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

Yosuke Higuchi\textsuperscript{1,2}, Brian Yan\textsuperscript{1}, Siddhant Arora\textsuperscript{1}, Tetsuji Ogawa\textsuperscript{2},
Tetsunori Kobayashi\textsuperscript{2}, Shinji Watanabe\textsuperscript{1}

\textsuperscript{1}Carnegie Mellon University, \textsuperscript{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.
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 \( O = (o_t \in \mathbb{R}^D | t = 1, \cdots, T) \) 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, \( o_t \) 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 \( O: \)

\[
\hat{W} = \arg\max_{W \in \mathcal{V}^*} p(W|O),
\]

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 \( \epsilon \) 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:

\[
p_{\text{ctc}}(W|O) = \sum_{A \in \mathcal{B}_{\text{ctc}}(W)} p(W|A, \emptyset)p(A|O)
\]

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

where \( \mathcal{B}_{\text{ctc}}^{-1}(W) \) 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

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

\[
p(a_t|O) = \text{Softmax}(\text{Linear}(h_t^{ae})), \quad (5)
\]
\[
h_t^{ae} \sim \text{AudioEnc}(O). \quad (6)
\]

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

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

\[
L_{ctc}(O, W) = -\log \sum_{A \in B_{ctc}(W)} \prod_{t=1}^{T} p(a_t|O). \quad (7)
\]

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: \( \hat{a}_t = \arg\max_{a_t} p(a_t|O) \). The most probable token sequence \( \hat{W} \) is then obtained by applying the collapsing function to \( \hat{A} \) as \( \hat{W} = B_{ctc}(\hat{A}) \).

### 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 \( \langle w_1, \cdots, w_{n_u-1} \rangle \). Let \( Z = \langle z_u \in V \cup \{\epsilon\} | u = 1, \cdots, T + N \rangle \) be an alignment used in RNN-T, and RNN-T factorizes \( p(W|O) \) similarly to Eq. (3) as

\[
p_{rnnt}(W|O) \approx \sum_{Z \in B_{rnnt}(W)} p(Z|O), \quad (8)
\]

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

\[
p(Z|O) = \prod_{u=1}^{T+N} p(z_u|z_1, \cdots, z_{u-1}, O) \quad (9)
\]

\[
\approx \prod_{u=1}^{T+N} p(z_u|w_1, \cdots, w_{n_u-1}, O) \quad (10)
\]

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

\[
p(z_u|w_1, \cdots, w_{n_u-1}, O) = \text{Softmax}(\text{JointNet}(h_u^{ae}, h_{n_u}^{bn})), \quad (11)
\]

\[
h_{n_u}^{bn} = \text{PredictionNet}(w_1, \cdots, w_{n_u-1}). \quad (12)
\]

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

**Training**  The RNN-T loss \( L_{rnnt}(O, W) \) 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)).
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 $\tilde{W} = (\tilde{w}_n \in \mathcal{V} \cup \{\text{MASK}\}) | n = 1, \ldots, N)$, 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 $\tilde{W}$ to obtain a masked sequence. Considering all possible $\tilde{W}$, the conditional probability $p(W|O)$ is factorized as follows:

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

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

$$p(W|\tilde{W}, O) = \sum_{A \in \mathcal{B}^{ctc}(W)} p(W, A|\tilde{W}, O)$$

$$\approx \sum_{A \in \mathcal{B}^{ctc}(W)} p(A|W, \tilde{W}, O)p(W|\tilde{W}, O).$$

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., $\mathcal{A} \times \mathcal{B}^{ctc}_W$). 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

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

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 ($w_1, \ldots, w_N$). 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

$$p(a_t, BERT(\tilde{W}), O) \triangleq \sum_{A \in \mathcal{B}^{ctc}(W)} \prod_{t=1}^{T} p(a_t|BERT(\tilde{W}), O),$$

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

---

Note that BERT(·) can be any pre-trained MLM.
trained end-to-end. The conditional probability \( p(\alpha_t | \text{BERT}(\hat{W}), O) \) is computed as
\[
p(\alpha_t | \text{BERT}(\hat{W}), O) = \text{Softmax}(\text{SelfAttn}_n(\hat{H}_{ae}, H_{\text{bert}})), \quad (18)
\]
\[
H_{\text{bert}} = \text{BERT}(\hat{W}). \quad (19)
\]
In Eq. (18), SelfAttn\(_n(\cdot)\) indicates the \( t \)-th output of stacked Transformer self-attention layers (Vaswani et al., 2017), which consume the concatenated \( \hat{H}_{ae} \) (from Eq. (6)) and \( H_{\text{bert}} \).\(^4\) In Eq. (19), BERT(\( \cdot \)) embeds a masked sequence \( \hat{W} \) into a sequence of \( \hat{d}_{\text{bert}} \)-dimensional hidden vectors \( H_{\text{bert}} = (h^n_{\text{bert}} \in \mathbb{R}^{d_{\text{bert}}} | n = 1, \cdots, N) \).

### 3.1 Training

The BERT-CTC objective is defined by the negative log-likelihood of Eq. (13) expanded with Eq. (15):
\[
-\log \sum_{\hat{W}} \sum_A p(A | W, O)p(W | \hat{W})p(\hat{W}|O). \quad (20)
\]
To deal with the intractable marginalization over \( \hat{W} \) in Eq. (20), we rewrite it under expectation with respect to the sampling distribution \( A(W) \):
\[
\approx -\mathbb{E}_{\hat{W} \sim A(W)} \left[ \log \sum_A p(A | W, O)p(W | \hat{W})p(\hat{W}|O) \right],
\]
whose upper bound can be derived by using the Jensen’s inequality as
\[
\leq -\mathbb{E}_{\hat{W} \sim A(W)} \left[ \log \sum_A p(A | W, O)p(W | \hat{W})p(\hat{W}|O) \right]
\leq -\mathbb{E}_{\hat{W} \sim A(W)} \left[ \log \sum_A \prod_{t} p(\alpha_t | \text{BERT}(\hat{W}), O) \right], \quad (21)
\]
where \( \mathcal{L}_{bc} \) 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 \( \hat{W} \) from \( A(W) \) 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).

### 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
\[
(1 - \lambda_{\text{ctc}}) \mathcal{L}_{bc}(O, W) + \lambda_{\text{ctc}} \mathcal{L}_{\text{ctc}}(O, W'), \quad (22)
\]
where \( \lambda_{\text{ctc}} \) 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
\[
\hat{W} = \arg \max_W \sum_{\hat{W}} p(W | \hat{W}, O)p(\hat{W}|O). \quad (23)
\]
\[
\approx \arg \max_W p(W | \hat{W}, O), \quad (24)
\]
where \( \hat{W} = \arg \max_W p(W|O) \). \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 \( N \) needs to be given in advance to start decoding (Gu et al., 2018). We determine the target length based on the auxiliary sequence \( W' \) predicted from the audio encoder output \( H_{ae} \) as \( N \sim |\hat{W}'| \). Given the estimated length, we initialize an initial masked sequence \( W(\hat{k} = 1) \) by filling all \( N \) positions with the mask token [MASK]. By feeding \( H_{ae} \) and \( H_{\text{bert}} \) (\( \Rightarrow \text{BERT}(\hat{W}(\hat{k} = 1)) \) to the self-attention module, a hypothesized sequence \( \hat{W}(\hat{k} = 1) \) is obtained via CTC inference. Here, \( \hat{W}(\hat{k} = 1) \) is predicted only from speech without any observations from output tokens, as they are all masked.

\(^4\)We apply simple embedding layers to \( H_{ae} \) and \( H_{\text{bert}} \) so that the dimensions of hidden vectors match, but we omit it for simplicity. See Appendix D.2 for detailed implementation.
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 $\hat{W}(k+1)$. Here, $m(k)$ is a linear decay function $m(k) = \lfloor |\hat{W}(k)| \cdot \frac{k}{K} \rfloor$, similar to (Ghazvininejad et al., 2019).

Token Prediction Step (Eq. (24)) $H^{ae}$ and $H^{bert}$ ($= BERT(\hat{W}(k+1))$) 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

$$p(y|BERT(\hat{W}), O) = \text{Softmax}(\text{SelfAttn}_{T+1}(H^{ae}, H^{bert})), \quad (26)$$

where we assume $y \in \mathcal{Y}$ as an intent label in a set of intents $\mathcal{Y}$. Note that SelfAttn$_{T+1}()$ 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

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

where $\lambda_{\text{slu}}$ is a tunable parameter.

Inference The most probable label $\hat{y}$ can be estimated at any timing of BERT-CTC inference by $\hat{y} = \arg\max_y p(y|\hat{W}, O)$. 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), TIDIOUM2 (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 $L_{ctc}$ (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 $L_{rnnt}$ (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 BERT$_{BASE}$ model 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 $\lambda_{ctc}$ (in Eq. (22)) to 0.3 for all the ASR tasks and $\lambda_{slu}$ (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
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.

<table>
<thead>
<tr>
<th>Model</th>
<th>LibriSpeech-100h</th>
<th>TED-LIUM2</th>
<th>AISHELL-1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dev WER (↓)</td>
<td>Test WER (↓)</td>
<td>Dev WER (↓)</td>
</tr>
<tr>
<td></td>
<td>clean</td>
<td>other</td>
<td>clean</td>
</tr>
<tr>
<td>CTC†</td>
<td>11.2</td>
<td>21.4</td>
<td>11.4</td>
</tr>
<tr>
<td>RNN-T†</td>
<td>9.7</td>
<td>21.5</td>
<td>9.8</td>
</tr>
<tr>
<td><strong>BERT-CTC</strong></td>
<td><strong>7.0</strong></td>
<td><strong>16.3</strong></td>
<td><strong>7.2</strong></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Model</th>
<th>Dev WER (↓)</th>
<th>Test WER (↓)</th>
<th>Acc. (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESPnet-SLU (Arora et al., 2022)</td>
<td>–</td>
<td>–</td>
<td>86.3</td>
</tr>
<tr>
<td>ASR + BERT (Arora et al., 2022)</td>
<td>–</td>
<td>–</td>
<td>85.7</td>
</tr>
<tr>
<td><strong>BERT-CTC (K = 1)</strong></td>
<td>19.1</td>
<td>87.0</td>
<td></td>
</tr>
<tr>
<td><strong>BERT-CTC (K = 20)</strong></td>
<td><strong>18.2</strong></td>
<td><strong>87.8</strong></td>
<td></td>
</tr>
</tbody>
</table>

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 BERT(·) 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.
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 \(p(W|\tilde{W},\varnothing)\)

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

<table>
<thead>
<tr>
<th>$\mathcal{V}$</th>
<th>Model</th>
<th>Dev WER (↓)</th>
<th>Test WER (↓)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{V}^{asr}$</td>
<td>CTC†</td>
<td>6.9 / 20.1</td>
<td>7.0 / 20.2</td>
</tr>
<tr>
<td>$\mathcal{V}^{asr}$</td>
<td>RNN-T†</td>
<td>5.7 / 17.0</td>
<td>6.0 / 17.2</td>
</tr>
<tr>
<td>$\mathcal{V}^{bert}$</td>
<td>CTC†</td>
<td>11.2 / 21.4</td>
<td>11.4 / 22.0</td>
</tr>
<tr>
<td>$\mathcal{V}^{bert}$</td>
<td>RNN-T†</td>
<td>9.7 / 21.5</td>
<td>9.8 / 22.3</td>
</tr>
<tr>
<td>$\mathcal{V}^{bert}$</td>
<td>BERT-CTC</td>
<td>7.0 / 16.3</td>
<td>7.2 / 16.6</td>
</tr>
</tbody>
</table>

**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 $\mathcal{V}^{asr}$ is an ASR vocabulary with a vocabulary size of 300 constructed from LibriSpeech transcriptions, and $\mathcal{V}^{bert}$ is the BERT vocabulary with a vocabulary size of 30522. We observed that, by using $\mathcal{V}^{asr}$, the performance of CTC and RNN-T improved over the results using $\mathcal{V}^{bert}$ 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 $\mathcal{O}(N^3)$ 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 (JPMJAX2101) 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**


Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda 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 $k > 1$. 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 $k > 1$. 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 $k > 1$.


Algorithm 1 BERT-CTC Inference

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

1. $A' = \arg\max_{A'} p(A'|O)$  \Comment{Obtain the most probable alignment from the audio encoder}
2. $\hat{W}' = B_{ctc}(A')$  \Comment{Obtain the target length from the intermediate prediction}
3. $\tilde{N} \sim |\hat{W}'|$  \Comment{Initialize a masked sequence}
4. $W = (w_n = [\text{MASK}]| n = 1, \cdots, \tilde{N})$  \Comment{Initialize an index for token position}
5. for $k = 1, \ldots, K$ do
   6. \hspace{1em} \# Token prediction
   7. $H_{\text{bert}} = \text{BERT}(\hat{W})$  \Comment{Forward BERT}
   8. $p(A|\cdot) = \text{Softmax}(\text{SelfAttn}(H^{ae}, H_{\text{bert}}))$  \Comment{Forward self-attention module}
   9. $\hat{A} = \arg\max_{A} p(A|\cdot)$  \Comment{Obtain the most probable alignment}
   10. $W = B_{ctc}(\hat{A})$
11. \# Token-level probability calculation
12. $\hat{P} = (\hat{p}_n = 0|n = 1, \cdots, |\hat{W}|)$  \Comment{Initialize token-level probabilities}
13. $n = 1$
14. $a_0 = \epsilon$
15. for $t = 1, \ldots, T$ do
   16. \hspace{1em} if $\hat{a}_t = \epsilon$ then
   17. \hspace{2em} if $\hat{a}_{t-1} \neq \epsilon$ then
   18. \hspace{3em} $n = n + 1$
   19. \hspace{2em} end if
   20. \hspace{1em} else
   21. \hspace{2em} $\hat{p}_n = \max(p(a_t = w_n|\cdot), \hat{p}_n)$  \Comment{Keep the maximum probability for each token}
   22. \hspace{2em} end if
   23. end for
14. \# Token masking
25. $M = \lfloor |\hat{W}|-\frac{K-k}{K} \rfloor$  \Comment{Calculate the number of masked tokens}
26. $\hat{W} = \text{MaskLowestProb}(\hat{W}, \hat{P}, M)$  \Comment{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, 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 $H^{(e)} = (h_t^{(e)} \in \mathbb{R}^{d_{ae}}| t = 1, \cdots, T)$ be an intermediate output of the $e$-th layer of the audio encoder, which is computed as in Eq. (6) as

$$h_t^{(e)} \sim \text{AudioEnc}^{(e)}(O),$$  \hspace{1em} (28)

where 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

$$p^{(e)}(a_t|O) = \text{Softmax}(\text{Linear}(h_t^{(e)})).$$  \hspace{1em} (29)
Finally, an intermediate CTC loss $L_{ctc}^{(e)}$ is defined similarly to Eq. (7) as
\[
L_{ctc}^{(e)}(O, W) = -\log \sum_{A \in B} \prod_{t=1}^{T} p^{(e)}(a_t | O).
\] (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 $L_{ctc}$ is extended as
\[
(1 - \lambda_{ic})L_{ctc}(O, W) + \lambda_{ic}L_{ctc}^{(e=6)}(O, W'),
\] (31)
where $\lambda_{ic}$ is a tunable weight for the intermediate loss, and we equally weighted each loss (i.e., $\lambda_{ic} = 0.5$) 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 $L_{rnn}$ is extended as
\[
(1 - \lambda_{ctc})L_{rnn}(O, W) + \lambda_{ctc}\{(1 - \lambda_{ic})L_{ic}(O, W') + \lambda_{ic}L_{ctc}^{(e=6)}(O, W')\},
\] (32)
where $\lambda_{ctc}$ is a tunable weight for CTC losses, and we set $\lambda_{ctc} = 0.3$ 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 $L_{bc}$ is extended as
\[
(1 - \lambda_{ctc})L_{bc}(O, W) + \lambda_{ctc}\{(1 - \lambda_{ic})L_{ic}(O, W') + \lambda_{ic}L_{ctc}^{(e=6)}(O, W')\},
\] (33)
where $\lambda_{ctc}$ is a tunable weight for CTC losses, and we set $\lambda_{ctc} = 0.3$. 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\(^3\), 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 $d_{fl}$ 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 $d_{model} = 256$, $d_{fl} = 2048$, and $d_{head} = 4$. Before feeding into the self-attention module, the hidden vectors, $H_{ae}$ and $H_{bert}$, were embedded 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 BERT\(_{BASE}\) 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.

---

\(^3\)https://github.com/espnet/espnet/tree/master/egs2/librispeech_100/asr
\(^6\)https://github.com/espnet/espnet/tree/master/egs2/tedlium2/asr
\(^7\)https://github.com/espnet/espnet/tree/master/egs2/aishell1/asr
\(^8\)https://github.com/espnet/espnet/tree/master/egs2/slurp/asr
\(^9\)https://github.com/huggingface/transformers
\(^10\)https://huggingface.co/bert-base-uncased
\(^11\)https://huggingface.co/bert-base-chinese
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Language</th>
<th>Speech Style</th>
<th># Train Hours</th>
<th># Valid. Hours</th>
<th># Test Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>LibriSpeech-100h</td>
<td>English</td>
<td>Read</td>
<td>100h</td>
<td>5.4h / 5.3h</td>
<td>5.4h / 5.1h</td>
</tr>
<tr>
<td>TED-LIUM2 (Rousseau et al., 2014)</td>
<td>English</td>
<td>Spontaneous</td>
<td>210h</td>
<td>1.6h</td>
<td>2.6h</td>
</tr>
<tr>
<td>AISHELL-1 (Bu et al., 2017)</td>
<td>Mandarin</td>
<td>Read</td>
<td>170h</td>
<td>10h</td>
<td>5h</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Language</th>
<th># Intents</th>
<th># Train Hours</th>
<th># Valid. Hours</th>
<th># Test Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLURP (Bastianelli et al., 2020)</td>
<td>English</td>
<td>69</td>
<td>40h + 43h</td>
<td>6.9h</td>
<td>10.3h</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dropout rate</td>
<td>0.1</td>
</tr>
<tr>
<td>LR schedule</td>
<td>Noam (Vaswani et al., 2017)</td>
</tr>
<tr>
<td>Max learning rate</td>
<td>best of [1e-3, 2e-3]</td>
</tr>
<tr>
<td>Warmup steps</td>
<td>15k</td>
</tr>
<tr>
<td>Epochs</td>
<td>best of [50, 70, 100]</td>
</tr>
<tr>
<td>Adam betas</td>
<td>(0.9, 0.98)</td>
</tr>
<tr>
<td>Weight decay</td>
<td>1e-6</td>
</tr>
</tbody>
</table>

Table 8: Training configuration for CTC model.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dropout rate</td>
<td>0.1</td>
</tr>
<tr>
<td>LR schedule</td>
<td>Noam (Vaswani et al., 2017)</td>
</tr>
<tr>
<td>Max learning rate</td>
<td>best of [1e-3, 2e-3]</td>
</tr>
<tr>
<td>Warmup steps</td>
<td>15k</td>
</tr>
<tr>
<td>Epochs</td>
<td>best of [50, 70, 100]</td>
</tr>
<tr>
<td>Adam betas</td>
<td>(0.9, 0.98)</td>
</tr>
<tr>
<td>Weight decay</td>
<td>1e-6</td>
</tr>
</tbody>
</table>

Table 9: Training configuration for RNN-T model.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dropout rate</td>
<td>0.1</td>
</tr>
<tr>
<td>LR schedule</td>
<td>Noam (Vaswani et al., 2017)</td>
</tr>
<tr>
<td>Max learning rate</td>
<td>2e-3</td>
</tr>
<tr>
<td>Warmup steps</td>
<td>15k</td>
</tr>
<tr>
<td>Epochs</td>
<td>best of [50, 70]</td>
</tr>
<tr>
<td>Adam betas</td>
<td>(0.9, 0.98)</td>
</tr>
<tr>
<td>Weight decay</td>
<td>1e-6</td>
</tr>
</tbody>
</table>

Table 10: Training configuration for BERT-CTC model.

## D.3 Tokenization

We used the same subword vocabulary as BERT for tokenizing target texts, where the vocabulary size $|V|$ was 30522 for English and 21128 for Mandarin. For the smaller-sized vocabulary $V'$ 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 TEDLIUM2 and SLURP. For AISHELL-1, we used character-level tokenization with 4231 Chinese characters.

\(^{12}\)https://github.com/google/sentencepiece

## 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 0.05$T$. 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.

\(^{13}\)https://github.com/espnet/espnet
\(^{14}\)https://github.com/pytorch/pytorch
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.

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

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).

| Model                                                | #iters | LibriSpeech-100h | TED-LIUM2 | |
|------------------------------------------------------|--------|------------------|-----------|
|                                                      |        | Dev WER (↓)      | Test WER (↓) |
|                                                      |        | clean | other | clean | other | clean | other | clean | other | |
| 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)                         | ≃ log₂(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   |

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.