# CTCBERT: ADVANCING HIDDEN-UNIT BERT WITH CTC OBJECTIVES

*Ruchao Fan*[1]*, *Yiming Wang*[2], *Yashesh Gaur*[2], *Jinyu Li*[2]

[1]University of California, Los Angeles [2]Microsoft Corporation

## ABSTRACT

In this work, we present a simple but effective method, CTCBERT, for advancing hidden-unit BERT (HuBERT). HuBERT applies a frame-level cross-entropy (CE) loss, which is similar to most acoustic model training. However, CTCBERT performs the model training with the Connectionist Temporal Classification (CTC) objective after removing duplicated IDs in each masked region. The idea stems from the observation that there can be significant errors in alignments when using clustered or aligned IDs. CTC learns alignments implicitly, indicating that learning with CTC can be more flexible when misalignment exists. We examine CTCBERT on IDs from HuBERT Iter1, HuBERT Iter2, and PBERT. The CTC training brings consistent improvements compared to the CE training. Furthermore, when loading blank-related parameters during finetuning, slight improvements are observed. Evaluated on the Librispeech 960-100h setting, the relative WER improvements of CTCBERT are 2%-11% over HuBERT and PERT on test-other data.

***Index Terms***— Self-supervised learning, HuBERT, CTC, CE

## 1. INTRODUCTION

Recently, self-supervised learning (SSL) has gained lots of popularity because of its superiority of learning from large amounts of un-annotated data [1–7]. This learning process is considered as *pretraining*. Later on, fine-tuning a system using the pretrained model as a starting point often obtains impressive results [8–10]. Such a good property motivates researchers to design SSL algorithms. The key of SSL is to construct a valuable task with un-annotated data in pretraining. Current SSL methods in speech can be categorized into either generative or discriminative approaches according to their loss functions.

Generative approaches predict future frames given history context similar to GPT [11], or reconstruct masked features given unmasked ones similar to BERT [12]. Representative models in generative approaches include APC, VQ-APC [13–15], DeCoAR [16], Mockingjay [17], TERA [18], and MPC [19]. These approaches, however, perform predictions on a continuous space which may not be compatible with the fine-tuning tasks. On the other hand, discriminative approaches employ discriminative loss functions such that the model can be aware of the classification concepts, which may be beneficial for fine-tuning. Representative models in this category are Wav2vec [20], Wav2vec2.0 [21], HuBERT [22] and PBERT [23]. Wav2vec series discriminate negative samples from positive ones using a contrastive loss to push the prediction close to the positive sample and away from the negative samples. However, sampling negative samples is not always a well-defined problem because the definition of "negative" is sometimes vague and can vary depending on specific tasks. HuBERT [22] uses an acoustic unit discovery system (K-means) to create pseudo-labels for each frame to avoid the process of sampling negative samples. However, HuBERT training needs multiple iterations to obtain a good performance. Instead of using k-means, PBERT [23] uses a model trained with the finetuning data to obtain the IDs by conventional automatic speech recognition (ASR) decoding. The quality of these IDs are much better than that of clustered IDs in HuBERT. Hence, one iteration for PBERT can already obtain better performance than HuBERT.

In this paper, we propose a simple but effective method, CTCBERT, for training ID-based SSL systems including HuBERT and PBERT. Conventionally, HuBERT and PBERT use cross entropy (CE) loss in the training, which is similar to most acoustic model training. However, CTCBERT performs the model training with the Connectionist Temporal Classification (CTC) objective [24] after removing duplicated IDs in each masked region. The idea is based on our observation that there are significant errors of alignments in learning IDs. CTC computes the summation of all possible alignments to labels and can learn the alignment implicitly. Hence, learning with the CTC objective can be more flexible when the misalignment exists. We examine CTCBERT on IDs from HuBERT Iter1, HuBERT Iter2, and PBERT. The CTC training has consistent improvements compared to the CE training, indicating that CTCBERT achieves better performance than HuBERT and PBERT, respectively. Another advantage of CTC training is that CTCBERT contains blank-related parameters that can potentially be used for finetuining. Slight improvements are observed when loading blank-related parameters from the pretrained model during finetuning.

The remainder of this paper is organized as follows. Section 2 introduces the proposed CTCBERT method. Experimental setups are described in Section 3. Results are shown and discussed in Section 4. We conclude the paper in Section 5.

## 2. CTCBERT

In this section, we introduce the proposed CTCBERT, from our motivations to the technical details.

### 2.1. Misalignment in Learning IDs

The core of CTCBERT is to use the CTC objective for ID-based SSL pretraining. Hence, we first introduce our motivation to use the CTC objective. HuBERT and PBERT can achieve good performance for self-supervised learning. Are they reaching the top performance? How can we further improve the performance of self-supervised learning? The simplest solution may be increasing the quality of the learning IDs by more iterations. However, multiple pre-training iterations would incur more computational cost and is not environmentally friendly.

Recently, we analyze the obtained IDs from PBERT and compare them with the ground-truth IDs. The ground-truth IDs are acquired by training a hybrid model with the entire train-960 data first.

*Work done during an internship at Microsoft.

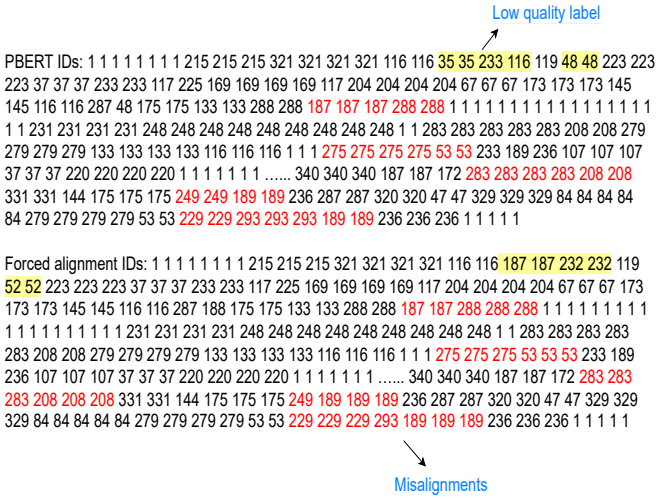


**Fig. 1**. PBERT and forced alignment IDs. Forced aligned IDs are obtained from a hybrid model trained with train-960 data, and can be regarded as the ground-truth of the PBERT IDs. Yellow indicates the low quality labels. Red represents the misalignment.

Then we use the ground-truth to perform the forced-alignment, and obtain the forced-aligned IDs, which, therefore, can be regarded as the ground-truth PBERT IDs. The comparisons of the two types of IDs are shown in Fig.1. As we can observe from the figure, there are two main errors in PBERT IDs. One is the low quality label in yellow. The other is the misalignment in red. The quality of label is hard to improve. However, it may be feasible to mitigate the misalignment problem. By carefully observing each misalignment in red in Figure 1, we hypothesize that CTC is a more appropriate loss function than CE loss for the misalignment problem. For example, the PBERT IDs are $\{187, 187, 187, 288, 288\}$ and the ground-truth IDs are $\{187, 187, 288, 288, 288\}$. If CTC loss is used, the targets are both $\{187, 288\}$. The alignment can be learned implicitly in the model with CTC loss. The model will be more flexible and tolerant to the misalignment, which is our motivation to train the ID-based SSL systems with the CTC objective.

### 2.2. HuBERT

HuBERT leverages K-means to create pseudo-labels for each frame, and then uses the pseudo-labels to pretrain the model. HuBERT consists of a feature extractor, a transformer encoder, a final projection layer and clustered ID embeddings. Mathematically, suppose the input raw waveform is $X = \{x_1, x_2, ..., x_n\}$, then after the feature extractor layers, we obtain the feature sequence $Z = \{z_1, z_2, ..., z_T\}$. HuBERT masks $Z$ sequence to predict masked regions given unmasked regions. Let masked regions be $Z_{mask}$. $Z$ with masked embeddings is fed to the transformer encoder $H$, and then the projection layer $P$. After computing the similarity between the projection layer outputs and the clustered ID embeddings $\{E_0, E_1, ..., E_{N-1}\}$, we finally obtain the softmax output for each frame, denoted as: $y = \{y_1, y_2, ...., y_T\}$.

Suppose the pseudo-label IDs are $C = \{c_1, c_2, ..., c_T\}$, then HuBERT loss can be described as:

$$L_{\text{HuBERT}} = -\sum_{t \in Z_{mask}} logP(c_t|Z) \tag{1}$$

HuBERT is essentially using cross-entropy (CE) loss to maximize the logits for each frame and improves the correctness of the predictions. As a result, the low quality labels would have a negative effect for the HuBERT pretraining.

### 2.3. PBERT

Instead of using the acoustic unit discovery system to create pseudo-labels for each utterance, PBERT uses a small amount of paired data (finetuning data) to train a hybrid model first, and then apply decoding on the training data to get the aligned IDs. The quality of the aligned IDs, from the perspective of speech recognition, is better than the clustered IDs. Hence, the performance of PERT is better than that of HuBERT, even with only one iteration. The pretraining model structure of PBERT is the same as HuBERT. PBERT uses CE loss as well to increase the prediction accuracy of each frame.

If using the CE loss in HuBERT and PBERT, the misalignment errors can be regarded as the low quality label. And CE loss will try to increase the accuracy with the low quality label, which hurts the model training. As mentioned in Section 2.1, using CTC loss may solve the problem of the misalignment.

### 2.4. Connectionist Temporal Classification

Connectionist Temporal Classification (CTC) is a well-known sequence-level loss for speech recognition. CTC adds the "blank" token to solve the length mismatch problem between the input sequence and the label sequence. Suppose the input sequence is $X' = \{x'_1, x'_2, ...., x'_T\}$ and the label sequence is $Y = \{Y_1, Y_2, ...., Y_u\}$. Define a mapping rule $\beta$ that removes blank and repetitive tokens of the input sequence. Then, the CTC can be described as a summation of all the paths that can be mapped by rule $\beta$ to the label $Y$. We can write the CTC loss function as:

$$L_{\text{CTC}} = -\sum_{a \in \beta^{-1}(Y)} \prod_{t=1}^{T} P(a_t|X') \tag{2}$$

where $\beta^{-1}$ is the inverse mapping function. CTC learns the alignment between the input and output sequence. It does not affected by the misalignment. We use another example in Figure 1: the aligned IDs $\{229, 229, 293, 293, 293, 189, 189\}$ and the ground-truth IDs $\{229, 229, 229, 293, 189, 189, 189\}$ are basically two paths of the label sequence $\{229, 293, 189\}$ and are both considered in the CTC loss. Hence, CTC loss is tolerant and not sensitive to the misalignment. Consequently, we assume that using CTC loss can well solve the problem of misalignment.

### 2.5. CTCBERT

Using the example in the section 2.4, we can easily find that if CTC loss is used, the targets are mapped to $\{229, 293, 189\}$ for both ID sequences. CTC can take care of the alignment in the sequence during training, indicating that training with CTC is more flexible and tolerant to the misalignment. Compared to CE training, which is frame-based and is very sensitive to label assumptions for each frame, CTC training is utterance/segment-based and is not sensitive to the frame alignment. Suppose the mask regions are $Z_{mask} = \{Z_1, Z_2, ..., Z_M\}$ and the corresponding masked IDs are $C = \{C_1, C_2, ...., C_M\}$. We define a rule $A$, which is to remove the repetitive IDs in each masked region. The loss function of CTCBERT can be described as:

$$L_{\text{CTCBERT}} = -\sum_{m=1}^{M} \sum_{a \in \beta^{-1}(A(C_m))} \prod_{t \in Z_m} P(a_t|Z) \tag{3}$$

where $\beta^{-1}$ is the inverse function of $\beta$ that maps all the paths to $A(C_m)$. However, in our preliminary experiments, we find that CTC training only improves on HuBERT Iter2 IDs. In other cases, we observe slow convergence of CTC training. Hence, we borrow canonical ideas from acoustic model training [25]: CTC and CE joint training or using CE warmup. For CTC and CE joint training, we set the task ratio of CTC loss as $\alpha$, the loss function can be rewritten as:

$$L_{\text{joint}} = \alpha L_{\text{CTCBERT}} + (1-\alpha) L_{\text{HuBERT}} \tag{4}$$

For CE warmup, it represents that the model is trained with CE loss for certain steps at the beginning, and then the model aims for CTC training. Both solutions can alleviate the slow convergence phenomena of CTC training. Another advantage of training ID-based SSL systems with the CTC loss may be the consistency of the pretraining and finetuning task (when using the CTC objective).

### 2.6. Blank-Related Parameters

Computing the CTC loss involves an additional blank token, which is the same as the finetuning task. Therefore, we are thinking whether the blank-related parameters used in the pretraining can be used for the finetuning task. When we use the CTC loss, the ID embeddings are $\{E_b, E_0, E_1, ...., E_{N-1}\}$, where $E_b$ is the embebdding for the blank token. If the weight of the projection layer is $W_p \in \mathbb{R}^{d_{embed} \times d_{model}}$ and its bias is $b_p \in \mathbb{R}^{d_{embed}}$, we can extract the blank-related parameters as follows:

$$\begin{aligned} W_b &= W_p^T * E_b \in \mathbb{R}^{d_{model}} \\ b_b &= b_p * E_{blank}^T \in \mathbb{R} \end{aligned} \tag{5}$$

$W_b$ and $b_b$ are then used for initializing the corresponding blank-related parameters in the last projection layer during finetuning. Note that this is effective when CTC loss is used during finetuning.

## 3. EXPERIMENTAL SETUP

### 3.1. Dataset

In this paper, we evaluate our method on the LibriSpeech 960h-100h benchmark [26]. Librispeech contains 960 hours of training data from read audiobooks and is being split into three subsets: train-clean-100, train-clean-360 and train-other-500. We use the entire 960h data as the pretraining data (without transcripts), and use the train-clean-100 subset as the finetuning data (with transcripts).

### 3.2. Pretraining

We conduct experiments on IDs from HuBERT Iter1, HuBERT Iter2, and PBERT with the same model architecture (HuBERT-Base) in [22]. For HuBERT Iter1 and HuBERT Iter2, their clustered IDs are obtained from K-means of MFCC and HuBERT intermediate layer features, respectively. For PBERT, the train-clean-100 portion is used to train a hybrid model. The acoustic model is a TDNN-F [27] trained with LF-MMI criterion [28]. We decode the entire train-960 data using the 3-gram LM in the Librispeech data, and then rescore the lattices using the 4-gram LM, which is also in the Librispeech data. The best path is converted to the ID sequence as the PBERT IDs. We use 347 distinct positional-dependent phonemes as the training targets. All models are trained on 32 GPUs, with a batch size of at most 87.5 seconds of audio per GPU for 400k steps. The mask strategies are the same for all model training including HuBERT, PBERT and CTCBERT. Specifically, the mask span is set to $l = 10$, and $p = 8\%$ of the waveform encoder output frames are randomly selected as mask start. We use AdamW optimizer [29] with weight decay 0.01 and $\beta = (0.9, 0.98)$. The learning rate ramps up linearly for the first 32k steps and then decays linearly back to 0. The peak learning rates is 5e-4. We replicate the results in the literature.

**Table 1**. The performance of CTCBERT using HuBERT iter1, HuBERT iter2, and PBERT IDs. "load blk." indicates loading blank-related parameters during finetuning.

| Model | LM | dev | | test | |
|---|---|---|---|---|---|
| | | clean | other | clean | other |
| Supervised baseline | None | - | - | 8.8 | 26.5 |
| | 4-gram | - | - | 5.0 | 16.8 |
| HuBERT iter1 [22] | None | 7.1 | 16.7 | 7.2 | 16.5 |
| | 4-gram | 3.2 | 9.6 | 3.8 | 9.5 |
| → CTCBERT | None | 6.9 | 15.9 | 7.0 | 15.8 |
| | 4-gram | 3.2 | 9.2 | 3.8 | 9.4 |
| + load blk. | None | 6.8 | 15.8 | 6.9 | 15.7 |
| | 4-gram | 3.2 | 9.2 | 3.7 | 9.3 |
| HuBERT iter2 [22] | None | 5.3 | 13.7 | 5.5 | 13.6 |
| | 4-gram | 2.8 | 8.7 | 3.4 | 8.7 |
| → CTCBERT | None | 5.2 | 12.3 | 5.4 | 12.1 |
| | 4-gram | 2.7 | 7.8 | 3.3 | 7.9 |
| + load blk. | None | 5.2 | 12.4 | 5.4 | 12.1 |
| | 4-gram | 2.7 | 7.9 | 3.3 | 7.9 |
| PBERT [23] | None | 4.6 | 11.7 | 4.8 | 11.8 |
| | 4-gram | 2.6 | 7.3 | 3.2 | 7.7 |
| → CTCBERT | None | 4.7 | 11.5 | 4.8 | 11.4 |
| | 4-gram | 2.5 | 7.1 | 3.2 | 7.5 |
| + load blk. | None | 4.6 | 11.3 | 4.8 | 11.3 |
| | 4-gram | 2.5 | 7.1 | 3.1 | 7.4 |

### 3.3. Finetuning

The model is fine-tuned on 8 GPUs with a batch size of 200 seconds of audio per GPU for 80k steps. The convolutional feature extractor is always fixed. Adam optimizer is used for all model training. A tri-stage scheduler of the learning rate is used, where the first 10% of updates are for warm up, the learning rate held constant for the next 40% and then linearly decayed for the rest of updates. The evaluation is based on the wav2letter++ [30] beam search decoder wrapped in Fairseq [31]. For language model-fused decoding, a beam size 1500 and a 4-gram LM are used.

## 4. RESULTS AND DISCUSSION

### 4.1. Main Results

In this section, we present the performance of CTCBERT on three different IDs: HuBERT iter1, HuBERT iter2, and PBERT. The results are shown in Table 1. For HuBERT iter1 IDs, CTCBERT improves the WER from 16.5% to 15.8% on the test other data when no language model is used. The improvement becomes smaller ($9.5\% \rightarrow 9.4\%$) when the 4-gram language model is used during decoding. Using HuBERT iter2 IDs, CTCBERT achieves the best improvements. CTCBERT obtains 11%, and 9% relative WER improvements compared to CE training when no language model and 4-gram language model is used, respectively. For PBERT, CTCBERT also achieves improvements from 7.7% to 7.5% on test-other data. Overall, the improvements on noisy data are larger

**Table 2**. The effect of CE warmup and CTC, CE joint training.

| Model | ce | | dev | | test | |
|---|---|---|---|---|---|---|
| | ratio | warmup | clean | other | clean | other |
| CTCBERT with iter1 IDs | 0.0 | 0 | 7.2 | 16.8 | 7.5 | 16.7 |
| | 0.0 | 32k | 7.2 | 15.8 | 7.3 | 15.8 |
| | 0.2 | 0 | 7.2 | 16.6 | 7.3 | 16.7 |
| | 0.5 | 0 | 6.9 | 15.9 | 7.0 | 15.8 |
| | 0.8 | 0 | 6.8 | 16.0 | 7.0 | 16.1 |
| | 1.0 | 0 | 7.1 | 16.7 | 7.2 | 16.5 |
| CTCBERT with iter2 IDs | 0.0 | 0 | 5.2 | 12.3 | 5.4 | 12.1 |
| | 0.0 | 32k | 5.2 | 12.6 | 5.4 | 12.4 |
| | 0.5 | 0 | 5.5 | 13.3 | 5.6 | 13.1 |
| | 1.0 | 0 | 5.3 | 13.7 | 5.5 | 13.6 |
| CTCBERT with PBERT IDs | 0.0 | 0 | 5.5 | 13.8 | 5.7 | 13.9 |
| | 0.0 | 150k | 4.7 | 11.6 | 4.9 | 11.7 |
| | 0.5 | 0 | 4.7 | 11.5 | 4.8 | 11.4 |
| | 1.0 | 0 | 4.6 | 11.7 | 4.8 | 11.8 |

than that on clean data. The reason may be because noisy speech are more likely to generate IDs with misalignments. CTCBERT achieves better performance than HuBERT or PBERT. Note that we replicated the results in the literature, and may cause difference in WER. However, the comparisons are reasonable because the same target IDs are used for CE and CTC training.

We can observe from Table1 that when using blank-related parameters during finetuning, most of the models get further improvements. For example, for HuBERT iter1, the WER drops from 9.4% to 9.3%, and for PBERT, the WER drops from 7.5% to 7.4%, all on the test other data. We do not see an improvement for HuBERT iter2. The reason may be that the performance of CTCBERT has already been very good. However, we do not see loading blank-related parameters in finetuning hurt the performance a lot. Loading blank-related parameters overall brings positive effect.

### 4.2. Stabilize the CTC training

As we mentioned in Section 2.5, CTC sometimes shows slow convergence, and needs strategies like CTC and CE joint training, or CE warmup to stabilize the training. For different IDs: HuBERT iter1, HuBERT iter2, PBERT, their best performance achieves differently. The results of the effect of CE warmup and CTC, CE joint training are shown in Table 2. We can observe from the table that for CTCBERT with HuBERT iter1 IDs, CE warmup training can improve the performance. However, the best performance achieves at $0.5\text{CE} + 0.5\text{CTC}$. For CTCBERT with HuBERT iter2 IDs, the pure CTC training achieves the best performance. For CTCBERT with PERBT IDs, the best performance also achieves at $0.5\text{CE}+0.5\text{CTC}$. In fact, $0.5\text{CE} + 0.5\text{CTC}$ also achieves improvements on HuBERT iter2 IDs compared to CE training.

Hence, how to choose the training strategie and the ratio in CE and CTC joint training? From our experience, CTC and CE joint training is always better than CE warmup training. For CTC and CE joint training, starting CTCBERT with the ratio of $0.5\text{CE} + 0.5\text{CTC}$ would be a good choice.

### 4.3. Quantitative Analysis

We want to perform a quantitative analysis to again confirm that CTC is more tolerate to misalignment of clustered and force aligned IDs. To do so, we first train a hybrid model with the train-clean-100h data. Then, using ground-truth and the hybrid model, we obtain the forced aligned IDs for train-960h and train-clean-100h data. Hence, the quality of alignment on train-clean-100h data is better than that

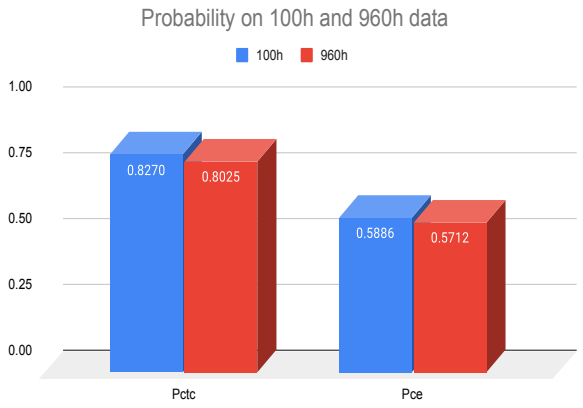


**Fig. 2**. Posterior probabilities of CTC and CE models calculated on train-clean-100h and train-960h data. IDs are obtained from force alignment using ground-truth and a hybrid model trained with train-clean-100 data. Hence, the IDs quality of train-clean-100h data is better than that of train-960 data.

of train-960h data. If CTC is more tolerate to misalignment, the model trained with the CTC loss should provide a smaller degradation of the posterior probability for data (an average probability of all ground-truth tokens) from train-clean-100h to train-960h than the model trained with CE loss. The results are shown in Fig.2. When we compute the relative posterior probability degradation, we get $(0.8270 - 0.8025)/0.8270 = 2.9\%$ for the CTC model, and $(0.5886 - 0.5712)/0.5886 = 3.1\%$ for the CE model. The CTC model has a smaller degradation of posterior probability from train-clean-100 to train-960 than the CE model. It confirms our conclusion that CTC is more tolerant to the misalignment. Note that the relative improvements of probability from CE to CTC ($3.1\% \rightarrow 2.9\%$) are similar to the relative WER improvements ($7.7\% \rightarrow 7.4\%$) when training PBERT with the CTC objective.

## 5. CONCLUSION

In this paper, we present a simple but effective method, CTCBERT, for advancing hidden-unit BERT (HuBERT). HuBERT uses a frame-level cross-entropy loss in the masked regions, which is similar to most acoustic model training. In contrast, the proposed CTCBERT performs model training with CTC objectives after removing duplicated IDs in each masked region. The idea stems from our observation that there can be significant errors in alignments when using clustered or aligned IDs. Through our quantitative analysis, we confirm that CTC is more tolerant to misalignment of IDs. This implies learning with the CTC objective can be more flexible when misalignment exists. CTCBERT is examined on the three different IDs: HuBERT Iter1, HuBERT Iter2, and PBERT. From our experiments, CTCBERT achieves consistent improvements on the three IDs compared to the CE training. Furthermore, slight improvements are observed, when loading blank related parameters during the finetuning. Under the setting of Librispeech 960h-100h, CTCBERT on HuBERT iter 2 achieves 11% relative WER improvements compared to CE training. For other IDs, CTCBERT can have a consistent relative WER improvements of around 4% compared to CE training. Empirically, experimenting CTCBERT with $0.5\text{CE}+0.5\text{CTC}$ would be a good starting point for stabilizing the training.

## 6. REFERENCES


[1] Junyi Ao and Ziqiang Zhang others, "Pre-Training Transformer Decoder for End-to-End ASR Model with Unpaired Speech Data," in *Interspeech 2022*, 2022, pp. 2658–2662.

[2] Shuo Ren, Shujie Liu, Yu Wu, Long Zhou, and Furu Wei, "Speech Pre-training with Acoustic Piece," in *Interspeech 2022*, 2022, pp. 2648–2652.

[3] A Arunkumar and Srinivasan Umesh, "Joint Encoder-Decoder Self-Supervised Pre-training for ASR," in *Interspeech 2022*, 2022, pp. 3418–3422.

[4] Chengyi Wang, Yu Wu, et al., "Improving self-supervised learning for speech recognition with intermediate layer supervision," in *ICASSP 2022*, 2022, pp. 7092–7096.

[5] Hung-yi Lee and Abdelrahman others Mohamed, "Self-supervised representation learning for speech processing," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics*, 2022, pp. 8–13.

[6] Abdelrahman Mohamed, Hung-yi Lee, et al., "Self-supervised speech representation learning: A review," *arXiv preprint arXiv:2205.10643*, 2022.

[7] Anuroop Sriram, Michael Auli, and Alexei Baevski, "Wav2Vec-Aug: Improved self-supervised training with limited data," in *Interspeech 2022*, 2022, pp. 4950–4954.

[8] Yu Zhang, Daniel S Park, Wei Han, et al., "Bigssl: Exploring the frontier of large-scale semi-supervised learning for automatic speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, 2022.

[9] Sanyuan Chen, Chengyi Wang, Zhengyang Chen, et al., "Wavlm: Large-scale self-supervised pre-training for full stack speech processing," *IEEE Journal of Selected Topics in Signal Processing*, 2022.

[10] Ruchao Fan, Yunzheng Zhu, Jinhan Wang, and Abeer Alwan, "Towards better domain adaptation for self-supervised models: A case study of child asr," *IEEE Journal of Selected Topics in Signal Processing*, 2022.

[11] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al., "Improving language understanding by generative pre-training," *OpenAI*, 2018.

[12] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

[13] Yu-An Chung, Wei-Ning Hsu, Hao Tang, and James R. Glass, "An unsupervised autoregressive model for speech representation learning," in *Interspeech 2019*, 2019, pp. 146–150.

[14] Yu-An Chung and James R. Glass, "Generative pre-training for speech with autoregressive predictive coding," in *ICASSP 2020*, 2020, pp. 3497–3501.

[15] Yu-An Chung, Hao Tang, and James R. Glass, "Vector-quantized autoregressive predictive coding," in *Interspeech 2020*, 2020, pp. 3760–3764.

[16] Shaoshi Ling, Yuzong Liu, Julian Salazar, and Katrin Kirchhoff, "Deep contextualized acoustic representations for semi-supervised speech recognition," in *ICASSP 2020*, 2020, pp. 6429–6433.

[17] Andy T. Liu, Shu-Wen Yang, Po-Han Chi, et al., "Mockingjay: Unsupervised speech representation learning with deep bidirectional transformer encoders," in *ICASSP 2020*, 2020, pp. 6419–6423.

[18] Andy T. Liu, Shang-Wen Li, and Hung-yi Lee, "TERA: self-supervised learning of transformer encoder representation for speech," *IEEE ACM Trans. Audio Speech Lang. Process.*, vol. 29, pp. 2351–2366, 2021.

[19] Dongwei Jiang, Wubo Li, Ruixiong Zhang, et al., "A further study of unsupervised pretraining for transformer based speech recognition," in *ICASSP 2021*, 2021, pp. 6538–6542.

[20] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli, "wav2vec: Unsupervised pre-training for speech recognition," in *Interspeech 2019*, 2019, pp. 3465–3469.

[21] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *NeurIPS 2020*, 2020.

[22] Wei-Ning Hsu, Benjamin Bolte, et al., "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE ACM Trans. Audio Speech Lang. Process.*, vol. 29, pp. 3451–3460, 2021.

[23] Chengyi Wang, Yiming Wang, Yu Wu, Sanyuan Chen, Jinyu Li, Shujie Liu, and Furu Wei, "Supervision-guided codebooks for masked prediction in speech pre-training," in *Interspeech 2020*, 2020.

[24] Alex Graves, Santiago Fernández, et al., "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *ICML 2006*, 2006, vol. 148, pp. 369–376.

[25] Shiliang Zhang and Ming Lei, "Acoustic modeling with DFSMN-CTC and joint CTC-CE learning," in *Interspeech 2018*, B. Yegnanarayana, Ed., 2018, pp. 771–775.

[26] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *ICASSP 2015*, 2015, pp. 5206–5210.

[27] Daniel Povey, Gaofeng Cheng, Yiming Wang, Ke Li, Hainan Xu, Mahsa Yarmohammadi, and Sanjeev Khudanpur, "Semi-orthogonal low-rank matrix factorization for deep neural networks," in *Interspeech 2018*, 2018, pp. 3743–3747.

[28] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur, "Purely sequence-trained neural networks for ASR based on lattice-free MMI," in *Interspeech 2016*, 2016, pp. 2751–2755.

[29] Ilya Loshchilov and Frank Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2018.

[30] Vineel Pratap, Awni Hannun, et al., "Wav2letter++: A fast open-source speech recognition system," in *ICASSP 2019*. IEEE, 2019, pp. 6460–6464.

[31] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli, "fairseq: A fast, extensible toolkit for sequence modeling," in *NAACL 2019*, 2019, pp. 48–53.