AdaBERT-CTC: Leveraging BERT-CTC for Text-Only Domain Adaptation in ASR

Tyler Vuong[*] **Karel Mundnich** **Dhanush Bekal** **Veera Raghavendra Elluru**
Srikanth Ronanki **Sravan Bodapati**
AWS AI Labs
{tylvuong, kmundnic, dkannang, veerare, ronanks, sravanb}@amazon.com

Abstract

End-to-end (E2E) automatic speech recognition (ASR) models are becoming increasingly popular in commercial applications, such as virtual assistants, closed captioning, and dictation systems. The accuracy of the ASR is crucial to their success. However, E2E models still struggle to recognize out-of-domain words such as proper nouns and domain-specific terms. In this paper we introduce AdaBERT-CTC, a domain adaptation technique that relies solely on textual data. Our method allows for text-only adaptation by fine-tuning a pre-trained self-supervised text encoder model. Additionally, we show that our method can be made parameter-efficient by adding bottleneck adapters to the pre-trained model. This allows for adaptation with less than a 5% increase in parameters and minimal computational overhead during inference. We demonstrate that our approach outperforms the base BERT-CTC model by up to 14% relative word error rate improvement on several out-of-domain, publicly available datasets.

1 Introduction

End-to-end (E2E) automatic speech recognition (ASR) models such as Connectionist Temporal Classification (CTC) (Graves et al., 2013; Vaswani et al., 2017) have become popular due to their ability to map acoustic features to text sequences using a single model. These architectures do not require an acoustic model (AM), language models (LM), nor explicit alignment information during training. These characteristics make them an attractive choice for large-scale production settings (He et al., 2019; Zhang et al., 2020). However, performance deteriorates on out-of-domain datasets not seen during training (Sainath et al., 2018), and adapting these models to out-of-domain data is a difficult task due to the lack of separate acoustic (AM) and language models (LM) (Shenoy et al., 2021), computational costs, catastrophic forgetting (Kirkpatrick et al., 2017), and an often lack of large amounts of labeled, domain-specific data.

Due to the aforementioned challenges, text-only adaptation methods are gaining popularity for ASR. In (Sato et al., 2022), the authors use a separate text-encoder network and additional encoder layers on top of an acoustic encoder to infuse text information. In (Stooke et al., 2023), the authors use random encoder features in place of real audio in a Transducer encoder and showcase improvements using text-only data. In (Thomas et al., 2022), the predictor network of the RNN-T model is adapted with text-only data using textogram representations. A different line of work where text-only adaptation is possible uses self-supervised learning (SSL) to train models with joint speech/text representations, such as JOIST (Sainath et al., 2023), MAESTRO (Chen et al., 2022b), and mSLAM (Bapna et al., 2022). Even though these models show promising results in downstream tasks, text-only adaptation remains challenging.

Bridging the gap between supervised and SSL models, the authors in (Higuchi et al., 2022b,a) combined fully supervised architectures with BERT (Devlin et al., 2019). Specifically in (Higuchi et al., 2022b), the authors proposed a BERT-CTC ASR model that expands the CTC model by combining the acoustic representations and BERT representations to perform ASR using self-attention (Vaswani et al., 2017). The BERT-CTC model conditions CTC outputs on BERT embeddings, incorporating explicit linguistic information into training/inference, while maintaining an iterative, non-autoregressive decoding.

In this paper, we present AdaBERT-CTC, a method to adapt BERT-CTC to out-of-domain data

[*] Work done while at AWS AI Labs.

using text only. We do so by training BERT with domain-specific text (Fig. 1) using the BERT-CTC loss. We further explore adding parameter-efficient adapters (Houlsby et al., 2019) to BERT and train only these adapters during the text adaptation phase to make the approach parameter-efficient and deployment friendly (Dingliwal et al., 2021, 2022). At inference time, we replace the original BERT used to train the BERT-CTC model with our adapted BERT. The main contributions of our work are: (1) we propose a simple yet effective text-only domain adaptation method for ASR that leverages the recently proposed BERT-CTC architecture, (2) we explore the behaviour of the BERT-CTC architecture when BERT is further fine-tuned with domain data using masked language model (MLM) loss, (3) we demonstrate the advantages of parameter-efficient adapters to fine-tune the BERT module, and (4) we present that our text-only domain adaptation approach complements the use of a domain-specific language model.

2 BERT-CTC

2.1 Model details and training steps

BERT-CTC adds to the CTC model by leveraging linguistic information from BERT embeddings. Similar to RNN-T (with an audio encoder and a prediction network serving as an internal language model), in BERT-CTC, the audio representations from the audio encoder and the text representations from BERT are fused to estimate the distribution over alignments. In contrast to RNN-T, in the BERT-CTC model, attention layers stacked over the fused representations help learn the masked (or partially observed) sequence (Higuchi et al., 2022b).

Let $X = (x_t \in R^D \mid t = 1, ..., T)$ be an input sequence of length T, and $\widehat{W} = \{\widehat{w}_n \in V \cup \texttt{[MASK]} \mid n = 1, ..., N\}$ be the corresponding output sequence of length N with a special mask token [MASK]. Here, x_t is a D-dimensional acoustic feature at frame t, $\widehat{w}_n$ is an output token at position n, and V is a vocabulary. Defining the alignment as $a = \{a_1, a_2, ..., a_T \in V \cup \texttt{[BLANK]}\}$, BERT-CTC computes the likelihood of the target sequence, W:

$$\mathcal{P}_{\text{BCTC}}(W|X) = \textstyle\sum_{\widehat{W} \in a(W)} \mathcal{P}(W|\widehat{W}, X)\mathcal{P}(\widehat{W}|X), \quad (1)$$

$$\mathcal{P}(W|\widehat{W}, X) = \textstyle\prod_{t=1}^{T} \mathcal{P}(a_t|BERT(\widehat{W}), X), \quad (2)$$

where $a(W)$ covers W with all possible masking patterns. Here, we interpret $p(\widehat{W}|X)$ as a prior distribution of sequences consisting of observed tokens that are easily recognized only from speech input. $BERT(\widehat{W})$ is the output of BERT representing the distribution of target sequences. Further, the conditional probability $\mathcal{P}(a_t|BERT(\widehat{W}), X)$ can be computed using the softmax function as:

$$\mathcal{P}(a_t|BERT(\widehat{W}), X) = \sigma(SelfAtt_t(H^{ae}, H^{BERT})), \quad (3)$$

where H^{BERT} are the embedded masked BERT tokens, H^{ae} are the representations from the acoustic encoder, and $\sigma(\cdot)$ is the softmax function. We concatenate these representations before feeding them to the self-attention layers.

The BERT-CTC objective function $\mathcal{L}_{\text{BCTC}}$ is defined by the negative log-likelihood Eq. 1 expanded:

$$-log \textstyle\sum_{\widehat{W}} \sum_a \mathcal{P}(A|W, X)\mathcal{P}(W|\widehat{W})\mathcal{P}(\widehat{W}|X). \quad (4)$$

During training we handle the intractability of Eq. 1 by randomly masking the text transcript before using Eq. 3 to compute the conditional probability and train using CTC loss. For more training details, please refer to (Higuchi et al., 2022b).

2.2 Inference

During inference, we use an iterative masked predict algorithm assisted by CTC inference for decoding the target tokens. The algorithm first initializes a target sequence with an estimated length, which is then followed by $k = 1, ..., K$ iterations of token masking and prediction steps.

Initialization (k = 1): BERT-CTC decoding requires the length of a target sequence, $\widehat{N}$, to be given in advance. The target length is predicted through CTC-only greedy decoding of audio encoder output. Given this estimated sequence length, a masked sequence $\widehat{W}(k = 1)$ is initialized by filling all $\widehat{N}$ positions with the mask token [MASK]. This masked sequence is passed through BERT and fused with the acoustic representations and then fed through the self-attention module to get an initial hypothesis via CTC greedy decoding.

Iterative Decoding: Given a current prediction $\widetilde{W}(k)$, tokens having low probability scores are masked with [MASK], which results in the next masked sequence. This masked sequence is again fed through BERT, fused with the acoustic representations and decoded using the self-attention + CTC module to get the next predicted sequence. This iterative greedy decoding is done for an additional $K - 1$ steps.

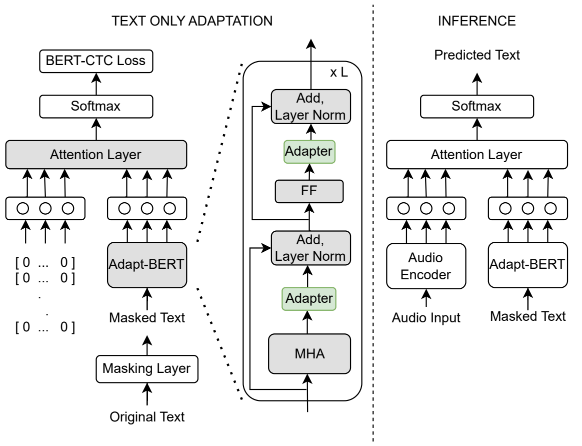

Figure 1: Schematic diagram of our proposed AdaBERT-CTC method using adapters. The grey blocks are frozen and the green blocks are trained during text-only adaptation. During adaptation, we pass zero-valued speech features and masked text into AdaBERT-CTC for text-only adaptation. At inference time, we iteratively pass the input audio features and masked text into the model.

3 AdaBERT-CTC: Adapted BERT-CTC

The BERT-CTC model conditions the output prediction on both the acoustic information and the BERT embeddings of the predicted sequence from the previous iteration. Our adaption approach uses text to modify the BERT module of a trained BERT-CTC model without changing the other parameters.

Our AdaBERT-CTC method is comprised of three steps: (1) training a base BERT-CTC model with minimal changes to the original training method; (2) adapting only the BERT model with our text-only adaptation methods while keeping the rest of the trained BERT-CTC network frozen (which we call Adapt-BERT); and (3) at inference time, we replace the original BERT model with our adapted BERT model, making no other changes to the original BERT-CTC inference framework. We call this method AdaBERT-CTC. In the next subsections, these three steps are further described.

3.1 Training the base BERT-CTC model

The training of the base BERT-CTC model is nearly identical to the original implementation (Higuchi et al., 2022b), but with a minor *but important* modification: in 10% of the batches during training, we mask entire audio embeddings and only provide text embeddings to the attention layer. Masking these ensures that the self-attention layer can handle text-only inputs in the absence of audio embeddings, which is necessary for text-only adaptation.

3.2 Adapting BERT-CTC

Zeroed speech features: Fig. 1 shows a schematic of AdaBERT-CTC. The model uses a self-attention layer to attend to both the acoustic embeddings and the textual embeddings. Since there is not an explicit fusion between the two embeddings, during the text-adaptation step we disregard the acoustic embeddings (Eq. 3) and have the self-attention layer attend to only the text embeddings from Adapt-BERT:

$$\mathcal{P}(a_t|BERT^*(\widehat{W}^*), X) = \sigma(SelfAtt_t(\vec{0}, H^{BERT^*})), \quad (5)$$

where $\sigma(\cdot)$ is the softmax function, and $\widehat{W}^*$, H^{BERT^*}, and $\vec{0}$ represent the masked text from the new domain, masked text embeddings from the Adapt-BERT model, and $\vec{0}$ vectors for the acoustic representations, respectively.

Text-only adaptation: For text-only adaptation, we only adapt the BERT model and freeze the rest of the BERT-CTC model parameters. We first obtain text from the new domain and then mask the text before using it as input to the BERT-CTC model without the audio information. Essentially, the model is trying to predict the full text from the partially masked text. The adapted BERT model is then trained with the same BERT-CTC loss used to train the base BERT-CTC model. In this scenario, the adaption step is similar to masked-language modeling since the self-attention layer is trying to predict the full text from the partially masked text. The simplicity of our approach lies in not requiring any additional optimization procedures from the one used during training, which mitigates any mismatch between adaptation and training objectives. For inference, we use the adapted BERT instead of the original pre-trained BERT, while keeping the rest of the base BERT-CTC the same.

3.3 Efficiently adapting BERT-CTC

To efficiently adapt BERT-CTC, we add parameter-efficient bottleneck adapters (Houlsby et al., 2019) to BERT and train the adapters (instead of full fine-tuning) using BERT-CTC loss. Adding adapters have shown to be effective for multiple downstream tasks (He et al., 2022). Specifically, bottleneck adapters are added after the multi-head attention and feed-forward layers in the transformer modules inside the BERT.

4 Experimental Setup

In this section, we describe the data and experiments performed to showcase the use of our proposed model and method.

4.1 Datasets

Training: We use two different training datasets to train separate base BERT-CTC models: (1) Librispeech 960hrs (LS) (Panayotov et al., 2015) sampled at 16 kHz, and (2) an internal dataset comprising 10000hrs of multi-accented English speech (10k hrs). The internal dataset has a mix of 8kHz (upsampled to 16KHz) and 16kHz audio. We use this dataset to obtain results with a model trained on more acoustic diversity than Librispeech (which contains clean read speech from audiobooks).

Adaptation and evaluation: We test our adaptation method on three datasets: (1) SLURP (Bastianelli et al., 2020), which contains 50628 training, 8690 development and 13078 test utterances. SLURP is a publicly available multi-domain dataset with single turn user interactions with a home assistant; (2) DSTC-2 (Henderson et al., 2014), which contains 11236 training, 3816 development and 9551 test utterances. The utterances are related to the restaurant domain; (3) WSJ (Paul and Baker, 1992), which contains 37416 training, 503 development (dev93) and 213 test (eval92) utterances drawn from WSJ news (train_si284). All datasets are sampled at 16kHz.

4.2 Input features

We initially experiment with standard 80-dimensional log-mel filterbank features obtained using a 25-millisecond Hamming window, 10-millisecond hop size and 512-point discrete Fourier transform as input features. We also experiment using WavLM Large (Chen et al., 2022a) representations as input features to improve generalization to different acoustic conditions. WavLM is pre-trained with the objective of masked speech prediction and denoising and has shown to be robust under noisy conditions.

4.3 Model configuration

The model configuration and training setup for the base BERT-CTC model closely follows (Higuchi et al., 2022b). For the audio encoder, we use a 12-layer Conformer (Gulati et al., 2020) architecture encoder. We use the BERT$_{\text{BASE}}$ model provided by HuggingFace (Wolf et al., 2020) for the text encoder. Finally, the self-attention module to combine the audio and text embeddings is a 6-layer Transformer encoder. All the experiments are done using the ESPnet (Watanabe et al., 2018a, 2021) recipe (Higuchi et al., 2022b).

4.4 Baselines

BERT-CTC: We use the BERT-CTC model as our initial baseline. This model is trained on paired data without any text adaptation. We evaluate the BERT-CTC model on Librispeech's test *clean* and *other* datasets to showcase the performance on a well-known dataset.

BERT-CTC + offline MLM adaptation: In this approach, we explore the performance when using a BERT model that has been fine-tuned offline using the masked language model (MLM) loss (Devlin et al., 2019) with text from the adaptation training sets. During inference, we replace the BERT model in BERT-CTC with this MLM fine-tuned BERT model. This allows us to compare (offline) MLM fine-tuning with the proposed approaches.

4.5 BERT-CTC Adaptation setup

During text-only adaptation, we adapt the BERT model and the rest of the BERT-CTC model parameters are frozen. We explore fully fine-tuning BERT and only training the added adapters. For adaptation, we randomly mask the text before passing it through BERT and adapt BERT using the BERT-CTC loss in Eq. 1. We describe three adaptation setups using our proposed approach below.

AdaBERT-CTC: After training BERT-CTC, we fine-tune the entire BERT module with the BERT-CTC loss using the method in Section 3.2

AdaBERT-CTC + adapters: We are also interested in studying a parameter-efficient approach for text adaptation. For this, instead of fine-tuning the entire BERT model, we add bottleneck adapters into BERT and train only the adapters using BERT-CTC loss while freezing everything else. We select the adapter sizes based on the validation set performance. In most cases, the selected adapters have less than 5% of the total number of parameters.

AdaBERT-CTC + adapters + offline MLM adaptation: Lastly, we modify the approach mentioned above by replacing the original BERT model with MLM fine-tuned using BERT model. Then we train the adapters using our BERT-CTC loss.

4.6 Inference setup

During inference, we set the total number of BERT-CTC decoding iterations to $K = 5$, as we observe that the performance does not improve beyond that. We first evaluate with greedy decoding to assess the influence of using text-only data for adapting our CTC model. Next, we combine beam search decoding to decode the CTC outputs with an external language model (LM) through Shallow Fusion (SF) with a LM weight of 0.6. This allows us to investigate whether our text-only adaptation approach can offer additional advantages. We perform SF with a transformer LM that is trained on the out-of-domain text of each of the individual datasets. The out-of-domain text used to train each of the domain-specific LMs is the same text used in our AdaBERT-CTC text-only adaptation approach. Each LM is a 4 layer transformer model and is trained using the standard setup in ESPNet (Watanabe et al., 2018b).

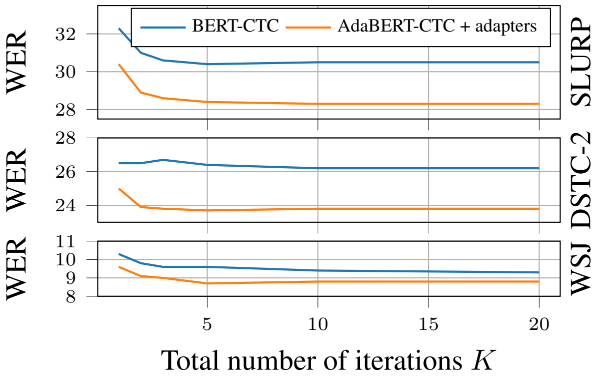

Figure 2: WER on the test sets using different number of total iterations during decoding. For each value of K, we restart the decoding.

5 Results and Analysis

5.1 AdaBERT-CTC: full fine-tuning versus adapters

Table 1 shows the text-only adaptation results that highlight the relative WER improvement (WERR) from adapting the BERT model using our AdaBERT-CTC method. We show results for the base BERT-CTC model trained on the Librispeech 960hrs and our internal 10k hrs dataset and the results on Librispeech test for reference. This model performs poorly on SLURP and DTSC-2, which we attribute to the acoustic and linguistic mismatch. We observe that adapting the BERT-CTC model trained on Librispeech using log-mel filterbank features with our fully fine-tuned adapted BERT shows 12%, 14%, and 5% WERR on SLURP, DSTC-2, and WSJ, respectively, compared to the original BERT-CTC model. When we replace log-mel filterbank features with WavLM features as input, WER reduces across all approaches. With WavLM features, using the fully fine-tuned BERT adapted by our AdaBERT-CTC method improves by a WERR of 8%, 11% and 5% on SLURP, DSTC-2 and WSJ respectively, compared to the base BERT-CTC model. However, we notice that the base BERT-CTC model trained on the 10k hrs internal data did not benefit from text adaptation on WSJ. We hypothesize that the BERT-CTC models trained using Librispeech has a greater benefit from the text in WSJ since the text from Librispeech and WSJ differ more. Nevertheless, the BERT-CTC models trained on the 10k hrs internal data benefit from the SLURP and DSTC-2 adaptation text and both show 11% WERR.

With respect to parameter-efficient adaptation, in most cases, we observe similar or better performance to fine-tuned BERT. These results indicate that training the adapters achieves comparable performance to fine-tuning the BERT model. Similar to previous studies (Houlsby et al., 2019; He et al., 2022), our results suggest that using adapters is a computationally efficient method to adapt BERT-CTC as compared to fine-tuned BERT.

5.2 The effect of offline MLM adaptation

The results for BERT-CTC with offline MLM adaptation shown in Table 1 indicate that using the training adaptation text to fine-tune the BERT model with the MLM loss degrades the performance across most scenarios. We believe this degradation is due to the objective mismatch between the MLM loss and the BERT-CTC loss used in training. Specifically, the new embeddings from the MLM fine-tuned BERT differ from the original BERT embeddings and may no longer be suitable for the attention layers that are kept frozen. However, when we added adapters to the offline adapted BERT and used in conjunction with our AdaBERT-CTC to train the adapters, we observe comparable or sometimes better results than the AdaBERT-CTC + adapters method. Since we originally observe poor performance for BERT-CTC with an offline MLM adaptation, these results suggest that the adapters are beneficial for aligning the MLM fine-tuned representations to be suitable for the self-attention layers. The above experiments highlight our AdaBERT-CTC method, which adapts the BERT model using the BERT-CTC objective, preventing the new embeddings from being unsuitable for the attention layers.

Table 1: WER for models trained on Librispeech 960hrs and 10k hrs of internal data. All results are reported using $K = 5$ and greedy decoding. We include results on the Librispeech test partition using the BERT-CTC model for reference.

Input	**Train**	**Model**	**$\text{BERT}_{\text{BASE}}$ Params**	**Offline MLM Adaptation**	**Dataset**				
					Librispeech (test)		SLURP	DSTC-2	WSJ
					Clean	*Other*	*Test*	*Test*	*eval92*
Log-mel	Librispeech	BERT-CTC	Frozen	✗	4.8	9.3	52.9	50.6	13.2
		AdaBERT-CTC	Frozen	✓			56.7	49.9	14.9
			Fine-tuned	✗			**46.8**	43.5	12.5
			Adapters	✗			47.1	44.6	**12.3**
			Adapters	✓			**46.8**	**43.1**	12.8
	10k hours	BERT-CTC	Frozen	✗	8.7	15.1	33.5	21.5	7.7
		AdaBERT-CTC	Frozen	✓			37.1	22.2	8.4
			Fine-tuned	✗			29.8	19.1	8.0
			Adapters	✗			29.5	**18.5**	**7.6**
			Adapters	✓			**28.8**	19.0	7.9
WavLM	Librispeech	BERT-CTC	Frozen	✗	2.6	4.7	30.4	26.4	9.6
		AdaBERT-CTC	Frozen	✓			30.3	24.9	9.6
			Fine-tuned	✗			**28.0**	**23.6**	9.1
			Adapters	✗			28.4	23.7	**8.7**
			Adapters	✓			28.1	23.7	**8.7**

Table 2: WER for models trained on Librispeech using WavLM features. All results are reported using $K = 5$ with greedy decoding and shallow fusion.

Model	**LM**	SLURP	DSTC-2	WSJ
		Test	*Test*	*eval92*
BERT-CTC	None	30.4	26.4	9.6
	Domain	27.5	22.8	8.4
AdaBERT-CTC	None	28.0	23.6	9.1
	Domain	**26.1**	**21.3**	**8.0**

5.3 Comparison with Shallow Fusion (SF)

Table 2 presents the results of SF using different LMs for BERT-CTC and AdaBERT-CTC. This analysis is performed using WavLM features, and only the adapters inside the BERT are trained during the adaptation process. The LM column indicates whether SF is applied. The following are the two possible values for the LM column: 1) **None**: greedy decoding is performed without the use of any language model, and 2) **Domain**: the LM is trained on the "training text" of the respective domain set. SF with a domain LM with BERT-CTC also provides domain adaptation. Applying SF to the BERT-CTC model output with the domain LM resulted in a WERR of 9.5% to 14%, whereas AdaBERT-CTC without any LM exhibited a WERR of 5% to 11%. These results suggest that AdaBERT-CTC is contributing 42% to 82% of what SF adds on top of BERT-CTC. Conversely, when SF is applied on top of AdaBERT-CTC, a WERR of 7% to 12% is observed. This outperformed the BERT-CTC with SF by a WERR of 5% to 6.6%. This suggests that AdaBERT-CTC not only contributes to most of the improvements observed with SF but also provides additional and complementary benefits when combined with SF.

5.4 The effect on number of decoding iterations in WER

In Fig. 2, we show the WER as a function of the number of decoding iterations. Both BERT-CTC and AdaBERT-CTC + adapters improve over multiple iterations. This is expected as the inputs to the BERT model are initialized with masks for $K = 1$, and the model gets a better context with each new decoding iteration. In this figure, we also observe that our method works better right from the first iteration, showing that bottleneck adapters are already biased to domain specific utterances. For both methods, approximately $K = 5$ iterations are enough to achieve the best performance.

6 Conclusions

In this work, we propose AdaBERT-CTC, a method for adapting BERT-CTC using text-only data. The adaptation approach modifies a trained BERT-CTC model by fine-tuning BERT (while keeping everything else frozen) using BERT-CTC loss with text-only input. Our results show that we achieve up to 14% WERR without the use of an external language model on publicly available datasets. Furthermore, we show that adapting BERT-CTC using our approach in a parameter-efficient manner with bottleneck adapters achieves comparable performance to fully fine-tuning BERT. To understand if our method complements the use of an external language model, we show that combing AdaBERT-CTC with SF improves gives a WERR of 6.6% compared to BERT-CTC with SF. For future work, we plan to evaluate our adaptation performance on real-world datasets.

Limitations

Our approach has the following limitations: 1) In this research, we used the entire training set to adapt our model without exploring how much text data is actually needed to achieve comparable performance. A future study that investigates the impact of varying amounts of text data would be useful to show the potential use case of our method in low-resource scenarios where text data is limited. 2) Our implementation of the text-only adaptation method makes use of the length of the audio segment. The length is used in order to create the zero vector of audio features shown in Eq. 5. Although there have been existing studies that predict the duration of the audio based on the text, we decided to just make use of the real audio length.

Ethics Statement

In this work, we focus on adapting BERT-CTC to well studied datasets using text only data from those datasets. Most of the datasets used for text only adaptation are public domain datasets. One concern is our 10K hour dataset used for pre-training of the base model is randomly sampled, and this data may not fully represent the all end-user use cases. We note that this makes our models susceptible to generating better outputs for certain use cases/users. While we do not explicitly address concerns around bias/sensitive content within our framework to date, we aim to incorporate these considerations, especially in the pre-training data and the text domains used for adaptation, as we move towards real-world scenarios covering a wide range of end-user use cases.

References

Ankur Bapna, Colin Cherry, Yu Zhang, Ye Jia, Melvin Johnson, Yong Cheng, Simran Khanuja, Jason Riesa, and Alexis Conneau. 2022. mSLAM: Massively multilingual joint pre-training for speech and text. *arXiv preprint arXiv:2202.01374*.

Emanuele Bastianelli, Andrea Vanzo, Pawel Swietojanski, and Verena Rieser. 2020. SLURP: A spoken language understanding resource package. In *Proc. EMNLP*.

Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, Jian Wu, Long Zhou, Shuo Ren, Yanmin Qian, Yao Qian, Jian Wu, Michael Zeng, Xiangzhan Yu, and Furu Wei. 2022a. WavLM: Large-scale self-supervised pre-training for full stack speech processing. In *IEEE Journal of Selected Topics in Signal Processing*, pages 1505–1518.

Zhehuai Chen, Yu Zhang, Andrew Rosenberg, Bhuvana Ramabhadran, Pedro J. Moreno, Ankur Bapna, and Heiga Zen. 2022b. MAESTRO: Matched speech text representations through modality matching. In *Proc. Interspeech*, pages 4093–4097.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL-HLT*.

Saket Dingliwal, Ashish Shenoy, Sravan Bodapati, Ankur Gandhe, Ravi Teja Gadde, and Katrin Kirchhoff. 2021. Prompt-tuning in ASR systems for efficient domain-adaptation. In *Proc. WeCNLP*.

Saket Dingliwal, Ashish Shenoy, Sravan Bodapati, Ankur Gandhe, Ravi Teja Gadde, and Katrin Kirchhoff. 2022. Domain Prompts: Towards memory and compute efficient domain adaptation of ASR systems. In *Proc. Interspeech*, pages 684–688.

Alex Graves, Abdel rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proc. ICASSP*, pages 6645–6649.

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 *Proc. Interspeech*, pages 5036–5040.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *Proc. ICLR*.

Yanzhang He, Tara N Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, et al. 2019. Streaming end-to-end speech recognition for mobile devices. In *Proc. ICASSP*, pages 6381–6385.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The second dialog state tracking challenge. In *Proc. SIGDIAL*, pages 263–272.

Yosuke Higuchi, Tetsuji Ogawa, Tetsunori Kobayashi, and Shinji Watanabe. 2022a. BECTRA: Transducer-based end-to-end ASR with BERT-enhanced encoder. *arXiv preprint arXiv:2211.00792*.

Yosuke Higuchi, Brian Yan, Siddhant Arora, Tetsuji Ogawa, Tetsunori Kobayashi, and Shinji Watanabe. 2022b. BERT meets CTC: New formulation of end-to-end speech recognition with pre-trained masked language model. In *Proc. EMNLP*, pages 5486–5503.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proc. ICML*, pages 2790–2799.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. In *Proc. National Academy of Sciences*, volume 114, pages 3521–3526.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: an ASR corpus based on public domain audio books. In *Proc. ICASSP*, pages 5206–5210.

Douglas B Paul and Janet Baker. 1992. The design for the wall street journal-based CSR corpus. In *Proc. Workshop on Speech and Natural Language*.

Tara N Sainath, Rohit Prabhavalkar, Ankur Bapna, Yu Zhang, Zhouyuan Huo, Zhehuai Chen, Bo Li, Weiran Wang, and Trevor Strohman. 2023. JOIST: A joint speech and text streaming model for ASR. In *Proc. SLT*, pages 52–59.

Tara N Sainath, Rohit Prabhavalkar, Shankar Kumar, Seungji Lee, Anjuli Kannan, David Rybach, Vlad Schogol, Patrick Nguyen, Bo Li, Yonghui Wu, et al. 2018. No need for a lexicon? evaluating the value of the pronunciation lexica in end-to-end models. In *Proc. ICASSP*, pages 5859–5863.

Hiroaki Sato, Tomoyasu Komori, Takeshi Mishima, Yoshihiko Kawai, Takahiro Mochizuki, Shoei Sato, and Tetsuji Ogawa. 2022. Text-only domain adaptation based on intermediate CTC. In *Proc. Interspeech*, pages 2208–2212.

Ashish Shenoy, Sravan Bodapati, Monica Sunkara, Srikanth Ronanki, and Katrin Kirchhoff. 2021. Adapting long context NLM for ASR rescoring in conversational agents. In *Proc. Interspeech*.

Adam Stooke, Khe Chai Sim, Mason Chua, Tsendsuren Munkhdalai, and Trevor Strohman. 2023. Internal language model personalization of E2E automatic speech recognition using random encoder features. In *Proc. SLT*, pages 213–220.

Samuel Thomas, Brian Kingsbury, George Saon, and Hong-Kwang J. Kuo. 2022. Integrating text inputs for training and adapting RNN Transducer ASR models. In *Proc. ICASSP*, pages 8127–8131.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. Neurips*.

Shinji Watanabe, Florian Boyer, Xuankai Chang, Pengcheng Guo, Tomoki Hayashi, Yosuke Higuchi, Takaaki Hori, Wen-Chin Huang, Hirofumi Inaguma, Naoyuki Kamo, et al. 2021. The 2020 ESPNet update: New features, broadened applications, performance improvements, and future plans. In *Proc. IEEE Data Science and Learning Workshop (DSLW)*, pages 1–6.

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. 2018a. ESPnet: End-to-end speech processing toolkit. In *Proc. Interspeech*, pages 2207–2211.

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. 2018b. ESPnet: Endto-end speech processing toolkit. In *Proc. Interspeech*, pages 2207–2211.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proc. EMNLP*, pages 38–45.

Yu Zhang, James Qin, Daniel S Park, Wei Han, Chung-Cheng Chiu, Ruoming Pang, Quoc V Le, and Yonghui Wu. 2020. Pushing the limits of semi-supervised learning for automatic speech recognition. In *Proc. Neurips*.