 500 for TED-LIUM2 and SLURP. For AISHELL-1, we used character-level tokenization with 4231 Chinese characters.

D.4 Training

All the models were implemented and trained using ESPnet (Watanabe et al., 2018)[13] and PyTorch (Paszke et al., 2019)[14]. In Tables 8, 9, and 10, we summarize training configurations for the CTC, RNN-T, and BERT-CTC models, respectively. We augmented speech data using speed perturbation (Ko et al., 2015) with a factor of 3 and SpecAugment (Park et al., 2019). For the hyperparameters in SpecAugment, we set the number of frequency and time masks to 2 and 5, and the size of frequency and time masks to 27 and $0.05T$. Note that the maximum size of the time mask depends on the utterance length T. After training, model parameters were averaged over 10 checkpoints with the best validation performance. For CTC, we trained models using a single RTX 2080 Ti GPU for 1 to 3 days, depending on the tasks and number of epochs. For RNN-T, we trained models using 4 V100 GPUs for 5 to 7 days, depending on the tasks and number of epochs. For BERT-CTC, we trained models using a single RTX 2080 Ti GPU for 3 to 5 days, depending on the tasks and number of epochs.

[12] https://github.com/google/sentencepiece
[13] https://github.com/espnet/espnet
[14] https://github.com/pytorch/pytorch

Model	#params [M] Total (Trainable)	Pre-trained AM	Pre-trained LM	Dev CER (↓)	Test CER (↓)
rePLM-NAR-ASR (Yu et al., 2022)	120 (120)	–	BERT	4.2	4.8
CTC/Attention (Deng et al., 2021)	161 (152)	wav2vec2.0	–	4.7	5.0
CTC/Attention (Deng et al., 2021)	218 (209)	wav2vec2.0	DistilGPT2	3.9	4.2
NAR-CTC/Attention (Deng et al., 2022)	204 (195)	wav2vec2.0	BERT	4.0	4.3
Wav-BERT (Zheng et al., 2021)	380 (380)	wav2vec2.0	BERT	**3.6**	**3.8**
BERT-CTC (ours)	143 (40)	–	BERT	3.9	3.9

Table 11: Comparison to prior works on AISHELL-1. The number of trainable parameters in BERT-CTC is fewer than in the others because BERT-CTC uses BERT without fine-tuning.

Model	#iters	LibriSpeech-100h Dev WER (↓) clean	LibriSpeech-100h Dev WER (↓) other	LibriSpeech-100h Test WER (↓) clean	LibriSpeech-100h Test WER (↓) other	TED-LIUM2 Dev WER (↓)	TED-LIUM2 Test WER (↓)
Mask-CTC (Higuchi et al., 2020)	10	7.2	20.3	7.5	20.6	8.9	8.5
Improved Mask-CTC (Higuchi et al., 2021b)	5	7.0	19.8	7.3	20.2	8.8	8.3
Align-Denoise (Chen et al., 2021)	1	8.0	22.3	8.4	22.5	9.0	8.7
Intermediate CTC (Lee and Watanabe, 2021)	1	6.9	19.7	7.1	20.2	8.5	8.3
Self-conditioned CTC (Nozaki and Komatsu, 2021)	1	**6.6**	19.4	**6.9**	19.7	8.7	8.0
KERMIT (Fujita et al., 2020)	$\simeq \log_2(N)$	7.1	19.7	7.4	20.2	9.1	8.2
BERT-CTC (ours)	20	7.0	**16.3**	7.2	**16.6**	**8.1**	**7.6**

Table 12: Comparison of BERT-CTC and non-autoregressive E2E-ASR models on LibriSpeech-100h and TED-LIUM2. The prior results are obtained from the comparative study conducted in (Higuchi et al., 2021a).

D.5 Inference

RTF was measured using a single V100 GPU (with a batchsize of 1) or a single Intel(R) Xeon(R) Gold 6148 CPU@2.4 GHz.

E Comparison to Prior Works

AISHELL-1 Table 11 lists results on AISHELL-1, comparing our BERT-CTC with recent approaches using a pre-trained acoustic model (AM) or/and LM. BERT-CTC achieved comparable performance to the state-of-the-art approach, Wav-BERT (Zheng et al., 2021), without using a pre-trained AM. Moreover, the number of trainable parameters in BERT-CTC was much fewer than in the other models because BERT-CTC used BERT as contextual embedding (without fine-tuning). We attribute this advantage of BERT-CTC to our well-defined formulation for conditioning CTC training/inference with BERT knowledge.

Non-autoregressive End-to-End ASR Table 12 compares our BERT-CTC with the previous non-autoregressive E2E-ASR models on LibriSpeech-100h and TED-LIUM2. It should be noted that we refer to (Higuchi et al., 2021a) for the prior results, and the comparison is not necessarily in an equivalent setting, e.g., we conducted experiments using ESPnet2 while the previous work used ESPnet1. Overall, BERT-CTC achieved better results than the other non-autoregressive models, thanks to the usage of BERT. In particular, we observed clear differences in the LibriSpeech "other" sets and TED-LIUM2. However, the performance on the LibriSpeech "clean" set was on par with the other approaches, which we attribute to the vocabulary mismatch problem we have discussed in the limitation section.