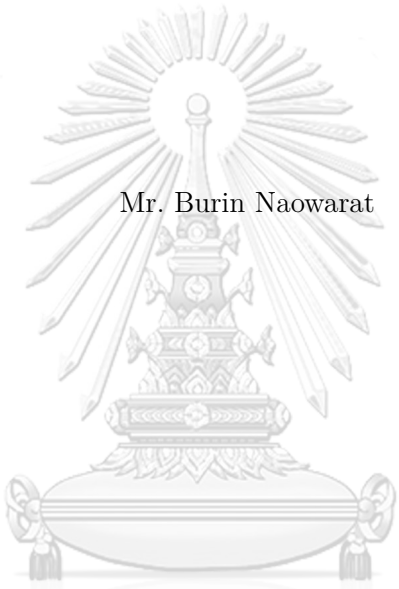
INCORPORATING CONTEXT INTO NON-AUTOREGRESSIVE MODEL USING CONTEXTUALIZED CTC FOR SEQUENCE LABELLING

Mr. Burin Naowarat



A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering in Computer Engineering

Department of Computer Engineering

FACULTY OF ENGINEERING

Chulalongkorn University

Academic Year 2021

การผสานบริบทเข้ากับนอนออโตรีเกรสซีฟโมเดลด้วยซีทีซีที่สามารถเรียนรู้บริบทสำหรับการติดป้ายตามลำดับ

นายบุรินทร์ เนาวรัตน์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์มหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2564

Thesis Title	INCORPORATING CONTEXT INTO NON-AUTOREGRESSIVE MODEL USING CONTEXTUALIZED CTC FOR SEQUENCE LABELLING
By	Mr. Burin Naowarat
Field of Study	Computer Engineering
Thesis Advisor	Assoc. Prof. Atiwong Suchato, Ph.D.
Thesis Co-advisor	Ekapol Chuangsuwanich, Ph.D.

Accepted by the FACULTY OF ENGINEERING, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master of Engineering

..... Dean of the FACULTY OF ENGINEERING

(Prof. Supot Teachavorasinskun, D.Eng.)

THESIS PROPOSAL COMMITTEE

..... Chairman

(Assoc. Prof. Proadpran Punyabukkana, Ph.D.)

..... Thesis Advisor

(Assoc. Prof. Atiwong Suchato, Ph.D.)

..... Thesis Co-advisor

(Ekapol Chuangsuwanich, Ph.D.)

..... External Examiner

(Sanparith Marukatat, Ph.D.)

บุรินทร์ เนาวรัตน์: การผสานบริบทเข้ากับนอนออโตรีเกรสซีฟโมเดลด้วยซีทีซีที่สามารถเรียนรู้บริบทสำหรับการติดป้ายตามลำดับ. (INCORPORATING CONTEXT INTO NON-AUTOREGRESSIVE MODEL USING CONTEXTUALIZED CTC FOR SEQUENCE LABELLING) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : รศ.ดร.อติวงศ์ สุชาโต, อ.ที่ปรึกษาวิทยานิพนธ์ร่วม : ดร. เอกพล ช่วงสุวนิช 71 หน้า.

เนื่องจากความง่ายในการใช้งาน คอนเนคชันนิสเทมโปรอลคลาสสิฟิเคชัน (ซีทีซี) จึงถูกนำมาใช้อย่างแพร่หลายในปัญหาการจำลองตามลำดับอาทิเช่นการรู้จำเสียงพูดอัตโนมัติและการรู้จำตัวอักษรลายมือเขียน ซีทีซีนั้นสามารถใช้เพื่อฝึกฝนโมเดลโครงข่ายประสาทเทียมแบบใดก็ได้ แต่มักจะถูกใช้คู่กับโมเดลโครงข่ายประสาทเทียมแบบเกิดซ้ำที่คำนึงถึงผลลัพธ์ในอดีตในการทำนายผลลัพธ์ในปัจจุบันเพื่อผ่อนคลายสมมติฐานของความน่าจะเป็นแบบเป็นอิสระของซีทีซี อย่างไรก็ตามงานวิจัยในช่วงหลังสนใจการใช้งานซีทีซีคู่กับโมเดลแบบไม่เกิดซ้ำโดยมีวัตถุประสงค์ที่จะลดประสิทธิภาพที่เกิดจากความสามารถในจากการพึ่งพาบริบทเพื่อเพิ่มความเร็วในการทำนายผล วิทยานิพนธ์นี้ได้เสนอคอนเทคชัวไลซ์คอนเนคชันนิสเทมโปรอลคลาสสิฟิเคชัน (ซีซีทีซี) สำหรับฝึกฝนโมเดลแบบไม่เกิดซ้ำที่ใช้ในปัญหาการจำลองตามลำดับ ซีซีทีซีใช้ประโยชน์จากการเรียนรู้หลากหลายงานพร้อมกันในการทำให้โมเดลแบบไม่เกิดซ้ำมีโอกาสที่จะเรียนรู้บริบทสำหรับใช้ในการทำนายผลผ่านการทำนายผลลัพธ์ที่อยู่รอบข้างและการทำนายผลลัพธ์หลักไปพร้อมกัน ผลการทดลองในการรู้จำเสียงพูดอัติโนมัติและการรู้จำตัวอักษรลายมือเขียนสำหรับภาษาไทยและอังกฤษแสดงให้เห็นว่าซีซีทีซีมีประสิทธิภาพสัมพัทธ์สูงกว่าซีทีซี 2.2-8.4% โดยที่ยังสามารถคงความเร็วในการทำนายผลไว้ได้เท่าซีทีซีแบบดั้งเดิม

ภาควิชา	วิศวกรรมคอมพิวเตอร์	ลายมือชื่อนิสิต	.....
สาขาวิชา	วิศวกรรมคอมพิวเตอร์	ลายมือชื่ออ.ที่ปรึกษาหลัก	.....
ปีการศึกษา	2564	ลายมือชื่อ.ที่ปรึกษาร่วม	

6270145221: MAJOR COMPUTER ENGINEERING
KEYWORDS: NON-AUTOREGRESSIVE, SEQUENCE MODELING, CONTEXTUAL

BURIN NAOWARAT : INCORPORATING CONTEXT INTO NON-AUTOREGRESSIVE MODEL USING CONTEXTUALIZED CTC FOR SEQUENCE LABELLING. ADVISOR : Assoc. Prof. Atiwong Suchato, Ph.D., THESIS COADVISOR : Ekapol Chuangsuwanich, Ph.D., 71 pp.

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. CTC itself has no architecture constraints, but it is commonly used with recurrent models that predict letters based on histories in order to relax the conditional independent assumption. However, recent sequence models that incorporate CTC loss have been focusing on speed by removing recurrent structures, hence losing important context information. This thesis presents Contextualized Connectionist Temporal Classification (CCTC) loss, 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.

Department	:	Computer Engineering	Student's Signature	.....
Field of Study	:	Computer Engineering	Advisor's Signature	.....
Academic Year	:	2021	Co-advisor's signature	.....

Acknowledgements

I would like to thank my supervisor Ekapol Chuangsuwanich for his support and guidance throughout the years as a student at spoken language system laboratory. I am sincerely grateful for his forbearance in walking me through my confusion, his patience with my poorly written drafts, and his assistance in finding external funds.

Many thanks to my close colleagues, Thananchai Kongthaworn, Artit Suwanbandit, and Pattaraporn Pongpanatapipat, for tons of their helps. Thank you to Konpat Preechakul, Chawan Piansaddhayanon, and Korrawe Karunratanakul for collaborations, great motivations, insightful discussions, and valuable perspectives. Thanks to Terapap Apiparakoon, Krit Gangwanpongpun, and everyone else at the lab for bringing pleasure and warmth to the place.

Lastly, I would like to thank CMKL University and Chulalongkorn University Technology Center for supporting the computational resources. Without these powerful workstations, it would be impossible to finish many of my experiments.

CONTENTS

Page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Chapter I

INTRODUCTION

Context has been extensively proved to be useful for various kinds of sequence modeling problems, such as Automatic Speech Recognition (ASR) (Gulati et al., 2020), Text-to-Speech (TTS) (Weiss et al., 2021), Handwritten Text Recognition (HTR) (Tassopoulou et al., 2021), Neural Machine Translation (NMT) (Wu et al., 2019), and Language Modeling (Brown et al., 2020). 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 (Qiu et al., 2020; Baek et al., 2019). 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 (Higuchi et al., 2021; Gu and Kong, 2020). 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 AR models (Gu et al., 2018; Li et al., 2019b) or trained by context-dependent objective functions (Wang et al., 2019; Sun et al., 2019). 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 (Graves et al., 2006, 2008a). CTC estimates the probability for the alignment between frame-level predictions and character-level ground 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 (Higuchi et al., 2020). 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 (Higuchi et al., 2021; Kriman et al., 2020; Chuang et al., 2021).

Teaching the model to predict future labels has led to improvements in the main prediction of monolingual hybrid ASR systems (Jaitly et al., 2014; Zhang et al., 2015, 2016). In Code-Switching ASR, in which multiple languages are used in a single conversation, the dedicated language identification subtask is commonly trained with the main ASR task in a multi-task manner (Li et al., 2019a; Luo et al., 2018; Zeng et al., 2019; Shan et al., 2019). This is done because the lack of context severely deteriorates the quality of the predictions as a spoken word can be spelled using alphabets of any languages. We hypothesize that providing the required contextual information to the model in these manner should also alleviate the misspelling issues in non-recurrent NAR CTC models without sacrificing speed. Nevertheless, predicting contexts in the CTC framework is not straightforward since there is no explicit frame-level alignment for computing the Cross-Entropy (CE) loss. Besides, outputs of the CTC model are mostly blanks, providing little contextual information.

In this work, we introduce Contextualized CTC (CCTC) loss, a variant of context-dependent objective functions, to incorporate contexts into non-recurrent and non-autoregressive (NAR) ASR models. CCTC tries to increase dependencies between predictions by relaxing the conditional independence assumption of the regular CTC loss in a way that preserves parallel decoding capability. Concretely, CCTC uses multi-task learning to encourage the model to predict left and right letters as well as the middle letter. This is done by adding secondary prediction heads to predict the surrounding characters in a multi-task manner. The novelty of our approach lies on how we obtain the labels for the surrounding characters. The target prediction for the surrounding characters is obtained by the prediction from the previous iteration which replaces the need for frame-based alignments from external models. To get the prediction target of the context we also ignore blanks and consecutive duplicate letters. This helps provide longer contextual information than predicting actual symbols of surrounding frames, which has shown to be more effective in (Zhang et al., 2015).

Experiments on two sequence modeling, ASR and HTR, in two different languages show that CCTC models generally outperform the baseline CTC models, especially when no external language models are applied. CCTC can mitigate intra-word language inconsistency for CS ASR and reduce misspellings for both ASR and HTR. 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.

1.1 Main contribution

The main contribution of this thesis can be summarized as follows:

- We introduce the novel CCTC loss for training NAR sequence models.
- We show the performances of CCTC on monolingual and code-switching ASR.
- We present the effectiveness of CCTC loss on HTR.

- We investigate the dependencies within predictions of different models using character-level perplexities.
- We study the trade-off between prediction quality and training time for contexts of different sizes.

1.2 Thesis overview

This thesis is organized as follows:

- Chapter 2 describes the fundamental backgrounds of sequence modeling and algorithms used for addressing the problem.
- Chapter 3 presents related works of using contexts for sequence modeling and incorporating context dependencies for CTC models.
- Chapter 4 introduces contextualized Connectionist Temporal Classification.
- Chapter 5 provides experiments verifying the effectiveness of CCTC loss.
- Chapter 6 discusses the characteristics of CCTC models.
- Chapter 7 summarizes the thesis and provides open directions for future works.

Chapter II

BACKGROUND

This chapter presents the brief definition of sequence modeling, its applications, and the existing algorithms being used to address the problems.

2.1 Sequence Modeling

The goal of sequence modeling is to create an algorithm that correctly produces an output sequence from an input sequence, given that a sequence is a series of strictly ordered objects. The relation between properties of inputs and outputs entirely depends on the downstream applications, having no specific constraints for general cases. Besides, their lengths can also be different, ranging from the shortest of one to any arbitrary finite number. For example, sequence modeling can be used for statistical machine translation and audio enhancement, of which both inputs and outputs share the same modalities (text or audio). Sequence modeling can also be used to transform inputs into outputs that have different modality such as transcribing spoken audio into transcription (automatic speech recognition), recognizing letters from a sequence of handwritten image pixels (handwritten text recognition), classifying a person's action from a sequence of video frames (action recognition), and many more. In this thesis, we will focus on two applications: automatic speech recognition (ASR) and handwriting recognition (HTR).

Since properties of ASR and HTR have a lot in common, we can mathematically define both problems as one. Formally, we have an input sequence of M dimensional real-valued vectors, $x = (x_1, x_2, ..., x_T) : x_t \in \mathbb{R}^M$, and a desired output sequence, $y^* = (y_1^*, y_2^*, ..., y_U^*) : y_u^* \in \mathbf{A}$, where $\mathbf{A}$ is a set containing alphabets which in this thesis are Thai and/or English alphabets. Let's the dataset $\mathcal{D}$ contains all possible pairs of input and desired output, (x, y^*). The goal is to learn the mapping $h : x \rightarrow y$ that yields the minimum errors between the predictions and the ground truths, $\mathbf{E}(y, y^*)$, of data in the test set, $\mathcal{D}_{test} \subset \mathcal{D}$. The mapping h is trained by minimizing an objective function

using supervision from the training set, $\mathcal{D}_{tr} \subset \mathcal{D} : \mathcal{D}_{tr} \cap \mathcal{D}_{test} = \emptyset$.

2.2 Connectionist Temporal Classification

CTC (Graves et al., 2006) is an alignment-free objective function for sequence modeling that the lengths of inputs, T, are less than or equal to the lengths of outputs, U. The general mechanism of CTC loss is described as follows. Suppose we have an input sequence $x = (x_1, x_2, ..., x_T) : x_t$, CTC loss maximizes the probability of predicting the ground truth transcription, $y^* = (y_1^*, y_2^*, ..., y_U^*) : y_u^* \in \mathbf{A}$. In addition, CTC framework includes a special token of blank, ϵ, in the alphabets set, $\mathbf{A}' = \mathbf{A} \cup \{\epsilon\}$, to handle noise, empty spaces, and consecutive duplicate characters in transcriptions. CTC model outputs a path, $\pi = (\pi_1, \pi_2, ..., \pi_T) : \pi_t \in \mathbf{A}'$, which has the same length as the input frames. Lastly, π is mapped to an inferred transcription, $y = (y_1, y_2, ..., y_K) : y_k \in \mathbf{A}$, using a mapping function $\mathcal{B} : \mathbf{A}' \rightarrow \mathbf{A}$. By applying the function $\mathcal{B}$, adjacent duplicate alphabets are merged, and blank tokens are removed. Remark that, adjacent duplicate alphabets are considered as the prediction of a single long pronunciation, which occupies many consecutive frames. In the case of double letters such as *ff* in *coffee*, CTC distinguishes this scenario from the previous by producing blank tokens in between.

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 (2.1). We depict the calculation of the CTC loss in (2.2).

$$P(y^*|x) = \sum_{\pi \in \mathcal{B}^{-1}(y^*)} P(\pi|x) = \sum_{\pi \in \mathcal{B}^{-1}(y^*)} \prod_t P(\pi_t|x) \tag{2.1}$$

$$\mathcal{L}_{CTC} = -\log P(y^*|x) \tag{2.2}$$

2.3 Automatic Speech Recognition

Automatic Speech Recognition (ASR) is named after the frameworks that automatically transcribe an acoustic signal of spoken audio into a textual representation. Nowadays, mainstream ASR frameworks can be classified into two categories: traditional models and end-to-end models.

2.3.1 Traditional ASR

A traditional ASR framework comprises three separate modules: acoustic model (AM), lexicon model, and language model (LM). For an overview, an AM recognizes pronunciations from acoustic properties of an incoming audio. A lexicon model composes pieces of recognized pronunciations into words. Lastly, a LM ranks the likelihoods of word candidates based on co-occurrence probabilities.

For a sequence of acoustic features x and the target transcription y^*, a traditional ASR model try to maximize the posterior of generating a transcription given a corresponding audio, $P(y^*|x)$. However, in the era of being introduced, discriminative models still had limitations, hampering direct posterior approximation. A traditional ASR was made possible using Bayes's Rule which allows maximizing posterior through likelihood, as shown in (2.3).

$$\begin{aligned} P(y^*|x) &= \frac{P(x|y^*)P(y^*)}{P(x)} \\ &\approx P(x|y^*)P(y^*) \end{aligned} \tag{2.3}$$

An acoustic model's likelihood, $P(x|y^*)$, is modeled using Hidden Markov Model (HMM), of which the emission probabilities are estimated using Gaussian Mixture Model (GMM). The prior, $P(y^*)$, is retrieved from a LM which usually is n-gram. The evidence, $P(x)$, is dropped as it is a constant for a given input x, making no distinction for maximum likelihood estimation (MLE) training. Since the likelihood, $P(x|y^*)$, is modeled by HMM which follows Markov chain properties, we can write (2.3) as the

total probability of happening x based on all possible HMM's states as shown in (2.4).

$$\begin{aligned} P(y^*|x) &= \sum_s P(x|s)P(s|y^*)P(y^*) \\ &= \sum_s [\prod_t P(x_t|s_t)P(s_t|s_{t-1})]P(s|y^*)P(y^*) \end{aligned} \tag{2.4}$$

where s is a sequence of HMM states, representing the conditions for the probability of each pronunciation in each specific time. The states depend on the pronunciation units of the ground truth transcription from the lexicon model and the acoustic properties of the input audio.

One of the limitations of traditional ASR is that, at the training time, the *alignment*, has to be known. The *alignment* is the mapping between the timing of transcription and the timing of acoustic signal. In other words, an *alignment* specifies specific time in the acoustic signal that has the pronunciation of the corresponding word in the transcription. Since the likelihood, $P(x_t|s_t)$, is modeled individually for each time t, we must know the exact ground truth at time t to enable MLE training. This limitation is addressed by using expectation maximization (EM). EM comprises two steps, estimating the *alignment* based on the current model parameters and optimizing model parameters using MLE regarding the estimated *alignment*. EM switches back and forth between its two steps for many iterations until the model converges.

In summary, a traditional ASR model infers a transcription for a given audio based on the probability that is estimated by GMM-HMM and LM models.

$$y = \text{argmax}_{\hat{y}} \sum_s P(x|s)P(s|\hat{y})P(\hat{y}) \tag{2.5}$$

where $\hat{y}$ is transcription candidates. For interested readers, more details can be found in the tutorial of (Rabiner, 1989).

2.3.2 End-to-End ASR

End-to-End ASR models directly produce characters from input audio without using intermediate pronunciation representations as in traditional ASR systems. The end-to-end approaches have been made possible using powerful deep neural networks.

The burdens on linguistic experts have been reduced because pronunciation units are no longer required. Moreover, an end-to-end model can be trained without the need for alignments, eliminating the complex training strategy of EM.

The exact equation for posterior computation depends on the architecture designs, of which three choices are in the mainstream, including CTC models (Graves and Jaitly, 2014), RNN transducers (RNN-T) (Graves, 2012), and sequence-to-sequence (Seq2Seq) models (Bahdanau et al., 2014). In this thesis, we will focus on CTC-based models since it is the only approach that allows non-autoregressive predicting.

CTC-based models are trained with CTC loss as presented in 2.2. The input sequence, x, can be either raw-valued audios or hand-crafted features. The output transcription, y, can be phonemes, characters, or subwords (Zenkel et al., 2017). For decoding, CTC models find the sequence with the highest total probability using argmax decoding or prefix searching with LM as follows.

$$y = \text{argmax}_{\hat{y}} P_{AM}(\mathcal{B}^{-1}(\hat{y})|x) P_{LM}(\hat{y}) \tag{2.6}$$

where P_{AM} denotes the probability estimated by an AM, P_{LM} indicates the probability modeled by a LM.

2.3.3 Language Modeling

LM provides the probabilities for the next tokens appearing after the previous tokens. These statistics can indirectly represent the grammatical and/or semantics properties of the tokens, which commonly are characters or words. LMs used in ASR usually are n-gram and neural language models.

N-gram language model is simply a counting of co-occurrence probabilities that tell how likely that the next unit will happen right after the previous N-1 units. N-gram provides probabilities for units in a finite set of vocabulary, $\mathcal{V}$, as shown below:

$$P(y_u|y_{u-1}, y_{u-2}, ..., y_{u-\text{N}-1}) \mid y_u \in \mathcal{V} \tag{2.7}$$

We primarily use n-gram models since they are simple, efficient and provide good performance.

Neural language models provide the co-occurrence probabilities as same as N-gram models. The difference is that the probability is now predicted by a recurrent neural network (RNN) instead of direct counting. Theoretically, RNN is capable of modeling infinite lengths of previous words as in (2.8). Nonetheless, there are numerical stability issues when dealing with very long dependencies.

$$P(y_u|y_{u-1}, y_{u-2}, ..., y_1) \mid y_u \in \mathcal{V} \tag{2.8}$$

Neural language models are commonly superior to n-gram models, but they require more memory and computational resources both in training and inference phases.

2.4 Handwritten Text Recognition

Handwritten Text Recognition (HTR) has the goal of identifying the sequence of alphabets in the images of handwritten scripts. There are two subcategories for HTR, offline and online. Specifically, an online HTR transcribes a sequence of pen stroke positions into text. An offline HTR directly recognizes text from a raw image, having no supervision of pen stroke. A sequence of pen stroke position helps the model to learn meaningful relation between pixels in the input image. The lack of pen stroke supervision for offline HTR enforces the model to learn relationship between raw pixels itself. Therefore, offline HTR, which is the focus of this thesis, is considered as a harder problem. As the handwriting letters are sequentially written letter by letter, it is obvious that online HTR is a sequence modeling problem. As for offline HTR, the lesser obvious, a sequence of text has to be generated from a sequence of features extracted from a 2-dimensional image.

The approaches for HTR have a lot in common to ASR. The HMM-based models are used at the very first (Margner and El Abed, 2007; Dreuw et al., 2009), followed by end-to-end CTC models (Graves et al., 2008a,b). The main differences between HTR and ASR are the feature extraction due to the distinct between images and audios. This

distinction affects the design choices of the models.

2.5 Deep Neural Network

This section provides a very brief introduction to neural network (NN) with the goal of making readers understand our work. We will present the basic of three NN variants: Fully-connected neural network (FNN), Recurrent Neural Networks (RNN), and Convolutional Neural Network (CNN). We advise readers who are interested in the topic of NN finding more details in (Goodfellow et al., 2016).

2.5.1 Generic properties

NN originally have been proposed as a mathematical model of neurons in the nervous system. We represents neuron k in the network using a mathematical operation $\mathcal{F}$ with weights W_k and biases b_k. Neuron k will operate on a given input vector, x, and return a scalar number, $y_k = \mathcal{F}(x; W_k, b_k)$, as an output. Outputs of every neurons are concatenated altogether into a single output vector, $Y = [y_1, y_2, ..., y_K]$. The operation $\mathcal{F}$ is simply defined as follows.

$$\mathcal{F}(x; W_k, b_k) = \theta(W_k x + b_k) \tag{2.9}$$

where θ is an activation function. The weight, W_k, and the bias ,b_k, reflect a characteristics of the neuron k.

We use the name *layer* to indicate a group of neurons that take the same inputs. As depicted in Figure 2.1, the first layer of NN takes raw input vectors, x, and produces *hidden* representation vectors, which are used as inputs for the second layer. The *hidden* vector is passed through $H - 1$ layers until it becomes the final outputs, $y = \mathcal{F}^H(x)$. If there are many layers, a NN will be called as a Deep Neural Network (DNN) instead.

Activation Function

Activation layers are usually placed in between of two consecutive NN layers. Though the choice of an activation function can be any, people usually opt for non-linearity functions as they enable complex function approximation for NN models. From

the equation of NN (2.9), we can interpret NN as a stack of matrix multiplications. Since the chain of matrix multiplications can be transformed into a single linear transformation matrix, NN can only approximate linear functions if the activation functions are also linear. Non-linearity activation functions, that are placed between matrix multiplication operations, will disallow NN from being represented by a single linear transformation matrix. Subsequently, a combination between linear transformation (matrix multiplication) and non-linearity transformation (activation function) should allow a NN to approximate any complex function (Goodfellow et al., 2016). In this work, two non-linearity functions involve: ReLU and Softmax.

Rectified Linear Unit (ReLU) is a gate that disallows negative values, where the formal definition is shown in (2.10). Comparing to alternative non-linearity functions, ReLU provides speed and stability, which ease the training of large scale deep neural network models.

$$\text{ReLU}(x) = \max(\vec{\mathbf{0}}, x) \tag{2.10}$$

Softmax does element-wise exponential normalization for an output vector, turning its elements into a probability distribution. Concrete definition is shown in (2.11).

$$\text{Softmax}(x) = \frac{\exp(x_i)}{\sum_k \exp(x_k)} \tag{2.11}$$

where $\exp(x)$ represents e^x.

Training

A NN model is trained to minimize the target loss function, $\mathcal{L}(y, y^*)$, that estimates the errors between predictions, y, and the references, y^*. By the word *training*, we simply refer to the process of calibrating the model's parameters such that the calibrated model yields the lower errors than the un-calibrated one. An algorithm used for calibration has the name of *optimizer*, of which one of the most popular variant is stochastic gradient descent (SGD).

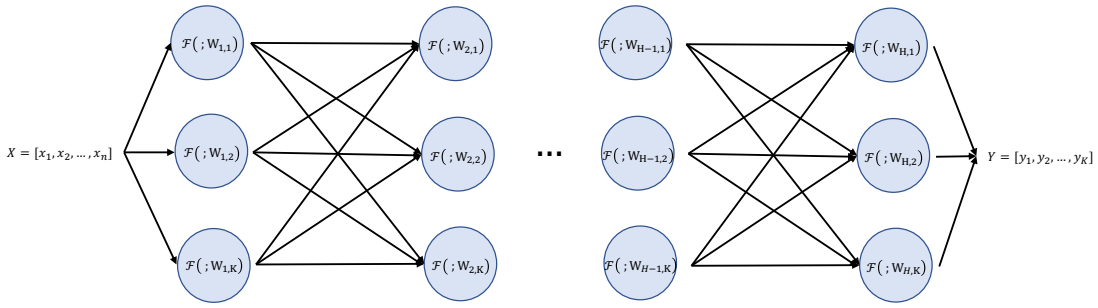


Figure 2.1: Deep Neural Network.

Concretely, SGD computes the gradient of the *differentiable* loss functions with respect to a particular weight at the input x. Then, the weights are adjusted by magnitude-scaled gradients in the direction that will produce the lower loss value, as shown in (2.12) and (2.13).

$$W \leftarrow W - r \times \nabla_W \mathcal{L}(\mathcal{F}(x; W, b), y^*) \tag{2.12}$$

$$b \leftarrow b - r \times \nabla_b \mathcal{L}(\mathcal{F}(x; W, b), y^*) \tag{2.13}$$

where r is the learning rate, a hyperparameter that controls the learning speed. The optimization procedure is recurrently applied until the gradient diminishes or the model converges to the possibly minimum loss value.

There are a lot of optimizers out there such as SGD with momentum (Sutskever et al., 2013), Adadelta (Zeiler, 2012), and Adam (Kingma and Ba, 2014). The main differences between those are the calibrated learning rate, the calibrated gradient magnitude. We encourage interesting readers to find more in (Goodfellow et al., 2016).

2.5.2 Fully-connected Neural Networks

Fully-connected Neural Networks (FNN) is the term used to call the variant of a NN such that all neurons in the same layer connect to all inputs. This architecture is the original NN and is depicted in 2.1. The name was given later when there are many variations of NN.

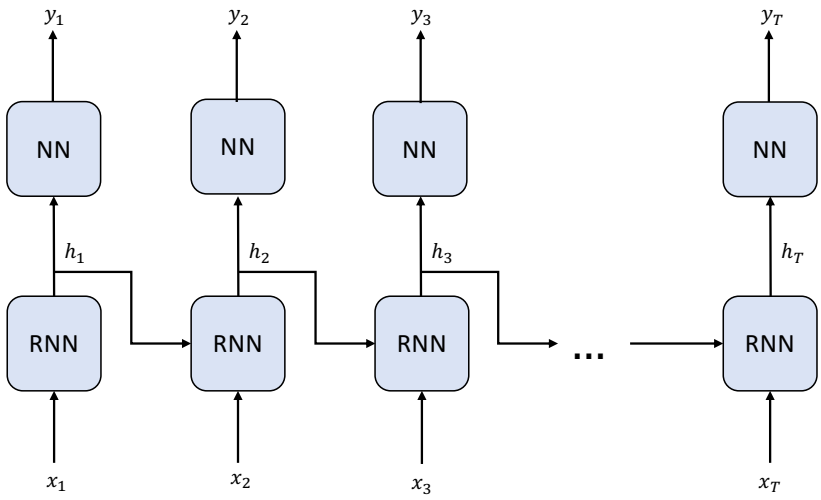


Figure 2.2: Recurrent Neural Networks

2.5.3 Recurrent Neural Networks

Recurrent Neural Network (RNN) is a variant of NN that allows temporal dependencies between consecutive inputs (Elman, 1990). In the previous section, DNN considers only the input of each timestep for making a prediction. RNN accumulates information of every timestep in the previous and propagates them to the last layer of the next step, as shown in (2.14).

$$h_t = \theta(W_i x_t + b_i + W_h h_{t-1} + b_h) \tag{2.14}$$

where W_i is the weights for an input, W_h is the weights for the previous hidden state, b_i is a bias for an input, b_h is a bias for previous hidden state, *theta* is an activation function. The outputted hidden state, h_t, can be directly used as a final output or an input to consecutive NN layers.

The state propagation allows the model to consider both the previous contexts and the current input in order to produce the prediction. Theoretically, RNN can model infinitely long dependencies. However, in practical, numerical stability hinders long dependencies modeling of RNN. Moreover, state propagating limits RNN from going parallel as it cannot make such a prediction without knowing what is the previous, resulting in a longer training time than other variants.

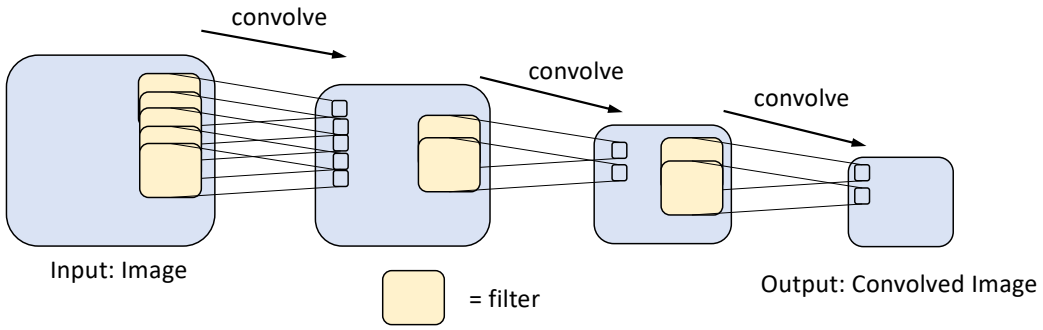


Figure 2.3: Convolutional Neural Networks

2.5.4 Convolutional Neural Networks

Convolutional Neural Network (CNN) (Lecun et al., 1998) has been proposed to be a translation-invariant spatial contexts extractor. CNN used the concept of *convolution*, which is element-wise multiplications between *filters* and inputs in each area, as shown in Fig 2.3. The convolution operation is used to capture signals in local areas of an input.

Unlike RNN, convolution does not require previous predictions. Thus, the outputs of each timestep can be determined in parallel. However, even though long-term dependencies between inputs can be learned implicitly using the large receptive fields provided by a deep stack of CNN layers, the explicit conditioning between outputs of a sequence could not possibly happen without external tools.

2.6 Performance Evaluation

Performance evaluation methods are specific to the tasks. In this dissertation, we use three different metrics: Character Error Rate, Word Error Rate, and Perplexity.

2.6.1 Levenshtein based Error Rate

The performance of ASR and HTR systems are usually be evaluated using Character Error Rate (CER) and Word Error Rate (WER). Both metrics are defined similarly as Levenshtein distances between hypotheses and ground truths divided by the length of the ground truths. The metrics can be interpreted as the number of operations needed

to transform hypotheses to ground truths. A concrete definition is defined in (2.15).

$$\text{error rate} = \frac{Lev}{R} = \frac{I + D + S}{R} \tag{2.15}$$

where Lev is Levenshtein distance, and R is the length of ground truths. 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.

2.6.2 Perplexity

Perplexity (PP) measures the uncertainty of a probability distribution. In other words, it tells the performance of predicting observed samples, x, for a given probability distribution, p, as defined in (2.16).

$$PP(p) = \prod_x p(x)^{-p(x)} = 2^{-\sum_x p(x) \log_2 p(x)} = 2^{H(p)} \tag{2.16}$$

Perplexity is one of the metrics used to measure the effectiveness of a LM. A high-quality LM should predict the next letters well and hence has a low perplexity.

2.7 Significance Testing

Significance testing measures the likeliness of having the observations under the believed system (null hypothesis). The low likeliness implies that the observations may not follow the null hypothesis and are sampled from another system instead. We would like to measure whether the believed system can precisely explain the observations. In practice, the null hypothesis will be rejected if the likelihood of the observations under the null hypothesis is less than the predefined significance level thresholds (α), which commonly are 0.05, 0.01, or 0.001. In other words, the difference between the underlying system of the observations and the current believed system is statistically significant if the probability of having at least as extreme as the observations under the null hypothesis is more than the significant level.

In this thesis, we use significant testing to measure the difference between the performances of two models, m_1 and m_2. We would like to find out whether the performances of m_1 and m_2 significantly differ from one another.

2.7.1 Matched-Pair Test

The matched-pair test is a significant testing tool that measures the difference between paired data, widely used for ASR (Gillick and Cox, 1989). Given an utterance, the predicted transcriptions of two different ASR are paired. The statistic is calculated on differences of every pair in the test set. Suppose that there are N utterances in the test set, o_j^i is the prediction for utterance i made by system j, and e_j^i be the number of errors on $o_j^i, 1 \leq i \leq N, 1 \leq j \leq 2$. We define $Z^i = e_1^i - e_2^i$ as the difference of errors for the utterance i, and $\hat{\mu}_Z = \sum_{i=1}^{N} Z_i/N$ as the estimated average error difference between two systems. The estimated variance of Z_i is $\hat{\sigma}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(Z_i - \hat{\mu}_Z)^2$. The test statistic for significance comparison is defined as:

$$W = \frac{\hat{\mu}}{(\hat{\sigma}/\sqrt{n})} \tag{2.17}$$

Matched-pair test evaluates the null hypothesis of $\hat{\mu_Z} = 0$ under the assumption that N is large enough. If the extremeness of W under unit normal distribution is less than the significant level, $2P(|w| \geq |W|; \mathcal{N}(0,1)) \leq \alpha$, the difference is statistical significant.

In this work, we use word-level Levenshtein distance between the prediction, o_j^i, and its ground truths as an error, e_j^i. The algorithm also has a formal name of Matched Pairs Sentence-Segment Word Error (MAPSSWE).[1]

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

Chapter III

RELATED WORKS

This chapter presents how traditional ASR and HTR models handle contexts, how they adopt CTC, and how they deal with the shortcomings of context independent training. We firstly provide the evidence of using contexts for ASR and HTR in Section 3.1 and Section 3.2. Then, we present the attempts to incorporate context into CTC in Section 3.3.

3.1 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 (Jurafsky and Martin, 2009).

For models based on deep learning, CTC has been proposed for training end-to-end models. These typically model letters instead of CD units. However, context dependencies are modeled implicitly using recurrent hidden states (Graves et al., 2006). Transducers (Graves et al., 2013) and Sequence-to-Sequence models (Chorowski et al., 2015), 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 (Amodei et al., 2016). 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 also suffer from performance degradation. Potential remedies include using rescoring or iterative decoding where successive refinements are performed on the previous outputs (Higuchi et al., 2020, 2021; Chan et al., 2020).

3.2 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 the HMM-based framework, which is analogous to lexicon-free ASR models (Hu et al., 1996; Bengio et al., 1995).

The combination of CTC and RNN was firstly adopted as an alternative to the existing HMM frameworks (Graves et al., 2008a,b; Dreuw et al., 2009). As CTC lacks dependencies, it was used with context rich models such as RNN and multi-dimensional RNN (Voigtlaender et al., 2016; Graves and Schmidhuber, 2009; Puigcerver, 2017; Carbune et al., 2020). Recently, non-recurrent models have shown promising results while reducing the computation latency. This introduces a wave of research based on non-recurrent models as they have no computation bottlenecks (Yousef et al., 2020; Coquenet et al., 2020; Sharma and Jayagopi, 2020).

3.3 Context-dependent CTC

Incorporating context dependencies into CTC models has been mostly based on using subword modeling such as Byte-Pair Encoding (Zenkel et al., 2017), WordPiece (Synnaeve et al., 2019), and context-dependent output units (Chorowski et al., 2019), 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 (Tassopoulou et al., 2021) or different WordPiece sizes (Sanabria and Metze, 2018), 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 (Liu et al., 2017) introduces a modification of the CTC loss that can aggregate the different possible segmentations of the output tokens on-the-fly. Recurrent transducer (Graves, 2012) autoregressively wraps posteriors of a CTC encoder with a language model and trains both modules together. Imputer (Chan et al., 2020) 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 (Zhang et al., 2015, 2016) that predicted future ground truths for recurrent hybrid ASR models.

3.4 Summary

The review of related works presented in this chapter aims to illustrate the developments in the usage of context dependencies in ASR and HTR. As described in Sec.3.3, we observed limitations of preserving the NAR decoding scheme that does not allows direct context conditioning. As we found that existing works have attempted to address the problems using hidden representation in order to *inducing* context conditioning for the independent predictions, we strongly believe that the *contextualized* objective function can improve the performance of ASR and HTR NAR models.

Chapter IV

CONTEXTUALIZED CONNECTIONIST TEMPORAL CLASSIFICATION

This chapter introduces Contextualized Connectionist Temporal Classification (CCTC), the novel approach for training NAR sequence modeling.

4.1 Motivation

The motivation of CCTC is to mitigate the misspelling problems that occurred from the CTC assumption of conditional independence without increasing the inference time. As we would like to preserve the inference speed, CCTC has to induce context conditions in the models solely in the training phase.

The obstacle of incorporating contextual information into CTC models comes from the characteristics of CTC preferring blanks over the actual alphabet for the predictions. This hindrance arises from the conditional independence assumption that encourages the model to isolate its predictions. CTC models mostly produce blank tokens, $\pi_t = \epsilon$, and only predict the actual alphabets, $\pi_t \in \mathcal{A}$, 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.

With the successes of predicting contexts in hybrid ASR systems, we hypothesize that providing the required contextual information to the model in this manner should also alleviate the misspelling issues in CTC models without sacrificing speed. Therefore, we propose to use context prediction subtask for NAR CTC models together with the original main task CTC in a multitask-learning. We then proposed the novel label obtaining algorithm that addresses sparseness of alphabet tokens within the predictions, hence allowing context dependencies without requiring external alignments.

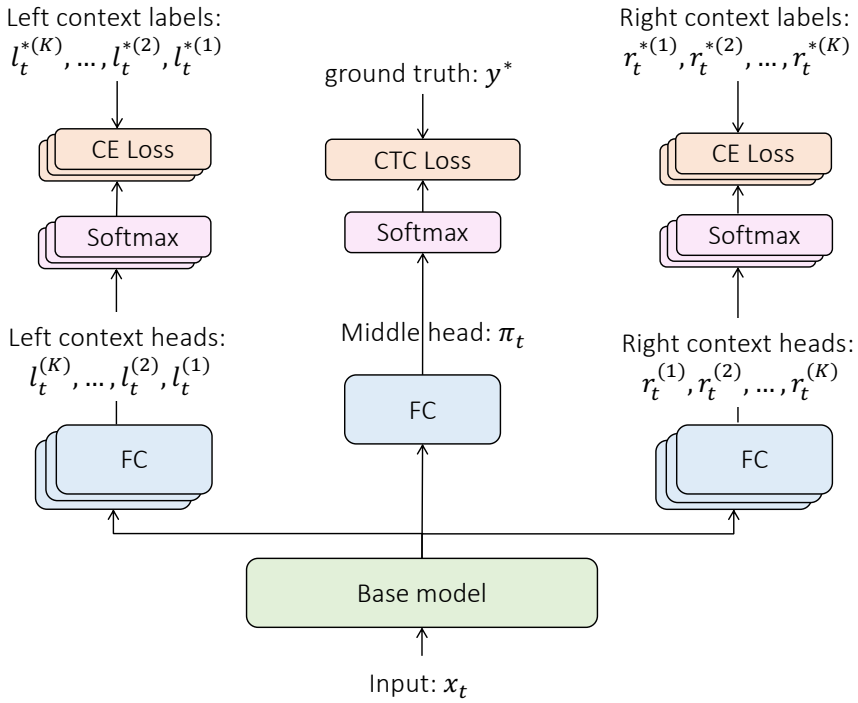


Figure 4.1: The CCTC architecture used in this work. The baseline models are modified by adding $2K$ extra prediction heads. The models are now aware of contexts as it learns to predict left, middle, and right characters simultaneously.

4.2 CCTC training loss

CCTC loss softens the strength of independent predictions by implicitly introducing context conditioning to 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.

The overview of our framework is depicted in Figure 4.1. 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 log probability for the context labels, which is the cross-entropy (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, $\mathcal{L}_{CT}$, can be defined as shown in (4.1).

$$\mathcal{L}_{CT} = -\sum_t \alpha \log P(l_t^*) + \beta \log P(r_t^*) \tag{4.1}$$

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 k^{th} 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:

$$\mathcal{L}_{CT} = -\sum_t^T \sum_k^K \alpha^{(k)} \log P(l_t^{*(k)}) + \beta^{(k)} \log P(r_t^{*(k)}) \tag{4.2}$$

Finally, the CCTC loss is the summation of the CTC loss and the normalized context loss as shown in (4.3).

$$\mathcal{L}_{CCTC} = \mathcal{L}_{CTC} + \frac{\mathcal{L}_{CT}}{U} \tag{4.3}$$

For inference, only the middle head is kept. Thus, models trained with CCTC 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.

4.3 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 first context heads should try to predict the adjacent outputs generated from the middle head for the current frame. 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 k$^{\text{th}}$-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, $h = (h_1, h_2, ..., h_L) : L \leq T$, 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, $p = (p_1, p_2, ..., p_T) : p_t \in [1, L], p_t \leq p_t + 1$. 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 k$^{\text{th}}$ 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 summarize the algorithm in Algorithm 1 and demonstrate this process with an example in Figure 4.2.

Algorithm 1 Acquiring labels for context losses

Given: π - 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 π_t.
$t \leftarrow 1$
while $t \leq T$ **do**
  $l_t^* \leftarrow \text{LeftSearch}(h, p_t, K)$ ▷ $l_t^* = (l_t^{*(1)}, l_t^{*(2)}, ..., l_t^{*(K)})$
  $r_t^* \leftarrow \text{RightSearch}(h, p_t, K)$ ▷ $r_t^* = (r_t^{*(1)}, r_t^{*(2)}, ..., r_t^{*(K)})$
  $t \leftarrow t + 1$
end while

4.4 Model Initialization

Since the labels for context heads are obtained from the main head predictions on-the-fly, optimizing CCTC loss for random networks may cause training difficulties due to noisy labels. Nevertheless, results from our preliminary experiments suggested that CCTC works well with both random and pretrained networks. In our experiments, we used pretrained weights as the initialization because they take less 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.

4.5 Context loss weights

In higher order CCTC 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.[1] As for weights of high order context losses, $\alpha^{(k)} : k > 1$, we have tried several heuristic approaches and found three 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,

[1] 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.

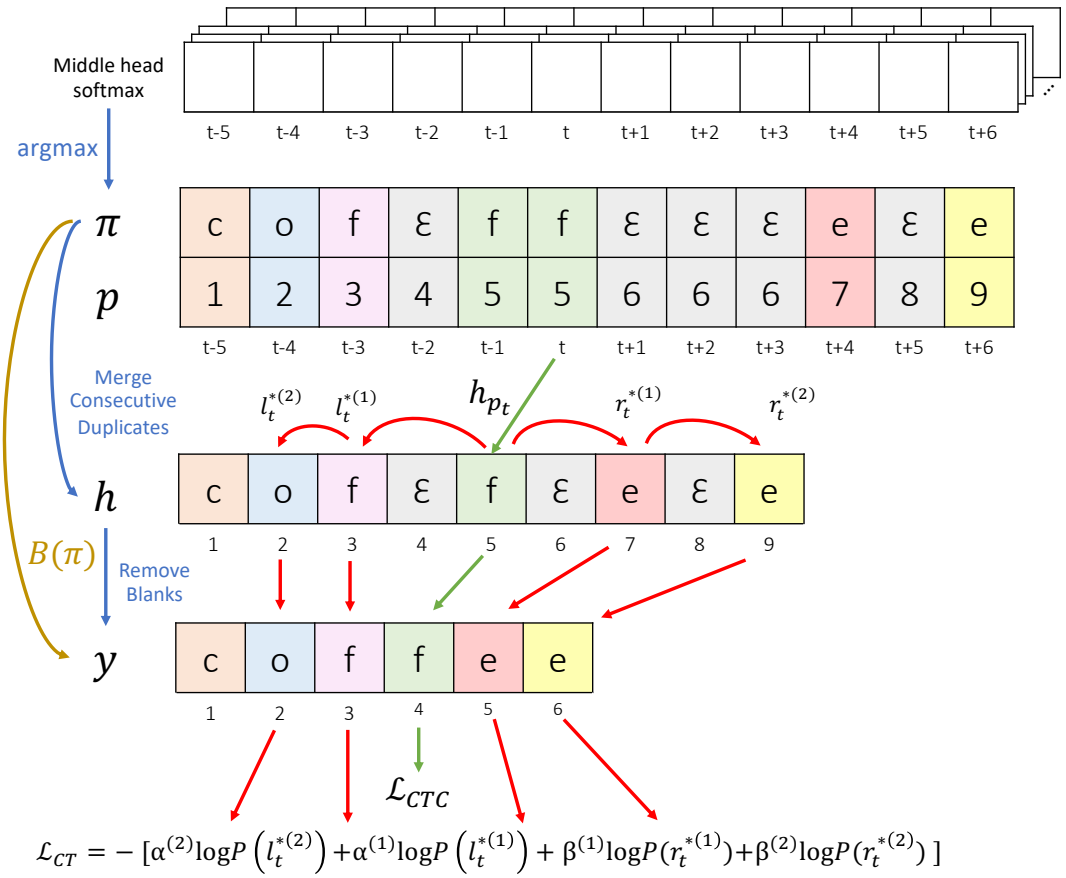


Figure 4.2: Label generation procedure for the CCTC context losses. Given a path from the output of the middle head's softmax, the labels for the left and right heads can be found by searching on a dense path h. For frame index t, the first target to the left and right are f and e. The second targets 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.

$\alpha^{(K)} := 1$. Then, we exponentially decrease weights as the order of the context loss shrinks, $\forall_k \alpha^{(k)} = \alpha^{(K)}/2^{K-k}$. For the third approach, we set the initial 1st-order context loss weight as the target total weight, $\mathbf{w} := \alpha^{(1)}$. Then, we find the exponential sum of weights that will achieve this w. In other words, $\sum_k \alpha^{(k)} \approx \mathbf{w} : \forall_k \alpha^{(k)} = \alpha^{(K)}/2^{K-k}$. This is to keep the total strength of the context loss to be the same as the one found using grid search on just the 1st-order context.

Chapter V

EXPERIMENTS

In this chapter, we provide the experiment results for validating the effectiveness of CCTC loss on three sequence modeling problems, including monolingual ASR, CS ASR, and HTR. We present the details for monolingual ASR and CS ASR experiments in Section 5.1 and Section 5.2, respectively. The models used in both experiments are nearly identical. The only difference is the number of output units. As for HTR, we used two different frameworks for English and Thai, of which their details are presented in Section 5.3 and Section 5.4, respectively.

5.1 Monolingual ASR

This section presents the effectiveness of CCTC loss for monolingual ASR using an in-house ASR framework on the public English dataset, LibriSpeech. Monolingual ASR is a special case of ASR such that the audio is spoken using a single language and can be transcribed using a single set of alphabets.

5.1.1 Dataset

We opted for a widely used LibriSpeech corpus (Panayotov et al., 2015), which comprises reading speech from English audio books. 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* sets. The audios had a 16kHz sampling rate and 16-bit depth. We filtered out utterances that are longer than 16.7 seconds, and applied no data augmentation techniques.

5.1.2 Experimental setups

We adopted a fully-convolutional NAR ASR model, Wav2Letter+ (Kuchaiev et al., 2018), a modified version of Wav2Letter (Collobert et al., 2016; Liptchinsky et al., 2017), as our base model. The model comprises 17 1D-convolutional layers and two fully-

connected layers at the end. We added context prediction heads right after the last layer of the base model, as shown in Figure 4.1.

The default settings of Wav2Letter+[1] were used with some exceptions. The Adam optimizer (Kingma and Ba, 2015) was used instead of the original SGD optimizer. The Layer-wise Adaptive Rate Control (You et al., 2017) and weight decay were not used as we found them hurting the performance. We replaced the polynomial decay with an exponential decay with a rate of 0.98. We trained the models with only the CTC loss for 300 epochs. Afterwards, the context losses were included, and the training resumed for an additional of 100 epochs. We set the starting learning rate to $1e-3$ for the first 300 epochs and then decreased it to $1e-4$ for the rest.

As for LibriSpeech 960 hours, we used the implementation of QuartzNet-5x5 (Kriman et al., 2020) from NeMo toolkit (Kuchaiev et al., 2019). 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.

LM rescoring was also applied to investigate more realistic setups. We used the official released LibriSpeech 3-gram word-level language model. A beam width of 64 was used as a the default. 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.

5.1.3 Results

CCTC models consistently outperform the baseline CTC in the scenario where LM rescoring was unavailable as illustrated in Table 5.1. We found CCTC model with the context size of 2 provided the best results with the relative improvement over the baseline by 3.8% and 3.3% on dev and test sets, respectively. We also provided results from the

[1] `https://github.com/NVIDIA/OpenSeq2Seq`
[2] `https://ngc.nvidia.com/catalog/models/nvidia:nemospeechmodels`

Model	dev-clean			test-clean		
	argmax	beam	3-gram	argmax	beam	3-gram
CTC (Pratap et al., 2019)	-	-	-	-	-	18.97
CTC (our run)	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 5.1: The WER (%) results for the LibriSpeech 100 hours setting. The $*$ symbol indicates a significant difference at $p < 0.05$ compared to the baseline CTC using MAPSSWE two-tailed test.

Model	dev-clean			test-clean		
	argmax	beam	3-gram	argmax	beam	3-gram
CTC (Kriman et al., 2020)	-	-	-	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**

Table 5.2: 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.

Wav2Letter++ (Pratap et al., 2019) model taken from the Wav2Letter tutorial[3] which was trained on the same subset as a strong baseline. 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 5.2. 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 6.2.

[3] `https://github.com/facebookresearch/wav2letter/tree/recipes-conv-glu-paper/tutorials/1-librispeech_clean`

	train	dev-th	dev-cs	test-th	test-cs
duration	150 Hr	22 Hr	1 Hr	24 Hr	7 Hr
#TH utterances	.2M	28K	-	33K	-
#TH-CS utterances	8K	-	1K	-	8K
#TH letters	7M	1M	47K	1M	.3M
#EN letters	84K	-	12K	-	72K
#TH words	2M	.3M	13K	.3M	82k
#EN words	14K	-	2K	-	13K
#TH vocabulary	36K	12K	2K	12K	4K
#EN vocabulary	3K	-	1K	-	3K

Table 5.3: Statistics of the dataset used for CS ASR experiments.

5.2 Code-Switching ASR

This section compares CTC and CCTC performances for CS ASR on an in-house Thai-English CS dataset. CS ASR is a task of transcribing language-mixed spoken audios. The input signal audio for CS ASR contains more than one language within a single sentence. Although no restricted rules for language switching are applied, CS utterances regularly contain two languages. The main language is used as a core grammar for a sentence, in which borrow words or borrow phrases of a side language appear.

One of the critical challenges for CS ASR is that the models need to handle transcribing ambiguity arising from overlapping pronunciation representations of distinct alphabets from many different supported languages. The regular character-based NAR models usually fail as they do not have dependencies between predicted letters in order to make consistent predictions across the sets of alphabet.

5.2.1 Dataset

For CS ASR experiments, we used 200 hours of manually transcribed Thai speech crawled from public Thai podcast YouTube channels. The recordings in the dataset were preprocessed to 16kHz and 16-bit depth. Utterances that are longer than 16.7 seconds were dropped. The training subset of Thai YouTube contains both monolingual Thai (TH) and CS Thai-English utterances. CS sentences were found in 4.4% of the training set. 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*. More details are shown in Table 5.3. The YouTube channels in the test set are different from the training and development sets. Therefore, speakers in the test set are not in the training data. Any hyperparameter tunings were done together on the combined development set.

5.2.2 Experimental setups

We trained Wav2Letter+ models using two-step training strategies and followed almost every configurations used in Section 5.1. Since CS ASR models have 3 times larger alphabet sizes than English ASR models, we decreased the initial learning to $1e-4$ and used the second learning rate of $4e-5$ to increase training stability.

We curated two corpora with 27M words/145M letters from Thai Wikipedia and 69M words/330M letters from Pantip (Thai Q&A forum). For each corpus, we did word tokenization using DeepCut (Kittinaradorn et al., 2019) and trained word-based n-gram models using KenLM (Heafield, 2011). The final LM is obtained by n-gram interpolation. A beam width of 64 was used for LM rescoring. LM weights and insertion penalties were obtained through grid search in the same manner as LibriSpeech in Section 5.1.

5.2.3 Results

Table 5.4 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 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.

Further qualitative analysis shows that CTCC mostly fixes the inconsistencies in the spelling. Figure 5.1 depicts the word “follower” spelled with a mixture of Thai and

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

Table 5.4: The WER (%) results for the Thai YouTube corpus setting. The $*$ and † symbols indicate significant differences at $p < 0.05$ to the baseline CTC and the CCTC$^{(1)}$, respectively. The hypotheses testing were conducted using MAPSSWE two-tailed tests.

English alphabets in the CTC model, while CCTC model outputs English alphabets consistently. The phoneme sequence /ol/ only appears in loanwords in Thai, making the model heavily prefers to output "ol."

Argmax	CTC	คน ที่ มี โฟลเวอร์ เพียง แค่ ห้า พัน
	CCTC	คน ที่ มี polower เพียง แค่ ห้า พัน
3-gram LM	CTC	คน ที่ มี ฟลัวร์ เพียง แค่ ห้า พัน
	CCTC	คน ที่ มี follower เพียง แค่ ห้า พัน
Ground truth		คน ที่ มี follower เพียง แค่ ห้า พัน

Figure 5.1: Comparison between CS ASR models for selected examples. Aligned differences are highlighted in different colors.

5.3 English HTR

In this section, we studied the feasibility of applying CCTC loss on HTR, showing the greater usability of the proposed method across sequence modeling with different input data modalities. From the model point of view, end-to-end FCN HTR models are close but not the same as end-to-end FCN ASR models. There can be some distinctions in features extracting layers, of which different sizes of receptive fields may be necessary for enabling task-specific features extraction. However, from the optimization point of view, both tasks can be defined as the same sequence modeling problem that can be addressed by both CTC and CCTC losses.

5.3.1 Dataset

We employed standard English HTR dataset, IAM (Marti and Bunke, 2002). 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 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 (Coquenet et al., 2020). We also followed the data preprocessing methods in (Coquenet et al., 2020), including grayscaling the images, normalizing the heights to 64 pixels, and standardizing the intensity values.

5.3.2 Experimental setups

We used GFCN proposed in (Coquenet et al., 2020) as the base model for the IAM dataset. The model comprises 12 2D-depthwise separable convolutional (Chollet, 2017) and 18 2D-convolutional layers. We added the context heads to the pretrained networks, provided by the authors[4], 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 (Paszke et al., 2019). We selected the checkpoint with the best validation CER for the comparison.

5.3.3 Results

The results on the IAM dataset are summarized in Table 5.5. 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 5.2 shows examples of the differences between model outputs. We found that CCTC models do noticeably better on hard-to-read handwriting. In Figure 5.2(a),

[4] `https://github.com/FactoDeepLearning/LinePytorchOCR`

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*

Table 5.5: The performance comparison on IAM. The $*$ symbol indicates word-level significant differences at $p < 0.05$ to the baseline using MAPSSWE two-tailed tests.

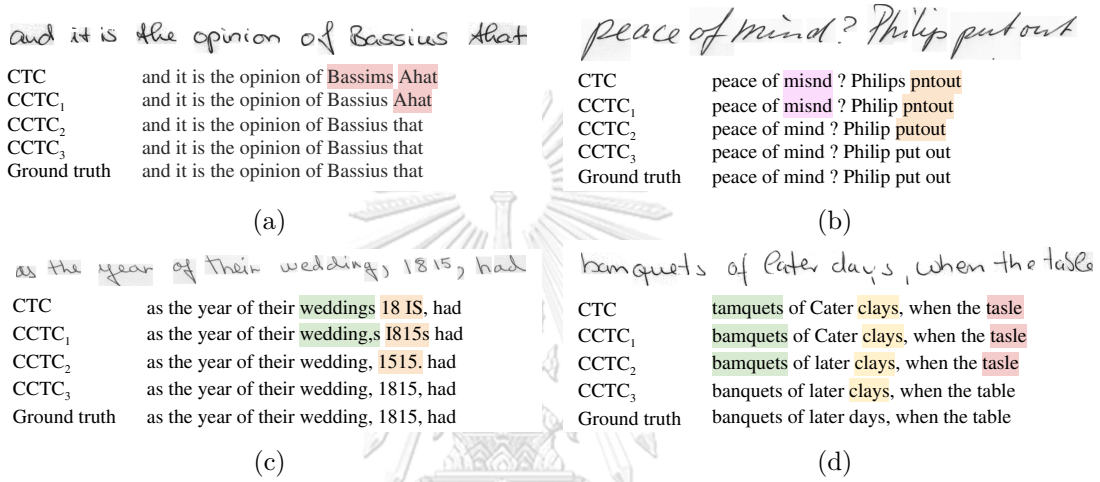


Figure 5.2: Selected prediction examples from the IAM test set. The mispredictions are highlighted in color.

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 Figure 5.2(b) is also very vague, and further contexts are needed to mitigate this problem. In Figure 5.2(c), 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.

5.4 Thai HTR

This section presents the performances of CCTC loss for training Thai HTR models in order to show the robustness of CCTC over languages.

5.4.1 Dataset

We use BEST for Thai HTR, which is a standard benchmark for handwritten text recognition provided by The National Electronics and Computer Technology Center (NECTEC) Sinthupinyo (2018); Sinthupinyo as part of the 2019 Thai handwritten text recognition contest[5]. 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 provided by Chaiyaroj et al. (2021). We followed the data splitting, augmentation strategies, pre-processing, and data synthesis methods as proposed in Chaiyaroj et al. (2021). 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.

5.4.2 Experimental setups

We used the model, with a single prediction head, proposed in (Chaiyaroj et al., 2021), as the base model. The model consists of 12 layers of 2-dimensional (2D) convolution, 2 layers of 2D depthwise separable convolution (Chollet, 2017), and 4 layers of 1-dimensional convolution. We followed the two-step training methodology. Since we already had the pretrained model, we attached context heads to the pretrained CTC network and additionally trained the model for another 400 epochs. Adam optimizer (Kingma and Ba, 2015) 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.

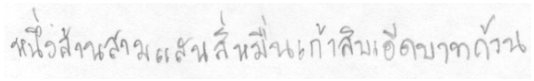
5.4.3 Results

For the BEST dataset, CCTC models consistently outperform the CTC model, as shown in Table 5.6. We opted for CER as the only evaluation metric since BEST

[5] `https://www.nectec.or.th/`

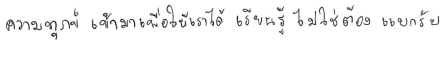
Model	validation	test-seen	test-unseen
CTC	10.62	11.38	35.03
CCTC$^{(1)}$	9.93	10.72	35.15
CCTC$^{(2)}$	9.83	**10.26**	**34.79**
CCTC$^{(3)}$	9.76	10.49	36.42
CCTC$^{(4)}$	**9.71**	10.42	35.42

Table 5.6: The argmax CER (%) evaluation on the BEST.



CTC	หนึ่งล้านสามแสนสี่หมื่นเก้าสิบเอ็ดบาทถ้วน
CCTC	หนึ่งล้านสามแสนสี่หมื่นเก้าสิบเอ็ดบาทถ้วน
Ground truth	หนึ่งล้านสามแสนสี่หมื่นเก้าสิบเอ็ดบาทถ้วน

(a) A case with narrow spacing handwriting.



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

(b) A case with ambiguous handwriting.

Figure 5.3: Output comparison between different Thai HTR models on selected examples. Highlights indicate prediction differences.

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. Figure 5.3(a) depicts handwriting with very narrow spacing between characters. The CTC model predicts an extra character that has a similar shape to the combination of two characters while Figure 5.3(b) 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.

Chapter VI

DISCUSSIONS

This chapter provides discussions regarding the characteristics and the effects of using CCTC for training sequence models.

6.1 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 that shows 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.

Figure 6.1 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 slight

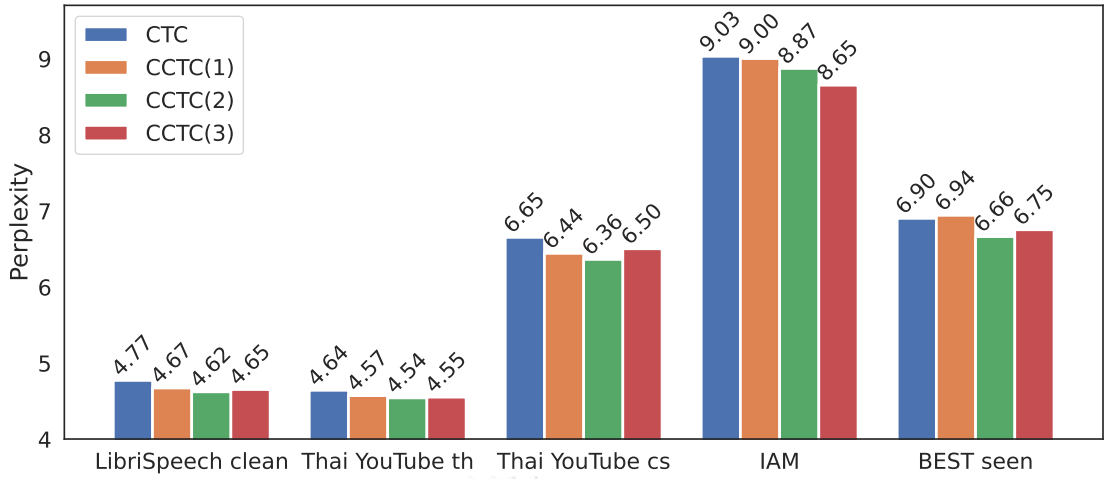


Figure 6.1: Perplexity comparisons between different context sizes. Perplexity scores are computed from the predictions using argmax decoding on each test sets.

different from the seen test set.

6.2 Discrepancy between context predictions and LM rescoring

Even though increasing context sizes for CCTC models can 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 5.1 and Table 5.4, CCTC$^{(1)}$ generally had the highest effectiveness when the external LM was incorporated. 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.

Figure 6.2 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 Figure 6.2(a), 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. Figure 6.2(b) depicts an example from LibriSpeech dev-clean that the LM causes the error in the CCTC model.

Aggressive context dependencies from CCTC may cause conflicts between the internal language representations and the external LM rescorer. Note that we used

Argmax	CTC	nfunction ที่ ใช้ นั้น 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.

Argmax	CTC	FIFTEEN OFFICERS OF OUR LITTLE HALF REGIMENT WER DEADTOR WOUNDED
	CCTC	FIFTEEN OFFICERS OF OUR LITTLE HALF REGIMENT WERE DEATTOR WOUNDED
3-gram LM	CTC	FIFTEEN OFFICERS OF OUR LITTLE HALF REGIMENT WERE DEAD OR WOUNDED
	CCTC	FIFTEEN OFFICERS OF OUR LITTLE HALF REGIMENT WERE DEAR WOUNDED
Ground truth		FIFTEEN OFFICERS OF OUR LITTLE HALF REGIMENT WERE DEAD OR WOUNDED

(c) Incompatibility between CCTC and LM scores.

Figure 6.2: Selected samples showcasing the possible mismatch between CCTC and LM rescoring. The words of interested are highlighted.

word level language models. Further investigation on methods that can learn CCTC jointly with the external LM might reduce this discrepancy.

6.3 Relations 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 6.1 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

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

Table 6.1: 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.

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 (Amodei et al., 2016).

6.4 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 (Guo et al., 2018) and DWA (Liu et al., 2019). 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 (Vandenhende et al., 2021; Sener and Koltun, 2018).

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 also 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 (Watanabe et al., 2017; Povey et al., 2016).

6.5 Optimal size for context losses

We found that the different models, trained on different datasets, had different optimal context sizes. In ASR, we reached the best performance with CCTC$^{(2)}$, but CCTC GFCN, which was trained on IAM, had the best results using the context size of 3. We hypothesize that large context sizes would benefit the performance if the models have proper loss weights and sufficient receptive fields. Since we used static manual-searched weights throughout the training, we might miss the optimal weight, hence encountering dissimilarities between experiments. Moreover, optimizing many loss functions at the same time can be unstable. A more sophisticate algorithm that stabilize the training process might also help.

6.6 Generalization of CCTC models

In experiment 5.4.3, we found mismatches between the performance on validation and test sets as we increased the context sizes. As presented in Table 5.6, CCTC models with large context sizes had the lower validation CER, but the CER of test-seen got worse after the context size was wider than 2. This mismatch implies overfitting, and it was more severe on the test-unseen, on which both the baseline and the proposed methods underperformed the pretrained model. The situation here might raise the concern that context losses may encourages remembering niche patterns, which could not be generalized.

Nevertheless, BEST was an extreme case since its training and validation sets were closely related, but the test-unseen was very distinct. Moreover, the dataset comprises plenty of duplication of few unique sentences. On the other hand, the overfitting was less likely to happen on LibriSpeech, Thai YouTube, and IAM, of which the training sets had diversity and were closer to the test sets.

Besides, these results also show that CCTC loss actually raises context awareness, which can lead to poor generalization if the diversity in the training data was not enough, supporting Section 6.1. In typical datasets, CCTC loss works perfectly fine.

Model	dev-clean			test-clean		
	argmax	beam	3-gram	argmax	beam	3-gram
CTC	7.09	6.48	**4.46**	7.53	6.88	**4.76**
CCTC$^{(2)}$	**6.86**	**6.44**	4.65	**7.01**	**6.57**	4.88

Table 6.2: The WER (%) comparison for from-scratch training of QuartzNet-5x5 on LibriSpeech 960 hours.

6.7 Necessity of pretraining for CCTC loss

Since references of context heads were obtained from predictions of the model, two-step training has been proposed to reduce the problem of noisy labels. Concretely, a random network is firstly trained using only CTC loss. CCTC loss is added afterward to continuing train the pretrained CTC model.

Nevertheless, we found that the two-step training is unnecessary. Table 6.2 shows that the CCTC$^{(2)}$ model, which is the best setting, surprisingly achieves a better WER than the CTC model in a from-scratch training setup. Therefore, we can use CCTC in both settings: training models from the ground up and improving the existing pretrained models.

Chapter VII

CONCLUSION

In this thesis, we proposed the novel CCTC loss for training NAR sequence modeling, specifically ASR and HTR. CCTC allows frame-level context conditioning without requiring ground truth alignments. We showed that the existing pretrained CTC networks could be continuing trained using CCTC loss, which generally provides more improvements than continuing trained using the original CTC. We illustrated that CCTC can leverage context conditioning for addressing ambiguous samples, resulting in more consistent predictions in CS ASR and less misspellings in ASR and HTR models. We also showed that CCTC models with wider context sizes are usually superior to the shallow ones but have more chance to inconsistent with the scores of the external LM. Moreover, we demonstrated that letters in the predicted sequences of CCTC models have tighter dependencies and share more information with LM than the regular CTC models. In the future, we plan to investigate joint training with the language model in order to fully utilize the implicit LM learned by CCTC.

REFERENCES

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al. 2016. Deep speech 2: End-to-end speech recognition in english and mandarin. In International conference on machine learning, pp. 173–182. :

Baek, J., Kim, G., Lee, J., Park, S., Han, D., Yun, S., Oh, S. J., and Lee, H. 2019. What is wrong with scene text recognition model comparisons? dataset and model analysis. In Proceedings of the IEEE International Conference on Computer Vision, pp. 4715–4723. :

Bahdanau, D., Cho, K., and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014):

Bengio, Y., LeCun, Y., Nohl, C., and Burges, C. 1995. Lerec: A nn/hmm hybrid for on-line handwriting recognition. Neural computation 7.6 (1995): 1289–1303.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. 2020. Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020):

Carbune, V., Gonnet, P., Deselaers, T., Rowley, H. A., Daryin, A., Calvo, M., Wang, L.-L., Keysers, D., Feuz, S., and Gervais, P. 2020. Fast multi-language lstm-based online handwriting recognition. International Journal on Document Analysis and Recognition (IJDAR) 23.2 (2020):

Chaiyaroj, A., Sri-isaranusorn, P., Wangkriangkri, P., Kobchaisawat, T., and Chalidabhongse, T. 2021. Handwritten recognition using convolutional neural networks on components of thai script. To be appeared at IEEE Access (2021):

Chan, W., Saharia, C., Hinton, G., Norouzi, M., and Jaitly, N. 2020. Imputer: Sequence modelling via imputation and dynamic programming. arXiv preprint arXiv:2002.08926 (2020):

Chollet, F. 2017. Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1251–1258. :

Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. 2015. Attention-based models for speech recognition. arXiv preprint arXiv:1506.07503 (2015):

Chorowski, J., Łańcucki, A., Kostka, B., and Zapotoczny, M. 2019. Towards Using Context-Dependent Symbols in CTC Without State-Tying Decision Trees. In Proc. Interspeech 2019, pp. 4385–4389. :

Chuang, S.-P., Chang, H.-J., Huang, S.-F., and Lee, H.-y. 2021. Non-autoregressive mandarin-english code-switching speech recognition with pinyin mask-ctc and word embedding regularization. arXiv preprint arXiv:2104.02258 (2021):

Collobert, R., Puhrsch, C., and Synnaeve, G. 2016. Wav2letter: an end-to-end convnet-based speech recognition system. arXiv preprint arXiv:1609.03193 (2016):

Coquenet, D., Chatelain, C., and Paquet, T. 2020. Recurrence-free unconstrained handwritten text recognition using gated fully convolutional network. In 2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 19–24. :

Dreuw, P., Rybach, D., Gollan, C., and Ney, H. 2009. Writer adaptive training and writing variant model refinement for offline arabic handwriting recognition. In ICDAR. :

Elman, J. L. 1990. Finding structure in time. Cognitive Science 14.2 (1990): 179 – 211.

Gillick, L. and Cox, S. J. 1989. Some statistical issues in the comparison of speech recognition algorithms. In International Conference on Acoustics, Speech, and Signal Processing,, pp. 532–535. :

Goodfellow, I., Bengio, Y., and Courville, A. 2016. Deep Learning. MIT Press. `http://www.deeplearningbook.org`.

Graves, A. 2012. Sequence transduction with recurrent neural networks. arXiv preprint arXiv:1211.3711 (2012):

Graves, A. and Jaitly, N. 2014. Towards end-to-end speech recognition with recurrent neural networks. In International conference on machine learning, pp. 1764–1772. :

Graves, A. and Schmidhuber, J. 2009. Offline handwriting recognition with multidimensional recurrent neural networks. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L. (ed.), Advances in Neural Information Processing Systems, volume 21. : Curran Associates, Inc.

Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd international conference on Machine learning, pp. 369–376. :

Graves, A., Fernández, S., Liwicki, M., Bunke, H., and Schmidhuber, J. 2008a. Unconstrained online handwriting recognition with recurrent neural networks. In Advances in Neural Information Processing Systems 20, NIPS 2008. :

Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., and Schmidhuber, J. 2008b. A novel connectionist system for unconstrained handwriting recognition. IEEE transactions on pattern analysis and machine intelligence 31.5 (2008): 855–868.

Graves, A., Mohamed, A.-r., and Hinton, G. 2013. Speech recognition with deep recurrent neural networks. In 2013 IEEE international conference on acoustics, speech and signal processing, pp. 6645–6649. :

Gu, J. and Kong, X. 2020. Fully non-autoregressive neural machine translation: Tricks of the trade. arXiv preprint arXiv:2012.15833 (2020):

Gu, J., Bradbury, J., Xiong, C., Li, V. O., and Socher, R. 2018. Non-autoregressive neural machine translation. In ICLR. :

Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., et al. 2020. Conformer: Convolution-augmented transformer for speech recognition. arXiv preprint arXiv:2005.08100 (2020):

Guo, M., Haque, A., Huang, D.-A., Yeung, S., and Fei-Fei, L. 2018. Dynamic task prioritization for multitask learning. In ECCV. :

Heafield, K. 2011. KenLM: faster and smaller language model queries. In Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation, pp. 187–197. Edinburgh, Scotland, United Kingdom:

Higuchi, Y., Watanabe, S., Chen, N., Ogawa, T., and Kobayashi, T. 2020. Mask ctc: Non-autoregressive end-to-end asr with ctc and mask predict. arXiv preprint arXiv:2005.08700 (2020):

Higuchi, Y., Inaguma, H., Watanabe, S., Ogawa, T., and Kobayashi, T. 2021. Improved mask-ctc for non-autoregressive end-to-end asr. In ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 8363–8367. :

Hu, J., Brown, M. K., and Turin, W. 1996. Hmm based online handwriting recognition. IEEE Transactions on pattern analysis and machine intelligence 18.10 (1996): 1039–1045.

Jaitly, N., Vanhoucke, V., and Hinton, G. E. 2014. Autoregressive product of multi-frame predictions can improve the accuracy of hybrid models. In INTERSPEECH. :

Jurafsky, D. and Martin, J. 2009. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall series in artificial intelligence. Pearson Prentice Hall. ISBN 9780131873216. Available from: https://books.google.co.th/books?id=fZmj5UNK8AQC .

Kingma, D. P. and Ba, J. 2015. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (ed.), 3rd International Conference on Learning

Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. :

Kingma, D. P. and Ba, J. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014):

Kittinaradorn, R., Achakulvisut, T., Chaovavanich, K., Srithaworn, K., Chormai, P., Kaewkasi, C., Ruangrong, T., and Oparad, K. 2019. DeepCut: A Thai word tokenization library using Deep Neural Network [Online]. Available from: http://doi.org/10.5281/zenodo.3457707 [2019,September].

Kriman, S., Beliaev, S., Ginsburg, B., Huang, J., Kuchaiev, O., Lavrukhin, V., Leary, R., Li, J., and Zhang, Y. 2020. Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions. In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6124–6128. :

Kuchaiev, O., Ginsburg, B., Gitman, I., Lavrukhin, V., Li, J., Nguyen, H., Case, C., and Micikevicius, P. 2018. Mixed-Precision Training for NLP and Speech Recognition with OpenSeq2Seq. arXiv preprint arXiv:1805.10387 (2018):

Kuchaiev, O., Li, J., Nguyen, H., Hrinchuk, O., Leary, R., Ginsburg, B., Kriman, S., Beliaev, S., Lavrukhin, V., Cook, J., et al. 2019. Nemo: a toolkit for building ai applications using neural modules. arXiv preprint arXiv:1909.09577 (2019):

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86 (12 1998): 2278 – 2324.

Li, K., Li, J., Ye, G., Zhao, R., and Gong, Y. 2019a. Towards Code-switching ASR for End-to-end CTC Models. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6076–6080. :

Li, Z., Lin, Z., He, D., Tian, F., Qin, T., Wang, L., and Liu, T.-Y. 2019b. Hint-Based Training for Non-Autoregressive Machine Translation. In EMNLP-IJCNLP. :

Liptchinsky, V., Synnaeve, G., and Collobert, R. 2017. Letter-based speech recognition with gated ConvNets. arXiv preprint arXiv:1712.09444 (2017):

Liu, H., Zhu, Z., Li, X., and Satheesh, S. 2017. Gram-CTC: Automatic unit selection and target decomposition for sequence labelling. arXiv preprint arXiv:1703.00096 (2017):

Liu, S., Johns, E., and Davison, A. J. 2019. End-to-end multi-task learning with attention. In CVPR. :

Luo, N., Jiang, D., Zhao, S., Gong, C., Zou, W., and Li, X. 2018. Towards end-to-end code-switching speech recognition. arXiv preprint arXiv:1810.13091 (2018):

Margner, V. and El Abed, H. 2007. Arabic handwriting recognition competition. In ICDAR 2007. :

Marti, U.-V. and Bunke, H. 2002. The IAM-database: an english sentence database for offline handwriting recognition. International Journal on Document Analysis and Recognition 5.1 (2002): 39–46.

Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. 2015. Librispeech: an ASR corpus based on public domain audio books. In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5206–5210. :

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. 2019. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (ed.), Advances in Neural Information Processing Systems 32, pp. 8024–8035. : Curran Associates, Inc.

Povey, D., Peddinti, V., Galvez, D., Ghahremani, P., Manohar, V., Na, X., Wang, Y., and Khudanpur, S. 2016. Purely sequence-trained neural networks for asr based on lattice-free mmi. In Interspeech. :

Pratap, V., Hannun, A., Xu, Q., Cai, J., Kahn, J., Synnaeve, G., Liptchinsky, V., and Collobert, R. 2019. Wav2letter++: A fast open-source speech recognition system. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6460–6464. :

Puigcerver, J. 2017. Are multidimensional recurrent layers really necessary for handwritten text recognition? In 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), volume 1, pp. 67–72. :

Qiu, Z., Li, Y., Li, X., Metze, F., Campbell, W. M., and AI, A. A. 2020. Towards context-aware end-to-end code-switching speech recognition. Proc. Interspeech 2020 (2020): 4776–4780.

Rabiner, L. R. 1989. A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE 77.2 (1989): 257–286.

Sanabria, R. and Metze, F. 2018. Hierarchical multitask learning with ctc. In 2018 IEEE Spoken Language Technology Workshop (SLT), pp. 485–490. :

Sener, O. and Koltun, V. 2018. Multi-task learning as multi-objective optimization. In NIPS. :

Shan, C., Weng, C., Wang, G., Su, D., Luo, M., Yu, D., and Xie, L. 2019. Investigating end-to-end speech recognition for Mandarin-English code-switching. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6056–6060. :

Sharma, A. and Jayagopi, D. B. 2020. Towards efficient unconstrained handwriting recognition using dilated temporal convolution network. Expert Systems with Applications 164 (2020): 114004.

Sinthupinyo, W. 2018. BEST 2019 (Handwritten Recognition) : การรู้จำลายมือเขียน – BEST [Online]. Available from: https://thailang.nectec.or.th/best/2018/06/20/best-2019-handwrittenrecognition-objective [2018,].

Sinthupinyo, W. BEST 2020 (Handwritten Recognition) : การรู้จำลายมือเขียน – BEST.

Sun, Z., Li, Z., Wang, H., Lin, Z., He, D., and Deng, Z.-H. 2019. Fast Structured Decoding for Sequence Models. In NeurIPS. :

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. 2013. On the importance of initialization and momentum in deep learning. In International conference on machine learning, pp. 1139–1147. :

Synnaeve, G., Xu, Q., Kahn, J., Likhomanenko, T., Grave, E., Pratap, V., Sriram, A., Liptchinsky, V., and Collobert, R. 2019. End-to-end asr: from supervised to semi-supervised learning with modern architectures. arXiv preprint arXiv:1911.08460 (2019):

Tassopoulou, V., Retsinas, G., and Maragos, P. 2021. Enhancing handwritten text recognition with n-gram sequence decomposition and multitask learning. In 2020 25th International Conference on Pattern Recognition (ICPR), pp. 10555–10560. :

Vandenhende, S., Georgoulis, S., Van Gansbeke, W., Proesmans, M., Dai, D., and Van Gool, L. 2021. Multi-task learning for dense prediction tasks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021):

Voigtlaender, P., Doetsch, P., and Ney, H. 2016. Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 228–233. :

Wang, Y., Tian, F., He, D., Qin, T., Zhai, C., and Liu, T.-Y. 2019. Non-Autoregressive Machine Translation with Auxiliary Regularization. In AAAI. :

Watanabe, S., Hori, T., Kim, S., Hershey, J. R., and Hayashi, T. 2017. Hybrid CTC/attention architecture for end-to-end speech recognition. IEEE Journal of Selected Topics in Signal Processing 11.8 (2017): 1240–1253.

Weiss, R. J., Skerry-Ryan, R., Battenberg, E., Mariooryad, S., and Kingma, D. P. 2021. Wave-tacotron: Spectrogram-free end-to-end text-to-speech synthesis. In

ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5679–5683. :

Wu, F., Fan, A., Baevski, A., Dauphin, Y. N., and Auli, M. 2019. Pay less attention with lightweight and dynamic convolutions. arXiv preprint arXiv:1901.10430 (2019):

You, Y., Gitman, I., and Ginsburg, B. 2017. Large batch training of convolutional networks. arXiv preprint arXiv:1708.03888 (2017):

Yousef, M., Hussain, K. F., and Mohammed, U. S. 2020. Accurate, data-efficient, unconstrained text recognition with convolutional neural networks. Pattern Recognition 108 (2020): 107482.

Zeiler, M. D. 2012. Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012):

Zeng, Z., Khassanov, Y., Pham, V. T., Xu, H., Chng, E. S., and Li, H. 2019. On the End-to-End Solution to Mandarin-English Code-Switching Speech Recognition. In Proc. Interspeech 2019, pp. 2165–2169. :

Zenkel, T., Sanabria, R., Metze, F., and Waibel, A. 2017. Subword and crossword units for ctc acoustic models. arXiv preprint arXiv:1712.06855 (2017):

Zhang, Y., Yu, D., Seltzer, M. L., and Droppo, J. 2015. Speech recognition with prediction-adaptation-correction recurrent neural networks. In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5004–5008. :

Zhang, Y., Chuangsuwanich, E., Glass, J., and Yu, D. 2016. Prediction-adaptation-correction recurrent neural networks for low-resource language speech recognition. In 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5415–5419. :

Appendix I

TRAINING OVERHEADS

We measured the dataset-specific training overheads of CCTC compared to CTC models, supporting Section 6.3. In each dataset, before we started the clock, we pre-loaded training samples into memory and trained the model for one iteration to warm up the caching system. Then, we conducted benchmarking and profiling on another ten iterations of training. Note that identical training samples were used throughout the ten training iterations. Therefore, the performance ratio results could be sensibly interpreted as averages of ten batch training. As CCTC loss is only compatible with CPU devices, we studied two different setups, calculating CTC loss on GPU and CPU, in order to get a sense of potential improvement from using CCTC on GPU. Remark that forward and backward passes were still done on GPU.

Table A.1 shows the periods consumed by ten batch training for different models.[1] The models for BEST are not compatible with CPU computing for CTC loss, hence showing only CTC GPU results. The duration was summed over ten training batches. We also showed the proportion of computation used by each operation using PyTorch Profiler[2] in Table A.2. The cuda sync prevents the CPU from proceeding with further executions until every GPU thread finishes, waiting for GPU bottlenecks.

Note that LibriSpeech and Thai YouTube models are identical, except for the number of output units. Utterances in LibriSpeech are generally longer than utterances in Thai YouTube by a factor of three. The performance ratios cannot be directly compared across datasets.

[1] The time consumption of CT and CTC losses for BEST was measured using our own implementation of a timer wrapper. For other datasets, we gathered the number using PyTorch Profiler.

[2] `https://pytorch.org/tutorials/recipes/recipes/profiler_recipe.html`

setup	dataset	model	ct loss (s)	ctc loss (s)	total cpu time (s)	performance ratio (x)
CTC GPU	LibriSpeech	CTC	0.00	0.02	3.13	1.00
		CCTC (1)	0.77	0.02	3.93	0.80
		CCTC (2)	1.18	0.02	4.41	0.71
		CCTC (3)	1.61	0.02	4.82	0.65
	Thai YouTube	CTC	0.00	0.01	1.59	1.00
		CCTC (1)	0.48	0.02	2.09	0.76
		CCTC (2)	0.69	0.02	2.31	0.69
		CCTC (3)	0.92	0.01	2.60	0.61
	IAM	CTC	0.00	0.01	2.42	1.00
		CCTC (1)	0.31	0.02	2.75	0.88
		CCTC (2)	0.57	0.01	3.07	0.79
		CCTC (3)	0.81	0.01	3.34	0.72
		CCTC (4)	1.07	0.01	3.53	0.68
	BEST	CTC	0.00	0.48	5.06	1.00
		CCTC (1)	0.11	0.48	6.42	0.79
		CCTC (2)	0.20	0.48	6.52	0.78
		CCTC (3)	0.28	0.48	6.60	0.77
		CCTC (4)	0.37	0.55	6.76	0.75
CTC CPU	LibriSpeech	CTC	0.00	0.69	5.13	1.00
		CCTC (1)	0.71	0.64	5.76	0.89
		CCTC (2)	1.09	0.66	6.31	0.81
		CCTC (3)	1.49	0.63	6.71	0.76
	Thai YouTube	CTC	0.00	0.12	1.90	1.00
		CCTC (1)	0.43	0.13	2.33	0.82
		CCTC (2)	0.62	0.14	2.59	0.73
		CCTC (3)	0.84	0.17	2.87	0.66
	IAM	CTC	0.00	0.07	2.64	1.00
		CCTC (1)	0.31	0.06	2.84	0.93
		CCTC (2)	0.56	0.10	3.14	0.84
		CCTC (3)	0.82	0.07	3.40	0.78
		CCTC (4)	1.02	0.07	3.71	0.71

Table A.1: Benchmarks for training overhead of CCTC

setup	dataset	model	ct loss (%)	ctc loss (%)	forward (%)	backward (%)	optimizer (%)	cuda sync (%)	the rest (%)
CTC GPU	LibriSpeech	CTC	-	0.6	4.8	3.9	4.4	80.0	6.3
		CCTC (1)	18.0	0.5	4.5	3.4	3.6	65.0	4.9
		CCTC (2)	24.7	0.4	5.8	3.4	3.3	58.0	4.4
		CCTC (3)	31.0	0.4	5.9	3.6	3.2	51.0	5.0
	Thai YouTube	CTC	0.0	0.9	10.0	7.7	8.4	61.0	11.9
		CCTC (1)	20.5	0.8	7.5	7.2	6.6	47.0	10.4
		CCTC (2)	26.8	0.7	7.1	6.6	6.5	43.0	9.2
		CCTC (3)	32.1	0.6	7.6	6.9	5.8	38.0	9.0
	IAM	CTC	-	0.4	18.1	7.8	8.7	1.5	63.6
		CCTC (1)	11.3	0.6	16.1	7.7	9.2	1.4	53.8
		CCTC (2)	18.3	0.5	15.8	7.6	7.4	1.3	49.1
		CCTC (3)	24.4	0.4	14.2	7.2	7.1	1.2	45.4
		CCTC (4)	29.0	0.4	13.7	6.8	6.9	1.1	42.1
CTC CPU	LibriSpeech	CTC	-	13.5	3.0	26.8	2.7	46.2	7.8
		CCTC (1)	12.5	11.1	2.9	23.2	2.5	40.5	7.2
		CCTC (2)	18.0	10.5	3.6	21.8	2.4	37.4	6.3
		CCTC (3)	22.7	9.4	3.9	21.8	2.3	33.9	6.0
	Thai YouTube	CTC	-	6.4	8.2	15.1	7.2	47.8	15.3
		CCTC (1)	18.6	5.6	6.4	12.8	6.0	38.3	12.5
		CCTC (2)	24.4	5.3	6.6	12.8	5.3	34.8	10.8
		CCTC (3)	29.3	6.0	6.9	10.9	5.4	30.1	11.4
	IAM	CTC	0.0	2.7	18.2	11.3	8.1	1.3	58.5
		CCTC (1)	10.7	2.2	15.6	9.8	7.3	1.3	53.1
		CCTC (2)	18.0	3.1	14.6	9.2	6.9	1.2	46.9
		CCTC (3)	23.3	2.0	13.5	9.2	6.6	1.1	44.3
		CCTC (4)	27.2	1.8	12.9	9.3	6.5	1.0	41.3

Table A.2: Profiling for CCTC training

Appendix II

RANDOMLY PICKED PREDICTION SAMPLES

This appendix depicts the differences between the predictions of CTC and CCTC models for each dataset using five random selections. The CER comparisons for the total 30 random samples indicate that CCTC$^{(1)}$ is better than, on par with, and worse than CTC on 14, 6, and 10 examples, respectively. The predictions are illustrated below.

LibriSpeech test

CTC DO NOT THEREFORE THINK THAT THE GOSSIC SCHOOLS NEASY ONE

CCTC(1) DO NOT THEREFORE THINK THAT THE GOSSIC SCHOOL S A NEASY ONE

CCTC(2) DO NOT THEREFORE THINK THAT THE GOFIC SCHOOLS NEASY ONE

CCTC(3) DO NOT THEREFORE THINK THAT THE GOSSIC SCHOOL NEASY ONE

Truth DO NOT THEREFORE THINK THAT THE GOTHIC SCHOOL IS AN EASY ONE

CTC IT FOUND I GOL THA COUNT IT STRANGE AND ALL THE MORT HAR BECAUSE IT HIS NANE CONDITION AND WHICH WAS

CCTC(1) IT FOUND A DOLL THAT COUND IT STRANGE AND ALL THE MORT HARE BECAUSE IT HIS ANE CONDITION AND WHICH WAS

CCTC(2) IT FOUND AD DOLL THAT FOUND IT STRANGE AND ALL THE MORT HAE BECAUSE IT HIS ANE CONDITION AND WHICH WAS

CCTC(3) IT FOUND A DOLL THAT FOUND IT STRANGE AND ALL THE MORT HARE BECAUSE IT HIS ANE CONDITION WHICH WAS

Truth IT SOUNDED DULL IT SOUNDED STRANGE AND ALL THE MORE SO BECAUSE OF HIS MAIN CONDITION WHICH WAS

CTC FOR AT THE PERIL OF HER OWN EXISTENCE THAND WHEN THE OTTER HALINGS HAD DESERTED HER SHE REPELLED THEN VADOR AND AFTHER OWN ACCORD GAVE LIBERTY TO ALL THE NATIONS WITHIN THE PILLARS

CCTC(1) FOR AT THE PERIL OF HER OWN EXISTENCE THAN WHEN THE OTTER HALLINGS HAD DESERTED HER SHE REPELLED THEN VADER AND OFTHER OWN ACCORD GAVE LIBERTY TO ALL THE NATIONS WITHIN THE PILLARS

CCTC(2) FOR AT THE PERIL OF HER OWN EXISTENCE THAN WHEN THE OUTTER HELLINGS HAD DESERTED HER SHE REPELLED THEN VADOR AND OF HER OWN ACCORD GAVE LIBERTY TO ALL THE NATIONS WITHIN THE PILLARS

CCTC(3) FOR AT THE PERIL OF HER OWN EXISTENCE THAND WHEN THE OTTER HELLINGS HAD DESERTED HER SHE REPELLED THEN VADOR AND OF HER OWN ACCORD GAVE LIBERTY TO ALL THE NATIONS WITHIN THE PILLARS

Truth FOR AT THE PERIL OF HER OWN EXISTENCE AND WHEN THE OTHER HELLENES HAD DESERTED HER SHE REPELLED THE INVADER AND OF HER OWN ACCORD GAVE LIBERTY TO ALL THE NATIONS WITHIN THE PILLARS

CTC HIS WIFE NOW LIES BESIDE HIM AN THE WHITE SHAFT THAT MARKS THEIR GRAVES GLEAMS ACROSS THE WHEET VIELDS

CCTC(1) HIS WIFE NOW LIES BESIDE HIM AND THE WHITE SHAFT THAT MARKS THEIR GRAVES GLEAMS ACROSS THE WHEAT VIELS

CCTC(2) HIS WIFE NOW LIES BESIDE HIM AND THE WHITE SHAFTE THAT MARKS THEIR GRAVES GLEAMS ACROSS THE WHEET VIELS

CCTC(3) HIS WIFE NOW LIES BESIDE HIM AN THE WHITE SHAFT THAT MARKS THEIR GRAVES GLEAMS ACROSS THE WHEET VIELS

Truth HIS WIFE NOW LIES BESIDE HIM AND THE WHITE SHAFT THAT MARKS THEIR GRAVES GLEAMS ACROSS THE WHEAT FIELDS

CTC BUT A WORD FURTHER CONCERNING THE EXPEDITION IN GENERAL

CCTC(1) BUT A WORD FURTHER CONCERNING THE EXPEDITION IN GENERAL

CCTC(2) BUT A WORD FURTHER CONCERNING THE EXPEDITION IN GENERAL

CCTC(3) BUT A WORD FURTHER CONCERNING THE EXPEDITION IN GENERAL

Truth BUT A WORD FURTHER CONCERNING THE EXPEDITION IN GENERAL

Thai YouTube test-th

CTC ส่วน ใหญ่ มัน จะ มี เรื่อง สำคัญ จริง จริง ครับ เพราะ ว่า การ ติดต่อ กัน มัน เป็น เรื่อง ยาก

CCTC(1) ส่วน ใหญ่ มัน จะ มี เรื่อง สำคัญ จริง จริง ครับ เพราะ ว่า การ ติดต่อ กัน มัน เป็น เรื่อง ยาก

CCTC(2) ส่วน ใหญ่ มัน จะ มี เรื่อง สำคัญ จริง จริง ครับ เพราะ ว่า การ ติดต่อ กัน มัน เป็น เรื่อง ยาก

CCTC(3) ส่วน ใหญ่ มัน จะ มี เรื่อง สำคัญ จริง จริง ครับ เพราะ ว่า การ ติดต่อ กัน มัน เป็น เรื่อง ยาก

Truth ส่วน ใหญ่ มัน จะ มี เรื่อง สำคัญ จริง จริง ครับ เพราะ ว่า การ ติดต่อ กัน มัน เป็น เรื่อง ยาก

CTC เพียงขาด

CCTC(1) เพียงขาด

CCTC(2) เที่ยงขาด

CCTC(3) เทียง ขาด

Truth ที่ ยัง ขาด

CTC จริงจริง แล้ว ก็ เรา สร้าง ทาง เลื่อง ให้ กับ ชีวิต น้อย เกิน ไป ต่างหาก

CCTC(1) จริงจริง แล้ว ก็ เรา สร้าง ทาง เลื่อก ให้ กับ ชีวิต น้อย เกิน ไป ต่างหาก

CCTC(2) จริงจริง แล้ว ก็ เรา สร้าง ทาง เลื่อง ให้ กับ ชีวิต น้อย เกิน ไป ต่างหาก

CCTC(3) จริงจริง แล้ว ก็ เรา สร้าง ทาง เลื่อง ให้ กับ ชีวิต น้อย เกิน ไป ต่างหาก

Truth จริงจริง แล้ว ก็ เรา สร้าง ทาง เรื่อง ให้ กับ ชีวิต น้อย เกิน ไป ต่างหาก

CTC ได้ รับ กฎหมาย แต่ ได้ รับ ปริญญา ทาง นั้น กฎหมาย โดย ใส่ ชุด ประ ปัญญา ตาม เพศสภาพ นะ ฮะ ซึ่ง

CCTC(1) ได้ รับ กฎหมาย แต่ ได้ รับ ปิญญา ทาง น้ำน กฎหมาย โดย ใส่ ชุด พระ ปิญญา ตาม เพศ สภาพ นะ คะ ซึ่ง

CCTC(2) ได้ รับ กฎหมาย แต่ ได้ รับ ปริญญา ทาง น้าน กฎหมาย โดย ใส่ ชุด ระบบปิญญา ตาม เพศ สภาพ นะ คะ ซึ่ง

CCTC(3) ได้ รับ กฎหมาย แต่ ได้ รับ ปิญญา ทาง น้าน กฎหมาย โดย ใส่ ชุด ประ ปิญญา ตาม เพศ สภาพ นะ คะ ซึ่ง

Truth ได้ รับ กฎหมาย ได้ รับ ปริญญา ทาง ด้าน กฎหมาย โดย ใส่ ชุด รับ ปริญญา ตาม เพศ สภาพ ซึ่ง

CTC ปา แล้ว เรา ต้อง ทิง เอา ไว้ ให้ กับ พวก เขา ะ ครับ เพราะ ฉะนั้น

CCTC(1) ป่า แลิ้ว เรา ต้อง ทิง เอา ไว้ ให้ กับ พวก เขา อ่ะ ครับ เพราะ ฉะนั้น

CCTC(2) ปา แลิ้ว เรา ต้อง ผิง เอา ไว้ ให้ กับ พวก เขา อะ ครับ เพราะ ฉะนั้น

CCTC(3) ปา แลิ้ว เรา ต้อง ทิง เอา ไว้ ให้ กับ พวก เขา อะ ครับ เพราะ ฉะนั้น

Truth ทิ้ง เอา ไว้ ให้ กับ พวก เขา ล่ะ ครับ เพราะ ฉะนั้น

Thai YouTube test-cs

CTC คุณ ตน เค มี บดไลค์ เหลอว กับ

CCTC(1) คุณ มตอน ค มี อดไลค์ เย่ยว กับ

CCTC(2) ค คุณกอน เคน มี อดไลค์ เดี้ยว กับ

CCTC(3) ค คุณ ว่า ตอน เท มี บดไลค์ เดี้ยว กิ

Truth ครับ คุณ ว่า คอนเท้นต์ นี้ คน like เยอะปะ

CTC เรา กิน แต่ ทุก อย่าง น่มัน เป็น เขา เรียก ว่า ได ทูด อ่ะ

CCTC(1) เรา กิน แต่ ทุก อย่าง มัน เขา เรียก ว่า ได ทูด อะ

CCTC(2) เรา กิน แต่ ทุก อย่าง นมัน เป็น เขา เรียก ว่า ได โทูด อ่ะ

CCTC(3) เรา กิน แต่ ทุก อย่าง น มัน เป เขา เรียก ว่า ไดพูด อ่ะ

Truth เรา กิน แต่ ทุก อย่าง เนี่ย มัน เป็น เขา เรียก ว่า dead food อะ

CTC ว เป็บ นึง นะ ครับ ที่ ชื่อ ว่า parion นั่น เอง นะ ฮะ

CCTC(1) แวเว็บ นึง นะ ครับ ที่ ชื่อ ว่า pacronn นั่น เอง นะ ฮะ

CCTC(2) แวก เว็บ นึง นะ ครับ ที่ ชื่อ ว่า pacron นั่น เอง นะ ฮะ

CCTC(3) แว แว็บ นึง นะ ครับ ที่ ชื่อ ว่า pacrirn นั่น เอง นะ ฮะ

Truth เว็บเว็บ นึง นะ ฮะ ที่ ชื่อ ว่า patreon นั้น เอง นะ ฮะ

CTC มี ฝ่าย คำ ว่า ขยัน เนี่ย พายัน ได้ ว่า เรา ใน ลาน เรา ตั้งใจ ใน สาลัม

CCTC(1) ปี ฝ่าย คำ ว่า ขยัน เนี่ย พยายัน ได้ ว่า เรา ใน ลาน เรา ตั้งใจ ใน สาลมำ

CCTC(2) อี ฝ่าย คำ ว่า ขยัน เนี่ย ขยัน ได้ ว่า เรา ลา เรา ตั้งใจ ใน สาลัมำ

CCTC(3) อี ฝ่าย คำ ว่า ขยัน เนี่ย พยาัน ได้ ว่า เรา ในลาน เรา ตั้งใจ ใน สาลัมำ

Truth defy คำ ว่า ขยัน นะ ขยัน ได้ ว่า เรา ตั้งใจ

CTC อาจ จะ ชาง การ เติม เติบโต ของ บริษัท ด้วยซ้ำ ม นะ คะ สิ่ง ที่ เรา ทำ ก็ คือ ว่า เรา จะ มี clas preaning สำหรับ

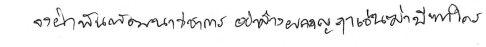
CCTC(1) อาจ จะ ช้าง การ เติม เติบโต ของ บริษัท ด้วยซ้ำ คือ มันะ คะ สิ่ง ที่ เรา ทำ ก็ คือ ว่า เรา จะ มี clas preaning สำหรับ fi

CCTC(2) อาจ จะ ช้าง การ เติม เติบโต ของ บริษัท ด้วยซ้ำ คือ มา นะ คะ สิ่ง ที่ เรา ทำ ก็ คือ ว่า เรา จะ มี crlas frenning สำหรับ f

CCTC(3) อาจ จะ ช้าง การ เติม เติบโต ของ บริษัท ด้วยซ้ำ คื้อ ม นะ คะ สิ่ง ที่ เรา ทำ ก็ คือ ว่า เรา จะ มี class freaenning สำหรับ

Truth อาจ จะ ช้า กว่า การ เติบโต ของ บริษัท ด้วยซ้ำ ค่ะ สิ่ง ที่ เรา ทำ ก็ คือ ว่า เรา จะ มี class training สำหรับ first

BEST test-seen



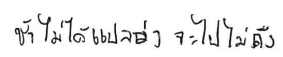
CTC จำย่าันพัฒณาวิชาการ อย่าล้างผลาญฤเช่นฆ่าบีฑาคร

CCTC(1) จำาย่าฟนพัฒนาวิธาการ อย่าฟ้างผลาญฤเขป่นฆ่าบีฑาใคร

CCTC(2) จำงย่าพันพัฒนาวิชาการ อย่าฟ้างผลาญฤเป่นฆ่าบีฑาใคร

CCTC(3) จำย่าพันพัฒนาวิอชาการ อย่าฟ้างผลาญฤาเป่นฆ่าบีฑาใคร

Truth จงฝ่าฟันพัฒนาวิชาการ อย่าล้างผลาญฤาเข่นฆ่าบีฑาใคร



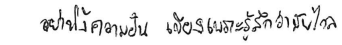
CTC ช้าไม่ได้แปลว่า จะไปไม่ถึง

CCTC(1) ช้าไม่ได้แปลว่ง จะไปไม่ถึง

CCTC(2) ช้าไม่แปลว่ง จะไปไม่ถึง

CCTC(3) ช้าไม่ได้แปลว่ง จะไปไม่ถึง

Truth ช้า ไม่ได้แปลว่า จะไปไม่ถึง



CTC อย่าปไม้ควาเฝ้น เขียงเรละรู้สึกว่ยงใกล

CCTC(1) อย่าฟไม้ควาเมฝัน เขียงเรละรู้สึกวั่นไก

CCTC(2) อย่าฟไม้ควาเฝ้น เขียงเราะรู้สึกว่าวังไก

CCTC(3) อย่าไม้ควาเมฝัน เขียงเรละรู้สึกวั่งไก

Truth อย่าทิ้งความฝัน เพียงเพราะรู้สึกว่ามันไกล

CTC พัทสุง

CCTC(1) พ้ำทลุง

CCTC(2) พั้ทลุง

CCTC(3) พ้ำลุง

Truth พัทลุง

CTC เกรดน้อย แล้วไง มาเรียนด้วยใจไม่ใช่สมอง

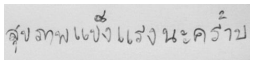
CCTC(1) เกรดน้อยแล้วได มาเรียนด้วยใจไม่ใช่สมอง

CCTC(2) เกรดน้อยแล้วไง มาเรียนด้วยใจไม่ใช่สมอง

CCTC(3) เกรดน้อยแล้วได มาเรียนด้วยใจไม่ใช่สมอง

Truth เกรดน้อยแล้วไง มาเรียนด้วยใจไม่ใช่สมอง

BEST test-unseen



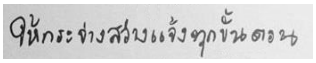
CTC สุลาพแป้งแรงนะคร้าบ

CCTC(1) สุลาพแป้งแรงนะคร้าบ

CCTC(2) สุลาพแป้งแรงนะคร้าบ

CCTC(3) สุลาพแป้งแรงนะคร้าบ

Truth สุขภาพแข็งแรงนะคร้าบ



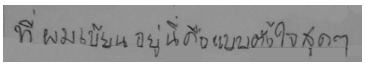
CTC กห้ทระจางสวางแจ้งตุกหันตอน

CCTC(1) วหทระจ่งสวางแจงตทุกหันตอน

CCTC(2) รหทระจ่งสวางแจ้งตทุกหันตอน

CCTC(3) รหทระจ่งสวงแจ็งทุกบันดอน

Truth ให้กระจ่างสว่างแจ้งทุกขั้นตอน



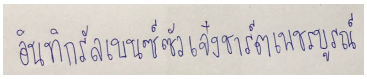
CTC พมเปียนอยู่นี่ดือแบนต์ใจสุดา

CCTC(1) ที่พมเบียนอยู่นี่ถือแบบต์ใจสุดา

CCTC(2) ที่พมเบียนอยู่นี่ถือแบบต์ใจสุดา

CCTC(3) ที่พมีเยนอยู่นี่ดือแบบต้จสุดา

Truth ที่ผมเขียนอยู่นี่คือแบบตั้งใจสุดๆ



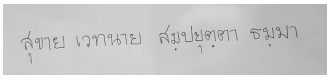
CTC อันทิกรัลเบนตซ์ตัวเจงธาริตเพชรบรูณ์

CCTC(1) อนทำกร้ลเบนซ์ตัวเจงชาริตเพชรบรูณ์

CCTC(2) น่ทำกร้ลเบนซ์ตัวเจิงธาริตเพชรบรูณ์

CCTC(3) อันทำกรัลเบนตซ์ตัวเจิงธาธริตเพชรบรูณ์

Truth อินทิกรัลเบนซ์ชัวเจ่งชาร์ตเพชรบูรณ์



CTC สุายเวทนายสมุปบุตตาธมุมา

CCTC(1) สุายเวทนายสมูปบุตุตาธมุมา

CCTC(2) สุายเวทนายสมุปบุตุตาธมูมา

CCTC(3) สุายเวทนายสมุปบุตุตาธมุมา

Truth สุขาย เวทนาย สมปยุตตา ธมมา

IAM test

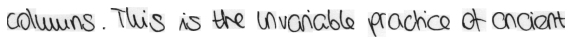
CTC I TV have postponed Malcoln Mnggeridge's " Appointment with,

CCTC(1) I TV have postponed Malcolm Muggeridge's " Appointment with

CCTC(2) I TV have postponed Malcolm Mnggeridge's " Appointment with

CCTC(3) I TV have postsoned Malcolm Mnggeridge's " Appointment with

Truth I T V have postponed Malcolm Muggeridge's " Appointment with



CTC coluuns. This is the invariable practice of oncient

CCTC(1) coluuns. This is the nvariable practice ot oncient

CCTC(2) coluuns. This is the nvariable practice of oncient

CCTC(3) coluuns. This is the invariable practice of cncient

Truth columns. This is the invariable practice of ancient

CTC froh at a certain spawning ground may

CCTC(1) fioh at a certain spawning ground may

CCTC(2) fioh at a certain spawning ground may

CCTC(3) fioh at a certain spawning groend may

Truth fish at a certain spawning ground may

CTC of blood fron the wand. Arterial diastole is

CCTC(1) of blood from the wand. Arterial diastole is

CCTC(2) of blood fron the wand. Arterial diastole is

CCTC(3) of blood fron the wond. Arterial diastole is

Truth of blood from the wound. Arterial diastole is

CTC being flung apainst the huge boulders did the rndge

CCTC(1) being flung apainst the huge boulders did the ndge

CCTC(2) being flung apainst the huge boulders did the nndge

CCTC(3) being flung apainst the huge boulders did the ndge

Truth being flung against the huge boulders did the judge

Biography

Name	Burin Naowarat
DATE OF BIRTH	November 1996
PLACE OF BIRTH	Thailand
INSTITUTIONS ATTENDED	Chulalongkorn University
HOME ADDRESS	184/11 M. 3 T. Bueng Phra Phitsanulok 65000