Received 20 April 2023, accepted 24 June 2023, date of publication 30 June 2023, date of current version 24 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3291343

RESEARCH ARTICLE

Context Conditioning via Surrounding Predictions for Non-Recurrent CTC Models

BURIN NAOWARAT, CHAWAN PIANSADDHAYANON, AND EKAPOL CHUANGSUWANICH
Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330, Thailand

Corresponding author: Ekapol Chuangsuwanich (ekapolc@cp.eng.chula.ac.th)

ABSTRACT Connectionist Temporal Classification (CTC) loss has become widely used in sequence modeling tasks such as Automatic Speech Recognition (ASR) and Handwritten Text Recognition (HTR) due to its ease of use. Recent sequence models that incorporate CTC loss have been focusing on speed by removing recurrent structures, hence losing important context information. This paper presents extensive studies of Contextualized Connectionist Temporal Classification (CCTC) framework, which induces prediction dependencies in non-recurrent and non-autoregressive neural networks for sequence modeling. CCTC allows the model to implicitly learn the language model by predicting neighboring labels via multi-task learning. Experiments on ASR and HTR tasks in two different languages show that CCTC models offer improvements over CTC models by 2.2-8.4% relative without incurring extra inference costs. We have also found that higher order of context information can potentially help the model produce better predictions.

INDEX TERMS CTC, contextualized CTC, non-recurrent, non-autoregressive, automatic speech recognition (ASR), handwritten text recognition (HTR).

I. INTRODUCTION

Context has been extensively proved to be useful for various kinds of sequence modeling problems, such as Automatic Speech Recognition (ASR) [1], Text-to-Speech (TTS) [2], Handwritten Text Recognition (HTR) [3], Neural Machine Translation (NMT) [4], and Language Modeling [5]. In NMT, the same word can have different meanings in different contexts. A NMT system must be aware of the context of an input word to infer its correct meaning. Moreover, when producing an output translation, the model must also be able to produce words that are coherent together. Modeling contexts are usually done using contextual representations and context-aware inferencing algorithms in order to properly model context in the input and output spaces. For ASR and HTR, context awareness is used for making consistent predictions, reducing misspellings, and handling ambiguous input samples [6], [7], [8]. Nevertheless, encapsulating contexts usually comes with additional computational complexity, especially for temporal contexts, which are typically done in a sequential manner.

Incorporating contextual information into models can be done using recurrent models and/or contextual hidden representations. Since strong dependencies between letters within target sequences are usually found in sequence modeling, predictions made by recurrent approaches such as autoregressive (AR), iterative, and beam search decoding are generally better than predictions that are independently produced [6], [9], [10]. However, recurrent models have to predict letters sequentially since they use conditions of previous predictions in order to make the next prediction. This sequential nature can be detrimental to the inference time because the computation cannot be done in parallel. On the other hand, non-recurrent models can produce context-dependent predictions based on contextualized hidden representations, which are distilled from recurrent models [11], [12] or trained by context-dependent objective functions [6], [13], [14]. Although these methods do not directly control the predictions in the output layer, improvements over context-independent decoding could still be observed while retaining parallel decoding capabilities.

Connectionist Temporal Classification (CTC) has been commonly used for training non-autoregressive (NAR) ASR and HTR models due to its effectiveness and efficiency [15], [16]. CTC estimates the probability for the alignment between frame-level predictions and character-level ground

The associate editor coordinating the review of this manuscript and approving it for publication was Chuan Li.

truths without the need for expensive frame-level labels. To make the computation feasible, CTC assumes independence between the frame-level outputs. Such assumption limits the model especially for ambiguous cases that require contextual information in order to resolve. Early works typically use CTC with recurrent networks which helps alleviate this drawback. However, recent works use CTC with non-recurrent models instead of recurrent ones to maximize throughput [17]. Even though this combination worsens the correctness of CTC's predictions, the runtime improvement is often worth the trade-off in latency-sensitive situations. Many works have tried to re-introduce context into these models [9], [18], [19].

Previously in [6], we proposed Contextualized CTC (CCTC), a multi-task leaning framework with a context-dependent training objective, to incorporate contexts into non-recurrent and non-autoregressive (NAR) ASR models. CCTC tries to increase dependencies between predictions by mitigating the conditional independence assumption of the regular CTC loss [15] in a way that preserves parallel decoding capability. Concretely, CCTC frameworks predict left and right letters as well as the middle letter. We have shown that CCTC models produced promising results, especially when dealing with ambiguous predictions. However, we only used CCTC models that considered adjacent letters and left investigations on CCTC with larger context sizes untouched. Moreover, the effectiveness of CCTC on other sequence modeling tasks is unclear, especially the tasks that have different modalities to ASR, such as handwritten text recognition (HTR).

Non-recurrent NAR CTC-based models are also increasingly adopted for HTR with the aim of achieving lower latency [20], [21]. These models are fast and generally have impressive performance. Nonetheless, they suffer from the same disadvantages as non-recurrent NAR ASR. The lack of dependencies can hinder non-recurrent NAR text recognition models from correctly recognizing letters in ambiguous cases. Models with context-dependent prediction usually outperform in this situation [8], [22], [23]. We believe that CCTC has the potential to improve HTR performance in the same manner that it has done for the ASR task.

In this work, we present an extension of [6] on finding optimal context sizes and studying the generalizability of CCTC. We conduct experiments on four corpora including two different tasks, ASR and HTR, and two different languages, Thai and English. We chose a Thai Youtube dataset [6] for Thai-English ASR, and LibriSpeech [24] for English ASR. As for HTR, we used BEST [25], [26] for Thai, and IAM [27] for English. Experimental results show that CCTC models outperform the baseline CTC models, especially when no external language models are applied. A larger context further improves the performance of the models. Further analysis using character-level perplexities shows that, during inference, CCTC models give a higher priority to language-related information, in other words, contexts, than regular CTC models.

To summarize, we make the following contributions.

- We reduce the error rates of CCTC systems by increasing their context sizes, which were previously limited to only one in our previous publication.
- We demonstrate the ability of CCTC to generalize across tasks by showcasing its performance on HTR in addition to ASR, which we have already established as effective in our previous work.
- We formulate a heuristic that effectively reduces the context weight search space, which increases significantly as the context size grows. This eases hyperparameter tuning of large context size CCTC training.
- We present analyses on CCTC behaviors including the relation between CCTC and language modeling, the trade-off between training costs and performance gains, and the impact of forced alignments.

The rest of the paper is organized as follows. We highlight some related works including the regular CTC in Section II. We presented details about our methods, the proposed CCTC, in Section III. We described our experimental setups in Section IV, showed the results in Section V, and provided some discussions in VI.

II. RELATED WORKS

This section describes how traditional ASR and HTR models handle contexts, how they adopt CTC, and how they deal with the shortcomings from context independent training.

A. CONTEXT FOR ASR

Context modeling has always been an important component in the ASR model. Traditional HMM-based ASR models comprise three components: an acoustic model (AM), a lexicon model, and a language model (LM), which model context on different levels. The AM is typically based on context-dependent (CD) units, which model several acoustic units together. On the other hand, the lexicon and LM focus on the word and grammatical structure of the sentence, disambiguating homophones and imperfections in the pronunciations [28].

For models based on deep learning, CTC has been proposed for training end-to-end models. CTC typically produces letters instead of CD units. Context dependencies are modeled implicitly using recurrent hidden states [15]. Transducers [29] and Sequence-to-Sequence models [30], which have been introduced later, explicitly model context by making predictions sequentially based on previous outputs. However, sequential prediction can become a computational bottleneck as the model size increases. Additional language model rescoring or beam search can be further introduced to reinforce context modeling with the cost of additional computation.

Recently, end-to-end ASR models have become enormous and require extensive computation resources [31]. The community interests have increasingly shifted towards models with non-recurrent and NAR, in which no further sequential decoding and post-processing are applied. Though non-recurrent CTC models have low decoding latency, they suffer from performance degradation. Potential remedies include using rescoring or iterative decoding where successive refinements are performed on the previous outputs [9], [17], [32].

B. CONTEXT FOR HTR

HTR frameworks heavily rely on contexts as they have to extract a sequence of dependent characters from an image. One of the early approaches is a HMM-based framework, which is analogous to lexicon-free ASR models [33], [34].

The combination of CTC and RNN was firstly adopted as an alternative to the existing HMM frameworks [16], [35], [36]. As CTC lacks dependencies, it was used with context rich models such as RNN and multi-dimensional RNN [37], [38], [39], [40]. Recently, non-recurrent models have shown promising results while reducing the latency. This introduces a wave of research based on non-recurrent models as they have no computation bottlenecks [20], [21], [41].

C. CONNECTIONIST TEMPORAL CLASSIFICATION

CTC [15] is an alignment-free objective function for sequence-to-sequence tasks such as ASR and HTR, where the input sequence length, T, is greater than or equal to the output sequence length, U. Given a set, $\mathbf{A}$, that contains the entire allowed letters in the language, the goal of CTC models is to produce a variable-length letter sequence, $y = (y_1, y_2, \ldots, y_U) : y_u \in \mathbf{A}$, from an input sequence, $x = (x_1, x_2, \ldots, x_T) : x_t \in \mathbb{R}^M$. In order to do so, an extra blank alphabet, ϵ, is introduced for handling blank spaces in images or silences in audios. CTC models produce a frame-level intermediate output path, $\pi = (\pi_1, \pi_2, \ldots, \pi_T) : \pi_t \in \mathbf{A}' = \mathbf{A} \cup \{\epsilon\}$. A letter that spans many consecutive input frames will cause consecutive duplicate outputs. Blank tokens can also be output in-between double letters to distinguish them from consecutive duplicates. Finally, the output path, π, is post-processed to an inferred sequence, $y = \mathcal{B}(\pi)$, using a mapping function $\mathcal{B} : \mathbf{A}' \rightarrow \mathbf{A}$, which merges consecutive duplicates and removes excessive blank tokens.

CTC models are trained using the CTC loss, which is the negative log probability of *all valid* paths for the ground truth. The idea is to strengthen the probability of any path that can be mapped to the target sequence instead of relying on the ground truth alignment. CTC assumes conditional independence between tokens within a path to ease the calculation. Thus, the probability for a path, $P(\pi|x)$, can be factorized as a product of the probability in each position as shown in (1). We depict the calculation of the CTC loss in (2).

$$P(y^*|x) = \sum_{\pi \in \mathcal{B}^{-1}(y^*)} P(\pi|x) = \sum_{\pi \in \mathcal{B}^{-1}(y^*)} \prod_t P(\pi_t|x) \quad (1)$$

$$\mathcal{L}_{\text{CTC}} = -\log P(y^*|x) \quad (2)$$

where an inverse function, $\mathcal{B}^{-1}$, is used to find valid paths, and the ground truth sequence is $y^* = (y_1^*, y_2^*, \ldots, y_U^*)$: $y_u^* \in \mathbf{A}$.

The independence assumption encourages the model to isolate its predictions. Consequently, CTC models mostly produce blank tokens, $\pi_t = \epsilon$, and only predict the actual alphabets, $\pi_t \in \mathbf{A}$, when they are extremely confident. Thereby, actual alphabets in paths have low dependencies as alphabets are surrounded by non-informative blanks, making context conditioning without external tools difficult.

D. CONTEXT-DEPENDENT CTC

Incorporating context dependencies into CTC models has been mostly based on using subword modeling such as Byte-Pair Encoding [42], WordPiece [43], and context-dependent output units [44], which are the composition of several letters. A natural extension to contextualized CTC is to use these subwords as the alphabet. The transcriptions are pre-tokenized based on available output units and used as ground truths for training CTC models. Moreover, different letter segmentations, such as different letter-grams [3] or different WordPiece sizes [45], can be used together to train a single CTC model simultaneously via multi-task learning, capturing different scales of context. Unlike our work, the distinct prediction heads have no relationship between them because they are trained by dedicated CTC losses on different pre-tokenized targets.

CTC has also been extended to handle modeling inter-dependencies between output letters. Gram CTC [46] introduces a modification of the CTC loss that can aggregate the different possible segmentations of the output tokens on-the-fly. Recurrent transducer [47] autoregressively wraps posteriors of a CTC encoder with a language model and trains both modules together. Imputer [32] iteratively predicts missing letters in previous outputs. The Imputer model is trained using a modified CTC that is suitable for partial transcriptions, which mimics incomplete predictions. However, this line of work explicitly model inter-dependencies and is very distinct from our work that implicitly encourages context dependencies for CTC. Closest works to ours are [48] and [49] that predicted future ground truths for recurrent hybrid ASR models.

III. METHODOLOGY

In this section, we present the CCTC framework, which simultaneously predicts the middle and surrounding letters. We use CTC and cross entropy (CE) losses for training the main and context predictions, respectively. CE training generally requires an alignment between input frames and the output token, which is not available in the CTC framework. The main contribution of CCTC is an algorithm that can obtain target labels for the CE loss on-the-fly. We provide further details of CCTC training and context label generating in III-A and III-B, respectively.

A. CONTEXTUALIZED CONNECTIONIST TEMPORAL CLASSIFICATION LOSS

CCTC softens the strength of independent predictions by implicitly introducing context conditioning to non-recurrent NAR models without the need of sequential processing. CCTC allows the model to predict the output as well as estimate the contexts for its own predictions. The context estimation raises the awareness of surroundings, which helps mitigate the interference between consecutive outputs and improves the coherency of the predicted sequence. The contexts are predicted simultaneously with the actual prediction in a multi-task manner since we do not want the model to wait for the previous outputs as in sequential decoding.

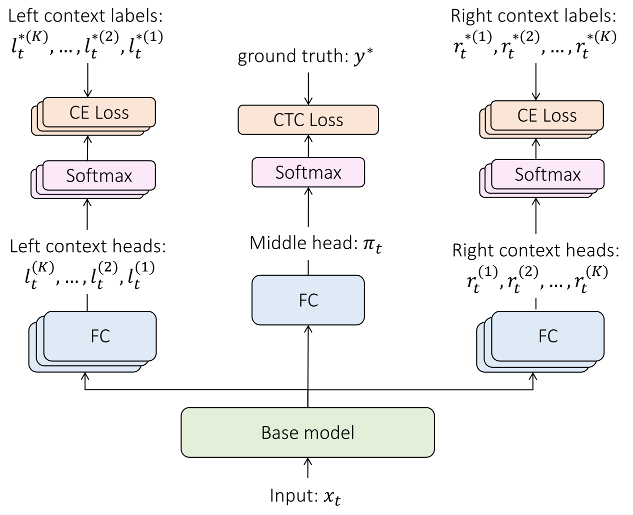


FIGURE 1. The CCTC architecture used in this work. The baseline models are modified by adding $2K$ extra prediction heads. The models are now aware of contexts as it learns to predict left, middle, and right characters simultaneously.

The overview of our framework is depicted in Fig. 1. A CCTC model has three groups of prediction heads: middle, left context, and right context. Given an input x_t, the output from the middle head (π_t) is the main output found in a typical CTC model. The left (l_t) and right (r_t) characters, predicted by context heads, are the likeliest contexts for the middle letter. We gather outputs from the middle head to form a single sequence for CTC training. The context heads are separately optimized for each input frame using context loss.

The context loss is the negative probability for the context labels, which is the CE loss for the frame-level context references. We denote l_t^* and r_t^* as labels for left and right context heads, respectively. The context loss, $\mathcal{L}_{\text{CT}}$, can be defined as shown in (3).

$$\mathcal{L}_{\text{CT}} = -\sum_t \alpha \log P(l_t^*) + \beta \log P(r_t^*) \tag{3}$$

where α and β are weights for the left and right contexts. In addition to contextualizing adjacent letters, the context size can actually be further increased to any arbitrary size. For the context size of $K \in \mathbb{Z}$, the CCTC$^{(K)}$ model predicts K consecutive letters to the left and K consecutive letters to the right. We introduce the superscript (k) for l_t^*, r_t^*, α, and β to indicate that $l_t^{*(k)}$ and $r_t^{*(k)}$ are kth left and right labels for the input x_t, respectively. The $\alpha^{(k)}$ and $\beta^{(k)}$ are weights for $l_t^{*(k)}$ and $r_t^{*(k)}$. We can construct the general form of the context loss as follows:

$$\mathcal{L}_{\text{CT}} = -\sum_t^T \sum_k^K \alpha^{(k)} \log P(l_t^{*(k)}) + \beta^{(k)} \log P(r_t^{*(k)}) \tag{4}$$

Finally, the training loss for CCTC frameworks is the summation of the CTC loss and the normalized context loss as shown in (5).

$$\mathcal{L}_{\text{CCTC}} = \mathcal{L}_{\text{CTC}} + \frac{\mathcal{L}_{\text{CT}}}{U} \tag{5}$$

For inference, only the middle head is kept. Thus, CCTC models have the same runtime as the base model, which is especially important for low latency applications. CCTC can also be incorporated into any model structure and decoding scheme without any additional changes since CCTC only affects the training stage. This is the main advantage of CCTC over other methods that also try to incorporate context.

B. ACQUIRING CONTEXT LABELS

The CTC algorithm is alignment-free which means that there are no explicit frame-level ground truths for the context heads. Therefore, the frame-level labels for the context losses are obtained from the paths that are predicted by the middle head. In other words, the context heads aim to predict the outputs produced by the middle head for the neighboring frames. However, the model may learn little to no context information from learning to predict blank tokens. Thus, we opt to train the context heads with dense character supervision from the prediction, $y = \mathcal{B}(\pi)$, instead. Concretely, the contexts $l_t^{*(k)}$ and $r_t^{*(k)}$ are the kth-nearest characters to the left and right of π_t that are not a blank token or a consecutive duplicate. The labels can be retrieved by conducting a naive search on a path. However, a naive search is computationally expensive as it has the time complexity of $O(T)$ for every position t, which results in the total of $O(T^2)$. Alternatively, we propose to obtain the labels using an efficient algorithm that operates in $\Theta(KT)$.

To reduce the time complexity in the label obtaining procedure, we propose to search on a dense path, $h = (h_1, h_2, \dots, h_L) : L \leq T$, instead of the usual CTC path, π. A dense path, h, is an intermediate result of applying $\mathcal{B}$, in which all consecutive duplicates are already merged but blanks are not yet removed. In order to search on a dense path, we have to know where the surroundings of π_t are in h. To do so, we store the relation between a path, π, and a dense path, h, in an index list, $p = (p_1, p_2, \dots, p_T) : p_t \in [1, L], p_t \leq p_{t+1}$. An index p_t indicates that a letter h_{p_t} is derived from a path token π_t. As we know that h_{p_t} is the representative of π_t, we can directly conduct a naive search on h using the predetermined start position of p_t. Since a dense path has no consecutive duplicates and only two categories of characters exist: the blank and the actual alphabet, we can obtain kth non-blank characters within $2K$ operations, regardless of the input length. In total, we can obtain the labels for an input length T within a tight bound of $\Theta(KT)$. We use blanks as context labels for the edge cases where the dense path has insufficient context on either side. Since blanks have no other uses for context prediction, the model can learn to exclusively use blanks for *no-context* scenarios. We summarize the algorithm in Algorithm 1 and demonstrate this process with an example in Fig. 2.

C. MODEL INITIALIZATION

Since the labels for context heads are obtained from the main head predictions on-the-fly, optimizing context loss for random networks may cause training difficulties due to noisy labels. In our experiments, we used pretrained weights as the initialization to prevent the issue and reduce the training time. This also highlights the use case where one might choose to further improve an existing CTC-trained model by incorporating additional CCTC training afterwards.

IV. EXPERIMENTAL SETUPS

In this section, we presented the experimental setups designed to highlight the effectiveness of the proposed CCTC model over the baseline regular CTC model. In order to show the generalizability of the proposed method, we evaluated CCTC using two sequence modeling problems, namely ASR and HTR, in two different languages, Thai and English.

The section is organized as follows. We described datasets in IV-A. We provided the training details for each dataset in IV-B. The methods we used for selecting context losses are described in IV-C. The setups for LMs, metrics, statistical testing, and baselines are presented in IV-D, IV-E, IV-F, and IV-G, respectively.

A. CORPORA

For our experiments, we used two ASR corpora and two HTR corpora as follows.

1) ASR CORPORA

We investigated two kinds of ASR tasks, monolingual and code-switching (CS). Monolingual speech is spoken audio in which only one language is presented. In contrast, many languages can be found within a single utterance for code-switched speech. Context is especially important for the code-switched dataset in order to produce a coherent spelling. We used LibriSpeech [24] for monolingual English and Thai YouTube [6] for Thai-English code-switching corpus. Both datasets have 16kHz sampling rates and 16-bit depth audios. Table 1 summarizes the ASR corpora.

LibriSpeech [24] is a common ASR corpus for English. We used the *train-clean-100* subset for small-scale experiments and the total 960 hours for large-scale comparisons. The evaluations were conducted on *dev-clean* and *test-clean*.

Thai YouTube [6] is a 200-hour speech corpus taken from public Thai podcast YouTube channels. The training subset

Algorithm 1 Acquiring Labels for Context Losses

```
Given: $\pi$ - CTC path, $K$ - context size
Result: $l^*$ - left context labels, $r^*$ - right context label
$T \leftarrow \text{Length}(\pi)$
$h \leftarrow H(\pi)$    ▷ $H$ merges consecutive duplicates.
$p \leftarrow \text{Indexing}(h, \pi)$ ▷ $p_t$ indicates that $h_{p_t}$ is derived from $\pi_t$.
$t \leftarrow 1$
while $t \leq T$ do
    $l^*_t \leftarrow \text{LeftSearch}(h, p_t, K)$    ▷ $l^*_t = (l^{*(1)}_t, l^{*(2)}_t, \ldots, l^{*(K)}_t)$
    $r^*_t \leftarrow \text{RightSearch}(h, p_t, K)$    ▷ $r^*_t = (r^{*(1)}_t, r^{*(2)}_t, \ldots, r^{*(K)}_t)$
    $t \leftarrow t + 1$
end while
```

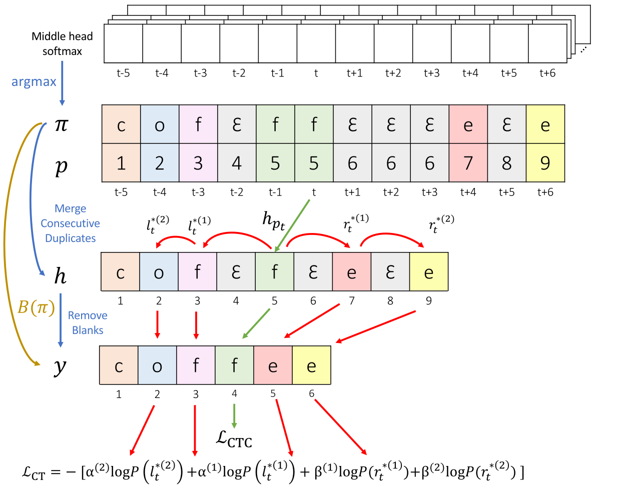

Middle head softmax
argmax
π
p
Merge Consecutive Duplicates
h
B(π)
Remove Blanks
y
$\mathcal{L}_{\text{CTC}}$
$\mathcal{L}_{\text{CT}} = -[\alpha^{(2)}\log P(l_t^{*(2)}) + \alpha^{(1)}\log P(l_t^{*(1)}) + \beta^{(1)}\log P(r_t^{*(1)}) + \beta^{(2)}\log P(r_t^{*(2)})]$

FIGURE 2. Label generation procedure for the CCTC context losses. Given a path from the output of the middle head's softmax, the labels for the left and right heads can be found by searching on a dense path *h*. For frame index *t*, the first target to the left and right are *f* and *e*. The second targets are *o* and *e*, respectively. The correspondences between π and *y* are color coded. A letter, y_i, is derived from the token, π_t, with the same background color.

of Thai YouTube contains both monolingual Thai (TH) and CS Thai-English utterances. As for validation and testing sets, monolingual and CS utterances were separated into TH and CS subsets, respectively. Concretely, the dataset has one TH-CS training set, two validation sets: *dev-th* and *dev-cs*, and two testing sets: *test-th* and *test-cs*. Unlike the previous work [6], we included additional CS utterances into test-cs, increasing the size by three times. Any hyperparameter tunings were done together on the combined validation set.

2) HTR CORPORA

We used two line-level handwritten text corpora, the Thai dataset, BEST, and the English dataset, IAM. An overview of the two corpora are presented in Table 2.

IAM is a standard English HTR dataset [27]. It is composed of grayscale line-level handwritten images from 657 writers. IAM has 79 characters in total, including 26 English lowercase, 26 English capital letters, 10 Arabic numbers, and

TABLE 1. The statistics of the ASR corpora used in this work. Numbers shown are in hours.

Corpora	Train	Valid	Test	#Symbol
LibriSpeech	960	5	5	28
- train-clean-100 subset	100	-	-	28
Thai YouTube	150	23	31	94
- th subset	-	22	24	67
- cs subset	-	1	7	94

17 special symbols. There are several versions of data splitting for IAM. We chose the one with 6482 training images, 976 validation images, and 2915 test images, similar to [20]. We also followed the data preprocessing methods in [20], including grayscaling the images, normalizing the heights to 64 pixels, and standardizing the intensity values.

BEST is a standard benchmark for handwritten text recognition provided by The National Electronics and Computer Technology Center (NECTEC) [25], [26] as part of the 2019 Thai handwritten text recognition contest[1]. BEST corpus has a total of 3550 images, including 417 unique sentences, 71 unique Thai alphabets, and 10 Arabic numbers. We combined the original BEST dataset with additional 220 images. We resized the height and width of the images to 64 and 625 pixels, respectively. We converted them to grayscale and standardized their intensities, following the same procedure as in IAM. We applied brightness and contrast augmentation to the images. The dataset has two test sets, *test-seen* and *test-unseen*. The test-seen set comes from the same source as the training and validation sets. However, test-unseen comes from a completely different domain.

B. IMPLEMENTATION DETAILS

In this section, we presented models, training conditions, and other details for experiments conducted on each dataset.

1) LIBRISPEECH

We chose the pretrained Wav2Letter+ model from [6] as the base model for train-clean-100 experiments. We added context heads to the base model and additionally trained for 100 epochs using a learning rate of 1e−4.

Adam optimizer [50] and exponential learning rate scheduler were kept identical to the previous work. We re-tuned LM weights for the baseline CTC and CCTC$^{(1)}$ using grid search with a wider boundary and a finer step size using the validation set.

As for LibriSpeech 960 hours, we used the implementation of QuartzNet-5 $\times$ 5 [18] from NeMo toolkit [51]. We trained the base model for 300 epochs from scratch using the default configuration[2]. Afterwards, we added the extra heads and context losses and resumed the training for a total of 600 epochs. The training was done using 8 GPUs with a batch size of 32 per GPU.

[1]https://www.nectec.or.th/
[2]https://ngc.nvidia.com/catalog/models/nvidia:nemospeechmodels

TABLE 2. The number of images for each HTR corpus.

Corpora	#Train	#Valid	#Test	#Symbol
IAM [27]	6482	976	2915	79
BEST [25], [26]	2736	342	692	81

2) THAI YOUTUBE

The pretrained Wav2Letter+ model from [6] was used for Thai YouTube. We attached context heads to the pretrained network and additionally trained the model for 100 epochs. Following [6], we used the initial learning of 4e−5 with an exponentially decaying rate of 0.98 every epoch and a mini-batch size of 64. We took the baseline CTC and CCTC$^{(1)}$ models from [6] and tuned new LM weights for them using the larger search space.

3) IAM

We used Gated Fully Convolutional Networks (GFCN) proposed in [20] as the base model for the IAM dataset. The model comprises 12 2D-depthwise separable convolutional [52] and 18 2D-convolutional layers. We added the context heads to the pretrained networks, provided by the authors[3], and resumed the training for an additional 400 epochs. We followed the training batch size of 2, the learning rate of 1e−4, and all other training hyperparameters as described in the original paper. The model was implemented using PyTorch [53]. We selected the checkpoint with the best validation CER for the comparison.

4) BEST

The base model for BEST consists of 12 layers of 2-dimensional (2D) convolution, 2 layers of 2D depthwise separable convolution [52], and 4 layers of 1-dimensional convolution. We followed the two-step training methodology proposed in [6]. We initially trained the base model using only CTC loss for 300 epochs. We attached context heads to the pretrained CTC network and additionally trained the model for another 400 epochs. Adam optimizer [50] was used with the fixed learning rate of 1e−4 and batch size of 64. The checkpoints with the best validation CER were selected for the comparison.

C. CONTEXT LOSS WEIGHTS

In higher order context losses, tuning the loss weights by grid search becomes impractical. In order to reduce the search space of context loss weights, we propose to derive the high order weights through closed-form formulas, based on the weight of the 1st-order context losses, $\alpha^{(1)}$. We simply set $\alpha^{(1)} = 1$ in most of our experiments as we found this value performs well in general. For maximum gains, one can obtain the better weight $\alpha^{(1)}$ through grid searching within the range of 0.5 to 2.5[4]. As for weights of high order context losses,

[3]https://github.com/FactoDeepLearning/LinePytorchOCR
[4]This range is effective for the NeMo implementation of CTC loss, which is not normalized by the number of letters as in the default setting of PyTorch.

$\alpha^{(k)} : k > 1$, we have tried several heuristic approaches and found two effective methods.

The first approach equally assigns $\alpha^{(1)}$ as weights for every context loss order, $\forall_k \alpha^{(k)} = \alpha^{(1)}$. The second approach sets the highest order context loss weight to one, $\alpha^{(K)} := 1$. Then, we exponentially decrease weights as the order of the context loss shrinks, $\forall_k \alpha^{(k)} = \alpha^{(K)}/2^{K-k}$.

Note that we set the weights of the right and left context losses to be the same, $\forall_k \beta^{(k)} = \alpha^{(k)}$.

D. LM RESCORING

For LibriSpeech, we used the official word-based 3-gram LM. The LM was applied to beam search decoding with a beam size of 64. The LM weights and insertion penalties of each model were tuned on the validation set using grid search from 0.0 to 2.0 and 0.0 to 5.0, respectively. The step size was 0.1 for both hyperparameters.

As for Thai YouTube, the word-level 3-gram LM was trained using text corpora from Thai Wikipedia and the Thai Q&A forum [6]. The beam size was set to 64. LM weights and insertion penalties were obtained through grid search in the same manner as LibriSpeech.

E. EVALUATION METRICS

As for the evaluation, we opted for the standard metrics, Character Error Rate (CER) and Word Error Rate (WER). Both metrics are defined similarly as the Levenshtein distance for correcting the hypothesis into the ground truth over the length of the ground truth. A concrete definition is defined in (6).

$$Lev = \frac{I + D + S}{R} \tag{6}$$

where Lev is Levenshtein distance, and R is the length of references. The number of insertion, deletion, and substitution operations are written as I, D, and S, respectively. The operations are calculated at the character level for CER and word level for WER.

F. STATISTICAL SIGNIFICANCE

We used Sentence-Segment Word Error (MAPSSWE) two-tailed test to evaluate statistical significant differences [54] between baselines and proposed methods. For word-level testing, we used off-the-shelf SCTK implementation[5]. For character-level testing, we treated each character unit as a word. We reported the significant differences between two ASR systems using the significance threshold of 0.05.

G. BASELINES

To evaluate the effectiveness of the proposed method, we compared the performances of CCTC models against the non-contextualized regular CTC model. We used similar architectures and training conditions for both methods. The distinctions between baseline and the proposed methods were the auxiliary context prediction heads, which were not used in the inference stage.

[5]https://github.com/usnistgov/SCTK

TABLE 3. The WER (%) results for the LibriSpeech 100 hours setting. The $*$ symbol shows a significant difference compared to the baseline CTC.

Model	dev-clean			test-clean		
	argmax	beam	3-gram	argmax	beam	3-gram
CTC	22.06	21.95	**15.00**	21.97	21.87	15.11
CCTC$^{(1)}$	21.27*	21.23*	15.02	21.32*	21.20*	**15.06**
CCTC$^{(2)}$	**21.22***	**21.14***	15.09	**21.24***	**21.14***	15.29
CCTC$^{(3)}$	21.36*	21.20*	15.17	21.37*	21.30*	15.31*

TABLE 4. The WER (%) results for the LibriSpeech 960 hours setting. The top row refers to the published results, while the second row refers to our run using the provided code.

Model	dev-clean			test-clean		
	argmax	beam	3-gram	argmax	beam	3-gram
CTC [18]	-	-	-	5.37	-	-
CTC (our run)	6.17	5.70	4.03	6.43	5.94	4.29
CCTC$^{(2)}$	**6.07**	**5.57**	**3.98**	**6.16***	**5.76**	**4.27**

V. EXPERIMENTS

A. AUTOMATIC SPEECH RECOGNITION

We start by comparing the effect of CCTC on ASR in two corpora with three different decoding algorithms: argmax, beam search (beam), and beam search with 3-gram language modeling (3-gram).

1) LIBRISPEECH

CCTC models consistently outperform the baseline CTC in the scenario where LM rescoring was unavailable as illustrated in Table 3. A larger context can be beneficial as shown by how CCTC$^{(1)}$ is outperformed by CCTC$^{(2)}$. The CCTC$^{(2)}$ model tends to achieve the best results for this setting with a relative improvement over the baseline of 3.8% and 3.3% on dev and test sets, respectively. On the other hand, the baseline CTC is slightly superior to the CCTC$^{(2)}$ model in the development set when the 3-gram language model was applied during beam search. Since additional context information is included in the decoding via the 3-gram language model, the benefits of CCTC can become smaller in this setting. However, the CCTC$^{(2)}$ model still outperforms the baseline on the test set.

Similar results can also be found for the larger 960-hour setup as depicted in Table 4. Overall, the CCTC$^{(2)}$ model is consistently superior to the baseline by around 4.2% and 3.0% relative using argmax and beam search decoding, respectively. If LM rescoring is applied, CTC and CCTC models are more comparable with each other. We will discuss more about this discrepancy in Section VI-B.

2) THAI YOUTUBE

Table 5 shows the performance of models on the Thai dataset. CCTC$^{(2)}$ is also the best model without LM rescoring. The CCTC$^{(2)}$ model outperforms the baseline by 2.5% on test-th and 2.0% relative on test-cs. With 3-gram LM, CCTC$^{(1)}$ is slightly better than CCTC$^{(2)}$. This is expected since the extra contextual constraint provided by the LM, helps reduce the dependency on the contexts from the model side. Note that

TABLE 5. The WER (%) results for the Thai YouTube corpus setting. The $*$ and $\dagger$ symbols indicate significant differences to the baseline CTC and the CCTC$^{(1)}$ model, respectively.

Model	dev-th			dev-cs			test-th			test-cs		
	argmax	beam	3-gram	argmax	beam	3-gram	argmax	beam	3-gram	argmax	beam	3-gram
CTC [6]	15.01	14.89	13.14	28.02	27.76	24.17	15.66	15.52	13.22	33.85	33.81	30.43
CCTC$^{(1)}$ [6]	14.67*	14.58*	**13.07**	27.57	27.43	**23.79**	15.30*	15.22*	**13.21**	33.38*	33.39*	**30.07***
CCTC$^{(2)}$	**14.62***	**14.55***	13.22	**27.48***	**27.22***	24.09	**15.28***	**15.20***	13.25	**33.17***†	**33.15***†	30.12*
CCTC$^{(3)}$	14.70*	14.64*	13.26*	27.80	27.69	24.13	**15.28***	15.21*	13.30	33.29*	33.35*	30.39†

TABLE 6. The performance comparison on IAM. The $*$ and $\dagger$ symbols indicate significant differences to the baseline CTC and the CCTC$^{(1)}$ model, respectively.

Model	validation		test	
	CER(%)	WER(%)	CER(%)	WER(%)
CTC	5.03	20.44	7.67	27.77
CCTC$^{(1)}$	4.94	19.78	7.61	27.13*
CCTC$^{(2)}$	4.73*†	19.14*†	7.43*†	26.59*†
CCTC$^{(3)}$	**4.69***†	**18.73***†	7.26*†	**25.84***†
CCTC$^{(4)}$	4.79*†	19.31*	**7.25***†	26.27*†

unlike in the English dataset, CCTC outperforms CTC in all decoder settings. This is due to the fact that context is more important in code-switching data than in monolingual in order to correctly predict the language being spoken.

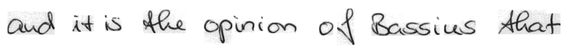
B. HANDWRITTEN TEXT RECOGNITION

In this part, we compare the performance of the model trained using CTC and CCTC losses on both Thai and English HTR datasets. We also present qualitative analyses in order to study the effects of adding context losses.

1) IAM

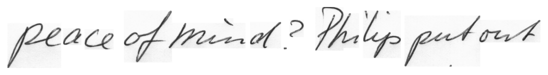
The results on the IAM dataset are summarized in Table 6. It is common on the IAM dataset to present both the CER and WER metrics with only greedy decoding to measure the performance of the model as a standalone. As the size of context increases the model becomes better until the context size of 3. CCTC$^{(3)}$ improves by 5.3% and 7.5% relative to the baseline CTC model on CER and WER, respectively.

Figure 3 shows examples of the differences between model outputs. We found that CCTC models do noticeably better on hard-to-read handwriting. In Fig. 3a, the letter t is highly ambiguous and looks like A. CCTC$^{(1)}$ would observe only the left space and the right letter h, which are inadequate for predicting the correct transcription. After we increased the context width, the high-order CCTC models were able to fix the issue. The sample in Fig. 3b is also very vague, and further contexts are needed to mitigate this problem. In Fig. 3c, CCTC encourages the consistent spellings of numerical letters. This is expected since context is very important to decipher ambiguous handwriting. In a sense, CCTC is able to embed the language model into the model without requiring an explicit LM. It is also interesting to note that the base model has a horizontal receptive field of 240 pixels, which covers roughly 4-7 characters for this dataset. This coincides with the best context size of three.



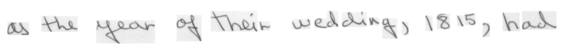
CTC	and it is the opinion of Bassims Ahat
CCTC$^{(1)}$	and it is the opinion of Bassius Ahat
CCTC$^{(2)}$	and it is the opinion of Bassius that
CCTC$^{(3)}$	and it is the opinion of Bassius that
Ground truth	and it is the opinion of Bassius that

(a)



CTC	peace of misnd ? Philips pntout
CCTC$^{(1)}$	peace of misnd ? Philip pntout
CCTC$^{(2)}$	peace of mind ? Philip putout
CCTC$^{(3)}$	peace of mind ? Philip put out
Ground truth	peace of mind ? Philip put out

(b)



CTC	as the year of their weddings 18 IS, had
CCTC$^{(1)}$	as the year of their wedding,s I815s had
CCTC$^{(2)}$	as the year of their wedding, 1515. had
CCTC$^{(3)}$	as the year of their wedding, 1815, had
Ground truth	as the year of their wedding, 1815, had

(c)

FIGURE 3. Selected prediction examples from the IAM test set. The mispredictions are highlighted in color.

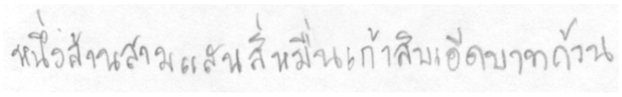
2) BEST

For the BEST dataset, CCTC models consistently outperform the CTC model, as shown in Table 7. We opted for CER as the only evaluation metric since BEST transcriptions were not properly tokenized, and Thai text has no word segmentation standard. CCTC$^{(4)}$ model achieves the lowest validation CER of 9.7%. However, this superior performance does not hold in the test sets. The test-unseen set which comes from a completely different domain does not work well with the implicit LM learned by the model. The best scoring model on the unseen test set is CCTC$^{(2)}$ which gives a good middle ground. Note that a context of 2 characters is still considerably weak as a LM and would not be detrimental even with domain mismatch, since it mostly learns about legal character sequences in the language.

Further inspections suggest that adding context losses can help improve the performances of character segmentation and ambiguous handwriting. Fig. 4a depicts handwriting with very narrow spacing between characters. The CTC model predicts an extra character that has a similar shape to the

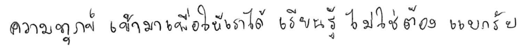
TABLE 7. The argmax CER (%) evaluation on the BEST. The $*$ and $\dagger$ symbols indicate significant differences to the baseline CTC and the CCTC$^{(1)}$ model, respectively.

Model	validation	test-seen	test-unseen
CTC	10.62	11.38	35.03
CCTC$^{(1)}$	9.93*	10.72*	35.15
CCTC$^{(2)}$	9.83*	**10.26***†	**34.79**
CCTC$^{(3)}$	9.76*	10.49*	36.42*†
CCTC$^{(4)}$	**9.71***	10.42*	35.42



CTC	หนึ่งลัทนสามแสนสี่หมื่นเก้าสิบเอ็ดบาทถ้วน
CCTC	หนึ่งล้านสามแสนสี่หมื่นเก้าสิบเอ็ดบาทถ้วน
Ground truth	หนึ่งล้านสามแสนสี่หมื่นเก้าสิบเอ็ดบาทถ้วน

(a) A case with narrow spacing handwriting.



CTC	ความทุกข์ เข้ามาเพื่อให้ีเราได้ เรียนรู้ ไม่ใช่ต้อง แบกรับ
CCTC	ความทุกข์ เข้ามาเพื่อให้เราได้ เรียนรู้ ไม่ใช่ต้อง แบกรับ
Ground truth	ความทุกข์ เข้ามาเพื่อให้เราได้ เรียนรู้ ไม่ใช่ต้อง แบกรับ

(b) A case with ambiguous handwriting.

FIGURE 4. Output comparison between different models on selected examples. Highlights indicate prediction differences.

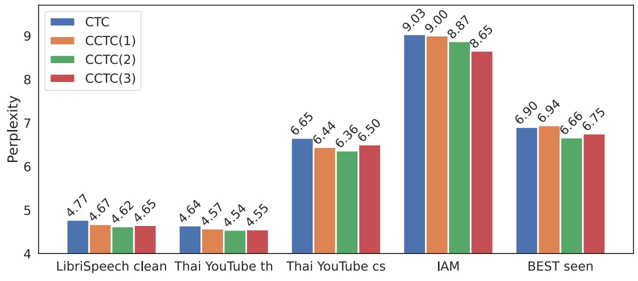


FIGURE 5. Perplexity comparisons between different context sizes. Perplexity scores are computed from the predictions using argmax decoding on each test sets.

Argmax	CTC	nfunction ที่ ใช้ นัน บ่อย บ่อย นะ ค่ะ
	CCTC	function ที่ ใช้ นั้น บ่อย บ่อย นะ ค่ะ
3-gram LM	CTC	function ที่ ใช้ งาน บ่อย บ่อย นะ ค่ะ
	CCTC	function ที่ ใช้ นั้น บ่อย บ่อย นะ ค่ะ
Ground truth		function ที่ ใช้ งาน บ่อย บ่อย นะ ค่ะ

(a) CCTC can be too confident in a wrong prediction.

Argmax	CTC	LOOK TONNY THAT'S HIS POISON I SAID
	CCTC	LOOK TONY THAT'S HIS POISON I SAID
3-gram LM	CTC	LOOK TONY THAT'S HIS POISON I SAID
	CCTC	LOOK TONY THAT HIS POISON I SAID
Ground truth		LOOK TONY THAT'S HIS POISON I SAID

(b) A correct prediction can be corrupted by the LM.

FIGURE 6. Selected samples showcasing the possible mismatch between CCTC and LM rescoring. The words of interested are highlighted.

combination of two characters while Fig. 4b also shows a similar occurrence where the CCTC can help disambiguate hard-to-decipher handwriting. These errors might get corrected with a LM. However, we would like to emphasize again that CCTC yields this improvement without any extra computation cost during inference.

VI. DISCUSSIONS

A. CCTC LEARNS AN IMPLICIT LANGUAGE MODEL

Our experiments have shown that CCTC can help improve the performance of ASR and HTR systems in various settings. In this section, we present some supporting evidence showing that the model trained with CCTC also learns the LM in the process, thereby improving the model in sequence prediction tasks. To detect this effect, we computed the perplexity of the *prediction outputs* (test set) using the language model learned from the *training text*. If the model learns any sequence information in the training process, this perplexity should decrease.

We used 7-gram character LMs trained on the training sets to compute the perplexity. The choice of 7-gram is so that the context size will cover up to the context size of CCTC$^{(3)}$. The text used to train the LM was deduplicated.

Fig. 5 illustrates perplexities on argmax decoding predictions for each test set. The baseline CTC model generally has the highest perplexity, and the value tends to decrease as the context size increases. The lower perplexities of CCTC models indicate that the predictions of the CCTC models are more congruent to the LM than the baseline CTC, supporting the claim that CCTC can learn an implicit LM. Note that for Thai YouTube, the ASR models were trained on the entire training set but tested separately in two different testing subsets. The Thai YouTube dataset is mostly monolingual Thai, causing the implicit LM learned by the CCTC models to be more focused on Thai. Thus the perplexity in the code-switching test set can increase, which is the case for CCTC$^{(3)}$. BEST is also another dataset that does not exhibit the expected trend. A large portion of the training data for BEST is based on the same set of patterns, which are slightly different from the seen test set.

B. DISCREPANCY BETWEEN CONTEXT PREDICTIONS AND LM RESCORING

Even though increasing context sizes for CCTC models provide performance gains using argmax and beam search decoding, CCTC models with a shallower context window tend to be more suitable for external LM rescoring than the wider ones. From Table 3 and Table 5, CCTC$^{(1)}$ generally had the highest effectiveness when the external LM was used. However, as context size increased, the performance might drop, especially in the monolingual setup, in which CCTC models were sometimes inferior to the baseline CTC.

Fig. 6 depicts selected samples of the scenario in which the CCTC model provides a better prediction with argmax but underperforms the baseline with 3-gram LM. In Fig. 6a,

TABLE 8. The trade-off between the accuracy gains and runtime performances (perf). Gains are reported in relative improvement over the original CTC (%rel). Performances are reported as the ratio between training times using the CTC's runtime as the baseline.

Model	LibriSpeech		Thai YouTube		IAM		BEST	
	gain (%rel)	perf (x)	gain (%rel)	perf (x)	gain (%rel)	perf (x)	gain (%rel)	perf (x)
CTC	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0
CCTC$^{(1)}$	3.0	.80	1.4	.76	0.8	.88	5.8	.79
CCTC$^{(2)}$	3.3	.71	2.2	.69	3.1	.79	9.9	.78
CCTC$^{(3)}$	2.7	.65	1.5	.61	5.4	.72	7.8	.77
CCTC$^{(4)}$	-	-	-	-	5.5	.68	8.4	.75

the sample from dev-cs of Thai YouTube shows that LM rescoring cannot fix the bad prediction of the CCTC model, but it can fix CTC's prediction. Fig. 6b depicts an example from LibriSpeech dev-clean that the LM causes the error in the CCTC model.

Aggressive context dependencies from context heads may cause conflicts between the internal language representations and the external LM rescorer. Further investigation on methods that can learn contexts jointly with the external LM might reduce this discrepancy.

C. RELATIONSHIP BETWEEN PERFORMANCE GAIN AND TRAINING TIME

Since the inference time for CCTC is always the same as the regular base model, we discuss the trade-off between the gain in evaluation metrics and the increase in training time in this section. Table 8 summarizes the trade-offs between gains and runtimes in the argmax decoding setup. The gains are shown in relative improvements of WER and CER for ASR corpora and HTR corpora, respectively. The runtime performances were measured using the ratio between the CTC and CCTC training time (higher is faster) and were averaged over ten-batch training, excluding data loading steps.

As expected the training speed of CCTC models reduces as the context size increases. Considering the trade-off, a context size of two seems to offer most of the benefit. The increase in training time varies across datasets due to the differences in encoders, input lengths, and alphabet sizes. For datasets with more input frames (LibriSpeech), a large proportion of the computation is used to compute the gradients, lessening the effect of adding context heads on the computation cost. Note that, the label generation process is not optimized to work in GPU memory in our implementation. With proper implementation, the performance drops should be further reduced, just like in the CTC loss [31].

D. APPLYING ADAPTIVE WEIGHT ASSIGNMENTS FOR CONTEXT LOSSES

To reduce the efforts of manual weight searching, we attempted to use adaptive task balancing methods such as DTP [55] and DWA [56]. However, we found no improvement due to the distinctive role of context prediction subtasks in comparison to subtasks of other multi-task learning setups, which require subtasks to perform well independently on their respective metrics [57], [58].

TABLE 9. Ablation studies for the WER(%) of wav2letter CCTC$^{(1)}$ models on LibriSpeech. The g.t. stands for ground truth.

Setup	dev-clean			test-clean		
	argmax	beam	3-gram	argmax	beam	3-gram
Baseline CTC	22.06	21.95	**15.00**	21.97	21.87	15.11
CCTC$^{(1)}$						
from-scratch	22.88	22.80	15.72	22.77	22.67	15.63
pre-train	**21.27**	**21.23**	15.02	**21.32**	**21.20**	**15.06**
with g.t. labels	21.81	21.71	15.33	21.34	21.22	15.08

TABLE 10. The weights of context losses used in each experiment.

Dataset	Model	$\alpha^{(1)}$	$\alpha^{(2)}$	$\alpha^{(3)}$	$\alpha^{(4)}$
Librispeech 100 hr	CCTC$^{(1)}$	0.2			
	CCTC$^{(2)}$	0.25	0.1		
	CCTC$^{(3)}$	0.25	0.1	0.05	
Librispeech 960 hr	CCTC$^{(2)}$	2.0	1.0		
Thai YouTube	CCTC$^{(1)}$	0.15			
	CCTC$^{(2)}$	0.05	0.075		
	CCTC$^{(3)}$	0.025	0.05	0.1	
IAM	CCTC$^{(1)}$	0.1			
	CCTC$^{(2)}$	0.05	0.1		
	CCTC$^{(3)}$	0.02	0.05	0.1	
	CCTC$^{(4)}$	0.01	0.02	0.05	0.1
BEST	CCTC$^{(1)}$	1			
	CCTC$^{(2)}$	1	1		
	CCTC$^{(3)}$	1	1	1	
	CCTC$^{(4)}$	1	1	1	1

In CCTC frameworks, the performance of the main task was exclusively considered. The context prediction tasks of CCTC are sided tasks, which intend to complement the main CTC task and heavily depend on the main CTC task. The degradation of sided tasks is acceptable for gaining the performance of the main task. To the best of our knowledge, existing frameworks with similar setups also employ fixed weights tuning [59], [60].

E. CCTC TRAINING FROM RANDOM INITIALIZATION

We found that starting CCTC training from the first iteration was not effective (*from-scratch*), as shown in Table 9. We observed training instabilities and worse WER compared to both the regular CCTC model, which used pre-trained weights for initialization, and the baseline CTC model. Since CCTC labels were computed on-the-fly using current model, the low-quality predictions at the early stage hurt the overall performance.

We ensured that every setup had the same the number of model updates. For *from-scratch*, a randomly initialized model was trained by CTC and context losses for 400 epochs. The pre-train approach were trained for 300 epochs using CTC loss, followed by another 100 epochs of CCTC training.

F. CHOICE OF CONTEXT LABELS

Table 9 illustrates that using labels from forced alignments for context labels (*with g.t. labels*) incurred a 2.5% relative degradation in the WER of CCTC models compared to utilizing the labels obtained from argmax predictions (*pre-train CCTC*). We hypothesized that the mismatches between the alignments and the CCTC model predictions were responsible for the performance degradation. Note that we computed the forced alignments from the wav2letter CTC model using CTC-Segmentation described in [61].

VII. LIMITATIONS

There are some considerations for applying CCTC. First, determining the optimal loss weights and context sizes requires hyperparameter tuning for the context weights. This can be mitigated by using search space reduction methods. Second, although the inference time remains the same, CCTC increases the training time. Finally, the impact of CCTC is less pronounced when used in conjunction with a language model. To address this limitation in future work, we plan to investigate training CCTC together with a language model.

VIII. CONCLUSION

CCTC is a framework that can incorporate context information into the CTC loss for training non-recurrent NAR models. Experiments on ASR and HTR benchmarks on Thai and English datasets have shown that CCTC can help improve the performance by learning an implicit character language model. CCTC models with a wider context are generally more superior up to a certain size but can have difficulties when used in conjunction with an external LM. In the future, we plan to investigate joint training with the language model in order to fully utilize the implicit LM learned by CCTC.

APPENDIX

HYPERPARAMETERS

Table 10 depicts the context loss weights assigned to each model. The weights vary due to the differences in the magnitudes of CTC losses. For the Librispeech 100 hr, Thai YouTube, and IAM models, the CTC losses were normalized by the length of transcriptions. In contrast, the remaining models used unnormalized CTC losses.

ACKNOWLEDGMENT

The authors would like to thank CMKL University (Apex cluster) and Chulalongkorn University Technology Center for supporting the computational resources.

REFERENCES

[1] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. Interspeech*, Oct. 2020, pp. 1–5.

[2] R. J. Weiss, R. Skerry-Ryan, E. Battenberg, S. Mariooryad, and D. P. Kingma, "Wave-tacotron: spectrogram-free end-to-end text-to-speech synthesis," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 5679–5683.

[3] V. Tassopoulou, G. Retsinas, and P. Maragos, "Enhancing handwritten text recognition with n-gram sequence decomposition and multitask learning," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 10555–10560.

[4] F. Wu, A. Fan, A. Baevski, Y. Dauphin, and M. Auli, "Pay less attention with lightweight and dynamic convolutions," in *Proc. ICLR*, 2019, pp. 1–14.

[5] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, and S. Agarwal, "Language models are few-shot learners," in *Proc. NIPS*, 2020, pp. 1877–1901.

[6] B. Naowarat, T. Kongthaworn, K. Karunratanakul, S. H. Wu, and E. Chuangsuwanich, "Reducing spelling inconsistencies in code-switching ASR using contextualized CTC loss," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 6239–6243.

[7] Z. Qiu, Y. Li, X. Li, F. Metze, and W. M. Campbell, "Towards context-aware end-to-end code-switching speech recognition," in *Proc. Interspeech*, Oct. 2020, pp. 4776–4780.

[8] J. Baek, G. Kim, J. Lee, S. Park, D. Han, S. Yun, S. J. Oh, and H. Lee, "What is wrong with scene text recognition model comparisons? Dataset and model analysis," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4714–4722.

[9] Y. Higuchi, H. Inaguma, S. Watanabe, T. Ogawa, and T. Kobayashi, "Improved mask-CTC for non-autoregressive end-to-end ASR," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 8363–8367.

[10] J. Gu and X. Kong, "Fully non-autoregressive neural machine translation: Tricks of the trade," in *Proc. Findings Assoc. Comput. Linguistics, ACL-IJCNLP*, 2021, pp. 1–12.

[11] J. Gu, J. Bradbury, C. Xiong, V. O. Li, and R. Socher, "Non-autoregressive neural machine translation," in *Proc. ICLR*, 2018, pp. 1–13.

[12] Z. Li, Z. Lin, D. He, F. Tian, T. Qin, L. Wang, and T.-Y. Liu, "Hint-based training for non-autoregressive machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 1–9.

[13] Y. Wang, F. Tian, D. He, T. Qin, C. Zhai, and T.-Y. Liu, "Non-autoregressive machine translation with auxiliary regularization," in *Proc. AAAI*, 2019, pp. 5377–5384.

[14] Z. Sun, Z. Li, H. Wang, Z. Lin, D. He, and Z.-H. Deng, "Fast structured decoding for sequence models," in *Proc. NIPS*, 2019, pp. 1–11.

[15] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 369–376.

[16] A. Graves, S. Fernandez, M. Liwicki, H. Bunke, and J. Schmidhuber, "Unconstrained online handwriting recognition with recurrent neural networks," in *Proc. NIPS*, 2008, pp. 1–8.

[17] Y. Higuchi, S. Watanabe, N. Chen, T. Ogawa, and T. Kobayashi, "Mask CTC: Non-autoregressive end-to-end ASR with CTC and mask predict," in *Proc. Interspeech*, Oct. 2020, pp. 1–5.

[18] S. Kriman, S. Beliaev, B. Ginsburg, J. Huang, O. Kuchaiev, V. Lavrukhin, R. Leary, J. Li, and Y. Zhang, "QuartzNet: Deep automatic speech recognition with 1D time-channel separable convolutions," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 6124–6128.

[19] S.-P. Chuang, H.-J. Chang, S.-F. Huang, and H.-Y. Lee, "Non-autoregressive Mandarin-English code-switching speech recognition with pinyin mask-CTC and word embedding regularization," in *Proc. ASRU*, 2021, pp. 465–472.

[20] D. Coquenet, C. Chatelain, and T. Paquet, "Recurrence-free unconstrained handwritten text recognition using gated fully convolutional network," in *Proc. 17th Int. Conf. Frontiers Handwriting Recognit. (ICFHR)*, Sep. 2020, pp. 19–24.

[21] M. Yousef, K. F. Hussain, and U. S. Mohammed, "Accurate, data-efficient, unconstrained text recognition with convolutional neural networks," *Pattern Recognit.*, vol. 108, Dec. 2020, Art. no. 107482.

[22] J. Michael, R. Labahn, T. Gruning, and J. Zöllner, "Evaluating sequence-to-sequence models for handwritten text recognition," in *Proc. Int. Conf. Document Anal. Recognit. (ICDAR)*, Sep. 2019, pp. 1286–1293.

[23] D. Yu, X. Li, C. Zhang, T. Liu, J. Han, J. Liu, and E. Ding, "Towards accurate scene text recognition with semantic reasoning networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 12110–12119.

[24] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 5206–5210.

[25] *BEST 2019 (Handwritten Recognition)*, W Sinthupinyo, Nat. Electron. Comput. Technol. Center (NECTEC), Pathum Thani, Thailand, 2018.

[26] *BEST 2020 (Handwritten Recognition)*, W Sinthupinyo, Nat. Electron. Comput. Technol. Center (NECTEC), Pathum Thani, Thailand, 2020.

[27] U.-V. Marti and H. Bunke, "The IAM-database: An English sentence database for offline handwriting recognition," *Int. J. Document Anal. Recognit.*, vol. 5, no. 1, pp. 39–46, Nov. 2002.

[28] D. Jurafsky and J. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Upper Saddle River, NJ, USA: Prentice-Hall, 2009.

[29] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.

[30] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Proc. NIPS*, 2015, pp. 1–9.

[31] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, and G. Chen, "Deep speech 2: End-to-end speech recognition in English and Mandarin," in *Proc. ICML*, 2016, pp. 173–182.

[32] W. Chan, C. Saharia, G. E. Hinton, M. Norouzi, and N. Jaitly, "Imputer: Sequence modelling via imputation and dynamic programming," in *Proc. ICML*, 2020, pp. 1403–1413.

[33] J. Hu, M. K. Brown, and W. Turin, "HMM based online handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 10, pp. 1039–1045, Oct. 1996.

[34] Y. Bengio, Y. LeCun, C. Nohl, and C. Burges, "LeRec: A NN/HMM hybrid for on-line handwriting recognition," *Neural Comput.*, vol. 7, no. 6, pp. 1289–1303, Nov. 1995.

[35] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 855–868, May 2009.

[36] P. Dreuw, D. Rybach, C. Gollan, and H. Ney, "Writer adaptive training and writing variant model refinement for offline Arabic handwriting recognition," in *Proc. 10th Int. Conf. Document Anal. Recognit.*, 2009, pp. 21–25.

[37] P. Voigtlaender, P. Doetsch, and H. Ney, "Handwriting recognition with large multidimensional long short-term memory recurrent neural networks," in *Proc. 15th Int. Conf. Frontiers Handwriting Recognit. (ICFHR)*, Oct. 2016, pp. 228–233.

[38] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Proc. NIPS*, 2009, pp. 1–8.

[39] J. Puigcerver, "Are multidimensional recurrent layers really necessary for handwritten text recognition?" in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, Nov. 2017, pp. 67–72.

[40] V. Carbune, P. Gonnet, T. Deselaers, H. A. Rowley, A. Daryin, M. Calvo, L.-L. Wang, D. Keysers, S. Feuz, and P. Gervais, "Fast multi-language LSTM-based online handwriting recognition," *Int. J. Document Anal. Recognit. (IJDAR)*, vol. 23, no. 2, pp. 89–102, Jun. 2020.

[41] A. Sharma and D. B. Jayagopi, "Towards efficient unconstrained handwriting recognition using dilated temporal convolution network," *Exp. Syst. Appl.*, vol. 164, Feb. 2021, Art. no. 114004.

[42] T. Zenkel, R. Sanabria, F. Metze, and A. Waibel, "Subword and crossword units for CTC acoustic models," in *Proc. Interspeech*, Sep. 2018, pp. 1–5.

[43] G. Synnaeve, Q. Xu, J. Kahn, T. Likhomanenko, E. Grave, V. Pratap, A. Sriram, V. Liptchinsky, and R. Collobert, "End-to-end ASR: From supervised to semi-supervised learning with modern architectures," 2019, *arXiv:1911.08460*.

[44] J. Chorowski, A. Lancucki, B. Kostka, and M. Zapotoczny, "Towards using context-dependent symbols in CTC without state-tying decision trees," in *Proc. Interspeech*, Sep. 2019, pp. 1–5.

[45] R. Sanabria and F. Metze, "Hierarchical multitask learning with CTC," in *Proc. IEEE Spoken Lang. Technol. Workshop (SLT)*, Dec. 2018, pp. 485–490.

[46] H. Liu, Z. Zhu, X. Li, and S. Satheesh, "Gram-CTC: Automatic unit selection and target decomposition for sequence labelling," in *Proc. ICML*, 2017, pp. 2188–2197.

[47] A. Graves, "Sequence transduction with recurrent neural networks," 2012, *arXiv:1211.3711*.

[48] Y. Zhang, D. Yu, M. L. Seltzer, and J. Droppo, "Speech recognition with prediction-adaptation-correction recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 5004–5008.

[49] Y. Zhang, E. Chuangsuwanich, J. Glass, and D. Yu, "Prediction-adaptation-correction recurrent neural networks for low-resource language speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 5415–5419.

[50] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015, pp. 1–15.

[51] O. Kuchaiev, J. Li, H. Nguyen, O. Hrinchuk, R. Leary, B. Ginsburg, S. Kriman, S. Beliaev, V. Lavrukhin, J. Cook, P. Castonguay, M. Popova, J. Huang, and J. M. Cohen, "NeMo: A toolkit for building AI applications using neural modules," 2019, *arXiv:1909.09577*.

[52] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1800–1807.

[53] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, and A. Desmaison, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. NIPS*, 2019, pp. 1–12.

[54] L. Gillick and S. J. Cox, "Some statistical issues in the comparison of speech recognition algorithms," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, 1989.

[55] M. Guo, A. Haque, D.-A. Huang, S. Yeung, and L. Fei-Fei, "Dynamic task prioritization for multitask learning," in *Proc. ECCV*, 2018, pp. 270–287.

[56] S. Liu, E. Johns, and A. J. Davison, "End-to-end multi-task learning with attention," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1871–1880.

[57] S. Vandenhende, S. Georgoulis, W. Van Gansbeke, M. Proesmans, D. Dai, and L. Van Gool, "Multi-task learning for dense prediction tasks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3614–3633, Jul. 2022.

[58] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," in *Proc. NIPS*, 2018, pp. 1–12.

[59] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid CTC/attention architecture for end-to-end speech recognition," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 8, pp. 1240–1253, Dec. 2017.

[60] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for ASR based on lattice-free MMI," in *Proc. Interspeech*, Sep. 2016, pp. 2751–2755.

[61] L. Kurzinger, D. Winkelbauer, L. Li, T. Watzel, and G. Rigoll, "CTC-segmentation of large corpora for German end-to-end speech recognition," in *Proc. SPECOM*, 2020, pp. 267–278.

BURIN NAOWARAT received the B.S. degree from Chulalongkorn University, Bangkok, Thailand, in 2019, where he is currently pursuing the M.S. degree in computer engineering with the Department of Computer Engineering, Chulalongkorn University. His research interests include speech processing, representation learning, language understanding, and self-supervised learning.

CHAWAN PIANSADDHAYANON received the B.S. degree in computer engineering from Chulalongkorn University, in 2020. He is currently pursuing the M.S. degree in computer engineering. His research interests include computer vision, deep learning, and medical imaging.

EKAPOL CHUANGSUWANICH received the B.S. and M.S. degrees in electrical and computer engineering from Carnegie Mellon University, in 2008 and 2009, respectively, and the Ph.D. degree from MIT, in 2016. Then, he joined the MIT Computer Science and Artificial Intelligence Laboratory and Spoken Language Systems Group. He is currently a Faculty Member of the Department of Computer Engineering, Chulalongkorn University. His research interests include speech processing, assistive technology, and health applications.