Implementation and Optimization of Lightweight End-to-End Speech Recognition System

Authors: Yifan Yang, Yang Yifan

Date: 2024-01-04T00:00:00+00:00

Abstract

Neural Transducer and Connectionist Temporal Classification (CTC) are popular end-to-end automatic speech recognition systems. Due to their frame-synchronous design, blank symbols are introduced to address the length mismatch between acoustic frame input sequences and output label sequences, which may introduce redundant computation. Previous research has accelerated the training and inference of Neural Transducers by discarding blank frames predicted by jointly trained CTC. However, this does not guarantee that the jointly trained CTC can maximize the proportion of blank symbols. This paper proposes two novel regularization methods that explicitly encourage CTC to label more blank symbols by constraining the self-loops of CTC non-blank symbols, enabling greater acceleration for Neural Transducers. Experiments on the LibriSpeech corpus demonstrate that the proposed method improves the inference speed of Neural Transducers by 4 times without sacrificing performance. Furthermore, greater performance improvements can be obtained when Neural Transducers are combined with external language models for decoding. Notably, the proposed regularization method enables the frame skipping rate of Neural Transducers to approach the theoretical limit, representing the first work to explore the feasibility of Neural Transducers with almost no blank symbols.

Full Text

Preamble

Title: Implementation and Optimization of Lightweight End-to-End Speech Recognition Systems
School: College of Intelligence and Computing
Major: Computer Science and Technology
Class of 2019
Advisors: Daniel Povey, Wang Longbiao

Declaration of Originality

I hereby declare that the graduation project (thesis) submitted is the result of research work conducted under the guidance of my advisor. Except for content already cited in the text, this graduation project (thesis) does not contain any research results previously published or written by others. Other individuals and groups who have contributed to the research work related to this graduation project (thesis) have been clearly acknowledged in the text. I assume legal responsibility for the originality statement of this graduation project (thesis).

Author's Signature: *{{{_}}{{}}}* **Date:** ***{{{***}}{{}}_}

I hereby declare that this graduation project (thesis) is a research achievement completed under my guidance as a student, and I have reviewed all content of the thesis.

Advisor's Signature: *{{{_}}{{}}}* **Date:** ***{{{***}}{{}}_}

Neural Transducer and Connectionist Temporal Classification (CTC) are popular end-to-end automatic speech recognition systems. Due to their frame-synchronous design, blank symbols are introduced to address the length mismatch between acoustic frame input sequences and output label sequences, which may introduce redundant computation. Previous studies accelerated the training and inference of neural Transducers by discarding blank frames predicted by a jointly trained CTC. However, this does not guarantee that the jointly trained CTC can maximize the proportion of blank symbols. This paper proposes two novel regularization methods that explicitly encourage CTC to label more blank symbols by constraining the self-loop of non-blank symbols in CTC, enabling greater acceleration for neural Transducers.

Experiments on the LibriSpeech corpus demonstrate that the proposed methods accelerate neural Transducer inference by 4 times without sacrificing performance. Furthermore, greater performance improvements are achieved when the neural Transducer is decoded with an external language model. Notably, the proposed regularization methods enable the frame reduction ratio of the neural Transducer to approach the theoretical limit. This is the first work to explore the feasibility of neural Transducers with almost no blank symbols.

Keywords: Speech Recognition, Neural Transducer, Connectionist Temporal Classification

Abstract

Neural Transducer and Connectionist Temporal Classification (CTC) are popular end-to-end automatic speech recognition systems. Due to their frame-synchronous design, blank symbols are introduced to address the length mismatch between acoustic frames and output tokens, which might bring redundant computation. Previous studies managed to accelerate the training and inference of neural Transducers by discarding frames based on the blank sym-

bols predicted by a co-trained CTC. However, there is no guarantee that the co-trained CTC can maximize the ratio of blank symbols. This paper proposes two novel regularization methods to explicitly encourage more blanks by constraining the self-loop of non-blank symbols in the CTC.

Experiments on the LibriSpeech corpus show that our proposed method accelerates the inference of neural Transducer by 4 times without sacrificing performance. It is interesting to find that the frame reduction ratio of the neural Transducer can approach the theoretical boundary. Additionally, a further gain can be observed when decoding with external language models. To the best of our knowledge, this is the first work to explore the feasibility of neural Transducers with almost no blank symbols.

1.1 Research Background and Objectives

End-to-end (E2E) architectures have gained increasing attention in the field of automatic speech recognition (ASR). In recent years, three representative architectures have been developed: Connectionist Temporal Classification (CTC) [1], Neural Transducer [2], and Attention-based Encoder-Decoder (AED) [3]. Among these three architectures, CTC and Neural Transducer share some common characteristics as they are both frame-synchronous systems, where each acoustic input frame is mapped to one or more target labels. In contrast, AED adopts label-synchronized decoding, generating a valid target label at each step. Due to its streaming processing capability and superior performance across multiple tasks, Neural Transducer has attracted growing attention from both academic research and industrial applications. However, compared to AED models, the decoding process of Neural Transducer is more computationally expensive because it needs to process the output corresponding to each acoustic frame in frame-synchronous decoding.

Since acoustic frame input sequences are typically much longer than target label sequences, a special blank symbol is introduced in frame-synchronous Neural Transducer and CTC architectures to represent “no output.” During inference, most acoustic input frames are classified as blank frames, which may lead to unnecessary computation.

1.2 Current Research Status

To accelerate decoding speed, Chen et al. [4] used efficient blank symbols and post-processing methods to transform the inference process of Hidden Markov Models and CTC models from frame-synchronous to label-synchronous, achieving significant acceleration while maintaining performance. Xu et al. [5] introduced additional blank symbols in standard Neural Transducers, referred to as big blanks, where outputting a big blank consumes two or more acoustic input frames, achieving substantial inference acceleration and slightly improved accuracy.

Furthermore, previous studies have investigated the identification of blank frames and the impact of discarding blank frames on decoding results. Chen et al. [6] studied the peaky posterior property of CTC models and found that blank frames contribute negligibly to decoding performance, and discarding blank frames does not affect the optimal decoding path. Similarly, Zhang et al. [7] discarded blank frames during Neural Transducer decoding and compressed the search space to speed up decoding. Tian et al. [8] discarded shared encoder output frames during inference based on the blank probability distribution generated by a jointly trained CTC, thereby reducing the number of encoder frames passing through the joint network. Similar to [8], Wang et al. [9] applied frame skipping techniques at intermediate layers of the shared encoder during both training and inference, achieving significant inference acceleration without affecting performance.

1.3 Main Work and Contributions

Previous work achieved inference acceleration for Neural Transducers by discarding blank frames predicted by a jointly trained CTC [8,9]. If the jointly trained CTC can accurately classify a larger proportion of acoustic input frames as blank frames, the inference speed of Neural Transducers can be further improved. To achieve this goal, this paper explores various methods to regularize the blank probabilities predicted by CTC. We explicitly encourage the CTC branch to label more blank frames by either penalizing the self-loop of non-blank symbols (Token) in the CTC topology graph with a penalty λ, or by limiting the maximum number K of consecutive repeated non-blank symbols in CTC. We demonstrate that by adjusting λ or K, the number of non-blank acoustic frames labeled by CTC can approach the number of target label sequences. Experiments on the LibriSpeech corpus show that the Neural Transducer guided by blank-regularized CTC achieves lower Word Error Rate (WER) than the baseline model without blank skipping.

In summary, the main contributions of this paper are threefold: * This paper proposes two novel regularization methods to explicitly encourage the jointly trained CTC to label more blank frames, thereby further accelerating the inference speed of Neural Transducers. * By applying the proposed strategies, the frame reduction ratio of Neural Transducers can approach the theoretical limit. * Experiments demonstrate that the proposed method improves the inference speed of Neural Transducer models by 4 times relative to standard Neural Transducer models through frame skipping techniques, without sacrificing performance.

Compared with competitive baseline models [9], this paper achieves a 1.5 times speed improvement. This paper uses the k2 [10] framework to modify CTC topology and compute loss functions. The relevant code has been released in the open-source project icefall[1].

[1]https://github.com/k2-fsa/icefall

2.1.1 Connectionist Temporal Classification (CTC)

CTC [1] is one of the earliest end-to-end speech recognition frameworks, consisting of an encoder and a linear layer as the decoder. To solve the length mismatch problem between acoustic input frame sequences and target label sequences, a blank symbol is added to the output vocabulary V, indicating no label output for that frame. Given an acoustic feature input sequence x = (x_1, ⋯, x_T) of length T, the encoder produces embedding vectors f = (f_1, ⋯, f_T). These embedding vectors then enter the CTC decoder to generate T conditionally independent posterior probability distributions p_1, ⋯, p_T corresponding to the vocabulary V . Given a ground-truth label sequence y = (y_1, ⋯, y_U) of length U, where y_u V, the CTC objective function is defined as the sum of probabilities of all possible alignments between x and y:

$$L(y) = \log p(\pi|x), \quad \pi \in \mathcal{B}^{-1}(y) \tag{2-1}$$

where $\mathcal{B}(\cdot)$ is a many-to-one mapping that removes consecutive duplicate labels and all blank symbols from alignments. Figure 2-1 [Figure 2: see original paper] illustrates the process of obtaining the target sequence from the symbol sequence predicted by CTC through mapping $\mathcal{B}(\cdot)$.

Based on the conditional independence assumption of CTC, its objective function formula (2-1) can be approximately expressed as:

$$L(y) \approx \log p(\pi_t|x), \quad \pi \in \mathcal{B}^{-1}(y) \tag{2-2}$$

Decoders based on Weighted Finite State Transducers (WFST) can efficiently implement CTC and its variants [11,12]. A typical CTC lattice contains three finite state transducers: * CTC Topology Graph (H), as an implementation of mapping $\mathcal{B}(\cdot)$; * Lexicon Graph (L), mapping sequences of lexicon units to words; * Dense Finite State Acceptor (Dense.FSA), where weights represent acoustic log probabilities.

The CTC lattice can be obtained through two steps. First, the CTC topology graph H and the lexicon graph L of the target sequence are composed into an HL transducer, which converts predicted symbol sequences into word sequences. Then, HL is intersected with Dense.FSA representing the probability distribution of CTC decoder output to generate the CTC lattice, which contains all valid alignments $\mathcal{B}^{-1}(y)$ for the target label sequence.

2.1.2 Attention-based Encoder-Decoder (AED)

AED [3] employs an attention mechanism [13] to implicitly identify and model acoustic inputs relevant to each output unit, avoiding the need for explicit alignment modeling. The AED model processes the entire acoustic input sequence at once. To explicitly indicate that the model has completed generation of all

output labels, an end-of-sentence symbol eos is added to the output vocabulary V.

As shown in Figure 2-2, the AED model consists of an encoder and an attention-based decoder. The encoder encodes the acoustic input frame sequence x = (x_1, ···, x_T) into higher-level representations h = (h_1, ···, h'_T). The decoder obtains probability distributions over the output vocabulary V eos based on the attention mechanism. Given a training sample pair (x, y), $\hat{y} = (y_1, \cdots, y_U, \langle eos \rangle)$ represents the output label sequence extended by adding the end-of-sentence symbol eos at the end. The sentence start symbol sos is used as the first input y_0 before the attention-based decoder generates any output. The AED objective function is defined as the conditional probability of $\hat{y}$:

$$L(\hat{y}) = p(\hat{y}|x) = \prod_i p(y_i|y_{i-1}, \cdots, y_0 = \langle sos \rangle, c_i) = \prod_i p(y_i|s_i, c_i), \tag{2-3}$$

where c_i represents the context vector, a linear mixture of encoder outputs h_1, ···, h'_T; s_i represents the decoder state after outputting the previous label sequence, obtained by updating the previous decoder state s_{i-1} with the previous context vector c_{i-1} and output label y_{i-1}:

$$s_i = \text{Decoder}(c_{i-1}, s_{i-1}, y_{i-1}) \tag{2-4}$$

At each time step i, the attention mechanism generates a context vector c_i containing acoustic information required to generate the next label. The attention model is content-based, matching the content of the current decoder state s_i with the content of encoder output h_u at each time step to generate an attention vector α_i, which is used to weight and fuse encoder outputs h to create c_i.

Specifically, at each decoding time step i, the attention mechanism uses vectors h_u and s_i to compute a scalar energy $e_{i,u}$ for each time step u. $e_{i,u}$ is converted to a probability distribution α_i over time steps through a Softmax function. Using α_i as weights, encoder outputs h_u at each time step are linearly mixed to create the context vector c_i:

$$e_{i,u} = \langle \text{MLP}_1(s_i), \text{MLP}_2(h_u) \rangle \tag{2-5}$$

$$\alpha_{i,u} = \frac{\exp(e_{i,u})}{\sum_u \exp(e_{i,u})} \tag{2-6}$$

$$c_i = \sum_u \alpha_{i,u} h_u, \tag{2-7}$$

where MLP_1 and MLP_2 are both Multilayer Perceptrons (MLP).

2.1.3 Neural Transducer

Neural Transducer [2] was proposed to address problems caused by the conditional independence assumption of CTC. The output probability y_u of Neural Transducer is determined conditioned on all previous labels $y_{\leq u-1}$. The decoder functions similarly to a language model, always receiving previously output labels as input. The joint network defines the probability $p(k|t,u)$ of outputting label k at time t after outputting u-1 previous labels by fusing acoustic embeddings and text embeddings. Similar to CTC, Neural Transducer also defines the objective function as the sum of probabilities over all possible alignments:

$$L(y) = \log p(\pi|x), \quad \pi \in \mathcal{A}^{-1}(y) \tag{2-8}$$

where $\mathcal{A}(\cdot)$ is a many-to-one mapping that removes blank symbols from alignments.

Notably, blank symbols in CTC and Neural Transducer have very similar functions in separating consecutive symbols and aligning acoustic frame input sequences with target label sequences. However, there are subtle differences between these two types of blank symbols. Specifically, when processing consecutive frames with similar acoustic information, CTC alignments allow one target label to correspond to multiple repeated symbols in the output sequence, where two consecutive identical labels require at least one blank symbol to separate them. In Neural Transducer, each target label corresponds to only one symbol in the output sequence, with all other output symbols in the alignment being blank symbols. It is reasonable to believe that the blank symbol predictions of these two models are highly correlated and have good consistency.

2.2 Mainstream Open-Source Toolkits for End-to-End Speech Recognition

As end-to-end automatic speech recognition systems have attracted increasing research interest, various powerful open-source toolkits have been developed to popularize the use of E2E ASR models, such as icefall based on K2 [10], Kaldi [14], ESPnet [15], WeNet [16,17], and Fairseq [18].

Since this paper is implemented based on the k2 framework, only k2 is discussed here.

2.2.1 Overview of k2

K2 integrates Finite State Automaton (FSA) and Finite State Transducer (FST) algorithms into autograd-based machine learning toolkits such as PyTorch. K2 supports both CPU and CUDA, and can process batches of FSTs simultaneously. K2 can be used to compute CTC loss functions, Lattice-free Maximum Mutual Information (LF-MMI) [19] loss functions, and for ASR decoding. FSA/FST in k2 have the following characteristics: * Only one start state; * Only one final

state; * The start state is numbered 0; * All other states have numbers greater than 0; * The final state has the largest number; * Arcs entering the final state must have -1 as the input label and a score of 0, and if there is an output label it must also be -1; * Arcs not entering the final state cannot have -1 as the input label, and if there is an output label it cannot be -1 either; * States do not have scores; * All scores are on arcs; * Scores represent log values of probabilities.

2.2.2 Implementing CTC-Related Algorithms with k2

This section uses the target label sequence "AB" as an example to demonstrate the process of building a CTC Lattice_{AB} through code and visualizing weighted finite state transducers, as well as the forward and backward propagation processes.

In this example, the vocabulary is defined as V_{AB} = ["A", "B"], and the blank symbol "blk" is introduced. The code to create the CTC topology graph is as follows, and the created H_{AB} is shown in Figure 2-3 [Figure 2: see original paper].

```
import k2

isym = k2.SymbolTable.from_{str}("""
blk 0
A 1
B 2
""")

osym = k2.SymbolTable.from_{str}("""
A 1
B 2
""")
```

Assuming the modeling unit is letters, the target label sequence "AB" is encoded as "A" and "B". The code to create the lexicon graph is as follows, and the created L_{AB} is shown in Figure 2-4 [Figure 2: see original paper].

By composing H_{AB} and L_{AB}, we can obtain HL_{AB}, which converts predicted symbol sequences into word sequences and contains all paths that can be converted to the target sequence "AB". The code is as follows, and the created HL_{AB} is shown in Figure 2-5 [Figure 2: see original paper].

Assuming the CTC decoder output includes three frames with predicted probability distributions as shown in Table 2-1 , the code to create the corresponding dense finite state acceptor is as follows, and the created Dense.FSA_{AB} is shown in Figure 2-6 [Figure 2: see original paper].

```
import torch

nnet_{output} = torch.log(torch.tensor(
```

```
[[[0.6,0.3,0.1],[0.25,0.6,0.15],[0.25,0.15,0.6]]],
dtype=torch.float32))
nnet_{output}.requires_{grad}(True)
supervision_{segments} = torch.tensor([[0, 0, 3]]).to(torch.int32)
dense_{fsa} = k2.DenseFsaVec(
torch.log(nnet_{output}), supervision_{segments}, allow_{truncate}=0)
dense_{{{fsa}}_{{vec}}} = k2.convert_{{{dense}}_{{to}}}_{{{fsa}}_{{vec}}}(dense_{fsa})
dense_{{{fsa}}_{{vec}}}[0].draw("dense_{fsa}.pdf")
```

Table 2-1 Probability distribution output by CTC decoder

frame	blk	A	B
0	log(0.60) = -0.5108	log(0.30) = -1.2040	log(0.10) = -2.3026
1	log(0.25) = -1.3863	log(0.60) = -0.5108	log(0.15) = -1.8971
2	log(0.25) = -1.3863	log(0.15) = -1.8971	log(0.60) = -0.5108

By intersecting HL_{AB} with Dense.FSA_{AB}, the CTC Lattice_{AB} can be generated, which contains all valid alignments $\mathcal{B}^{-1}(y)$ for the target label sequence "AB". The code is as follows, and the created CTC Lattice_{AB} is shown in Figure 2-7 [Figure 2: see original paper].

```
ctc_{lattice} = k2.intersect_{dense}(
dense_{fsa}, HL_{AB}, 20.0, seqframe_{{{idx}}_{{name}}}="seqframe_{idx}")
ctc_{lattice}[0].draw("ctc_{lattice}.pdf")
```

Unlike traditional CTC which uses the forward-backward algorithm [1], this paper uses a differentiable dynamic programming method[1] to compute the total score of the CTC Lattice and optimize the CTC objective function formula (2-2).

```
ctc_{{{object}}_{{scores}}} = ctc_{lattice}.get_{{{tot}}_{{scores}}}(
log_{semiring}=True, use_{{{double}}_{{scores}}}=True)
print(ctc_{{{object}}_{{scores}}})
# tensor([-0.8983], dtype=torch.float64)
```

In this example, using the log semiring, the mathematical derivation of the forward propagation process is as follows:

$$
\begin{aligned}
s_1 &= \text{arc}_{01} = -0.5108 \\
s_2 &= \text{arc}_{02} = -1.2040 \\
s_3 &= \log(e^{s_1+\text{arc}_{13}} + e^{s_2+\text{arc}_{23}}) = -0.6162 \\
s_4 &= s_2 + \text{arc}_{24} = -2.5903 \\
s_5 &= s_2 + \text{arc}_{25} = -3.1011 \\
s_6 &= \log(e^{s_3+\text{arc}_{36}} + e^{s_4+\text{arc}_{46}} + e^{s_5+\text{arc}_{56}}) = -0.9263 \\
s_7 &= s_5 + \text{arc}_{57} = -4.4874 \\
s_8 &= \log(e^{s_6+\text{arc}_{68}} + e^{s_7+\text{arc}_{78}}) = -0.8983
\end{aligned}
$$

where s_8 at the final state equals $ctc\_object\_scores = -0.8983$, which is the value of the CTC objective function formula (2-2) in this example—the sum of probabilities of all valid alignments $\mathcal{B}^{-1}(y)$ for the target label sequence "AB".

Most operations in k2 support automatic differentiation. The code for backward propagation to compute gradients is as follows:

```
ctc_{{{object}}_{{scores}}}.backward()
print(nnet_{output}.grad)
# tensor([[[0.5304, 0.4696, 0.0000],
# [0.1105, 0.7956, 0.0939],
# [0.0276, 0.0000, 0.9724]]])
```

The mathematical derivation of the backward propagation process is as follows:

[The detailed gradient calculations from the original text are preserved here with proper formatting]

The correspondence between CTC Lattice_{AB} and CTC decoder output is shown in Figure 2-8 [Figure 2: see original paper]. The *nnet_output.grad* represents the gradients of $p_{00}, p_{01}, \cdots, p_{22}$, which correspond to the results of the backward propagation derivation above.

2.3 Language Model Integration for End-to-End Speech Recognition Systems

Traditional hybrid systems explicitly learn an Acoustic Model (AM) and directly integrate an External Language Model (ELM) through Bayes' rule:

$$\hat{p}(y|x) = p_{AM}(x|y)p_{ELM}(y) \tag{2-9}$$

In end-to-end systems, the posterior probability $p(y|x)$ is jointly learned through a unified model, where the implicit acoustic model and language model are highly correlated. This makes language model integration in end-to-end models less straightforward than in hybrid models.

2.3.1 Shallow Fusion (SF)

Shallow Fusion [20–22] uses a Recurrent Neural Network Language Model (RNNLM) trained on additional text data to improve language modeling capability. During decoding, the non-blank posterior probabilities of the Neural Transducer and RNN language model are added in the log domain:

$$\log \hat{p}(y|x) = \log p_{RNN-T}(y|x) + \lambda \log p_{ELM}(y), \tag{2-10}$$

where $\hat{p}(y|x)$ is the actual posterior probability used in decoding, $p_{RNN-T}(y|x)$ is the posterior probability of y given x in the Neural Transducer, and $p_{ELM}(y)$ is the probability of y generated by the RNN language model.

2.3.2 Internal Language Model Estimation (ILME)

The language model information captured by the Neural Transducer model from transcribed text during training is considered to be contained in a so-called Internal Language Model (ILM). If the internal language model of the Neural Transducer can be estimated in some way, language model integration for Neural Transducer models can be achieved by first subtracting the posterior probability of the internal language model $p_{ILM}(y)$ and then adding the posterior probability of the external language model $p_{ELM}(y)$:

$$\hat{p}(y|x) = \frac{p_{RNN-T}(x|y)}{p_{ILM}(y)} p_{ELM}(y) \tag{2-11}$$

Meng et al. [23] estimated the internal language model of the Neural Transducer model by zeroing out acoustic hidden states and normalizing non-blank labels in the joint network output.

2.3.3 Density Ratio (DR)

Although shallow fusion is widely used in practice, it simply performs log-linear interpolation of probabilities from the speech recognition model and RNN language model during decoding, which does not strictly follow Bayes' theorem. To address this, Density Ratio [24] was proposed as an extension of shallow fusion, making the following assumptions: * There exists some true joint distribution $p_S(y, x)$ about text and audio in the source domain (the domain where the Neural Transducer is trained); * There exists another true joint distribution $p_T(y, x)$ about text and audio in the target domain; * The end-to-end model for the source domain can reasonably model $p_S(y|x)$; * Independently trained language models can reasonably model $p_S(y)$ and $p_T(y)$ respectively; * The source and target domains are acoustically consistent, i.e., $p_S(y|x) \approx p_T(y|x)$; * The posterior distribution of the target domain $p_T(y|x)$ is unknown.

Based on the above assumptions, the target domain posterior can be estimated as:

$$\widehat{p}_T(y|x) = \frac{p_S(x)}{p_T(x)} \frac{p_T(y)}{p_S(y)} p_S(y|x) \tag{2-12}$$

Formula (2-12) enables estimation of the pseudo-posterior for Neural Transducers:

$$\log \widehat{p}(y|x) = \log p_{RNN-T}(y|x) + \lambda_1 \log p_{ILM}(y) + \lambda_2 \log p_{ELM}(y), \tag{2-13}$$

where λ_1 represents the source domain language model weight and λ_2 represents the target domain language model weight.

2.3.4 Low-order Density Ratio (LODR)

Some studies [25,26] have shown that Neural Transducers only learn low-order language model information, while traditional DR uses RNNLMs trained with full context, which may not be suitable for ILM estimation and could lead to degraded language model integration performance. Based on the DR method, Zheng et al. [27] proposed a Low-order Density Ratio (LODR) method that estimates linguistic data in the source domain using low-order weak language models.

3.1 CTC-Guided Neural Transducer

Inspired by the observation that blank symbols play similar roles in CTC and Neural Transducer, some previous studies have attempted to use blank symbols in CTC to guide and simplify the inference process of Neural Transducer systems.

Tian et al. [8] proposed a method to discard shared encoder output frames based on blank probabilities predicted by the CTC branch. By reducing the number of frames from encoder output entering the joint network, decoding speed can be greatly improved. However, frame skipping was not performed during training. The inconsistent handling of blank frames between training and inference led to suboptimal performance. To achieve consistent frame skipping behavior during both training and inference, Wang et al. [9] performed frame skipping at intermediate layers of the shared encoder based on blank probabilities predicted by the jointly trained CTC model. During forward propagation, the CTC branch computes blank probabilities, and frames with blank probabilities higher than a predetermined threshold do not participate in RNN-T loss computation. The authors reported that this method achieved significant acceleration for Neural Transducer models in both training and inference without degrading performance.

3.2 Blank Regularization Strategies

Existing methods [8,9] improve Neural Transducer inference speed by utilizing jointly trained CTC to skip blank frames. However, few studies have explored the feasibility of skipping non-blank frames. According to the definition of $\mathcal{B}(\cdot)$ in formula (2-1), CTC models not only delete all blank symbols but also merge consecutive duplicate symbols. If the CTC branch can treat frames corresponding to consecutive repeated non-blank symbols in the output sequence as blank frames, then more shared encoder output frames can be discarded during Neural Transducer decoding, further accelerating inference. Inspired by this observation, this paper proposes two strategies to reduce the peakiness of non-blank symbol posterior distributions in CTC and force the CTC model to output fewer consecutive repeated non-blank symbols.

3.2.1 Soft Restriction

The first proposed strategy introduces a fixed penalty term λ (e.g., 0.05) on all non-blank self-loops in the standard CTC HL transducer during training, as shown in Figure 3.1a [Figure 3: see original paper]. This means that alignments containing more consecutive repeated non-blank symbols will receive larger penalties, as shown in Figure 3.1b [Figure 3: see original paper], making the CTC model tend to output alignments with fewer consecutive repeated non-blank symbols. Notably, this non-blank self-loop penalty term λ is only added during training. By adjusting the magnitude of penalty term λ, the proportion of blank frames output by the CTC branch can be controlled. This strategy is called soft restriction.

3.2.2 Hard Restriction

In soft restriction, the CTC branch produces a larger proportion of blank frames by penalizing CTC alignments with consecutive repeated non-blank symbols. However, alignments with different numbers of repeated non-blank symbols still participate in optimization during training because, by definition, these are all valid alignments. To address this, another strategy called hard restriction is proposed, which explicitly limits the maximum number K of consecutive repeated non-blank symbols (including the first non-blank symbol) during training. Figures 3.1c [Figure 3: see original paper] and 3.1d [Figure 3: see original paper] show HL transducers when $K = 1$ and $K = 2$, respectively. It can be seen that alignments exceeding K consecutive repeated non-blank symbols are pruned in the corresponding HL transducer.

3.2.3 Implementing Blank-Regularized CTC with k2

This section also uses the target label sequence "AB" as an example to demonstrate the process of building HL transducers for blank-regularized CTC through code and visualizing weighted finite state transducers.

In this example, the vocabulary is defined as V_{AB} = [“A”, “B”], and the blank symbol “blk” is introduced. The code to directly create the standard HL transducer is as follows, and the created HL_{AB} is shown in Figure 3-2 [Figure 3: see original paper].

```
import k2

isym = k2.SymbolTable.from_{str}("""
blk 0
A 1
B 2
""")

osym = k2.SymbolTable.from_{str}("""
A 1
B 2
""")
```

By applying a fixed penalty term $\lambda = 0.05$ to all non-blank self-loops (i.e., arc_{11}, arc_{33}) in HL_{AB}, the soft-restricted HL transducer can be obtained. The code to create the soft-restricted HL transducer with $\lambda = 0.05$ is as follows, and the created HL^{λ=0.05}_{AB} is shown in Figure 3-3 [Figure 3: see original paper].

```
HL_{soft} = k2.ctc_{graph}([[1,2]], modified=False)
all_{{{self}}_{{blanks}}}_{idx} = (HL_{soft}.arcs.values()[:, 0] == HL_{soft}.arcs.values()
blank_{{{self}}_{{loops}}}_{idx} = HL_{soft}.arcs.values()[:, 2] == 0
HL_{soft}.draw("HL_{soft}.pdf")
```

By limiting the maximum number of consecutive repeated non-blank symbols to K = 2 in HL_{AB}, the hard-restricted HL transducer can be obtained. The code to create the hard-restricted HL transducer with K = 2 is as follows, and the created HL^{K=2}_{AB} is shown in Figure 3-4 [Figure 3: see original paper].

```
HL_{hard} = k2.fast_{{{ctc}}_{{graph}}}(
[[1,2]], modified=False, max_{repeat}=2)
HL_{hard}.draw("HL_{hard2}.pdf")
```

By limiting the maximum number of consecutive repeated non-blank symbols to K = 1 in HL_{AB}, the hard-restricted HL transducer can be obtained. The code to create the hard-restricted HL transducer with K = 1 is as follows, and the created HL^{K=1}_{AB} is shown in Figure 3-5 [Figure 3: see original paper].

```
HL_{hard} = k2.fast_{{{ctc}}_{{graph}}}(
[[1,2]], modified=False, max_{repeat}=1)
HL_{hard}.draw("HL_{hard1}.pdf")
```

3.3 Frame Skipping

Figure 3-6 [Figure 3: see original paper] Schematic diagram of frame skipping

During training, encoder output frames are filtered based on the blank probability $p_t^{\emptyset}$ predicted by the jointly trained CTC branch. If $p_t^{\emptyset}$ exceeds a preset threshold β (e.g., 0.85), frame t is classified as a blank frame and is discarded during RNN-T loss computation. The frame skipping process is shown in Figure 3-6 [Figure 3: see original paper]. The set of discarded encoder output frames can be described as:

$$f_{skipped} = \{f_t | p_t^{\emptyset} > \beta\} \tag{3-1}$$

During inference, a trade-off between frame skipping rate and recognition accuracy can be achieved by adjusting the threshold β for skipping blank frames.

4.1 Experimental Design

Dataset
The LibriSpeech [28] corpus is used in experiments, containing 960 hours of audiobook recordings with transcripts. Lhotse [29] is used for data preparation. Speed perturbation [30] with factors of 0.9 and 1.1 is applied to training data for data augmentation. SpecAugment [31] and noise augmentation [32] are used during training to improve model robustness. Input features are 80-channel filterbank features (FBank) extracted with a window size of 25ms and frame shift of 10ms. Modeling units use 500-class Byte Pair Encoding (BPE) [33] word pieces.

System Architecture
As shown in Figure 4-1 [Figure 4: see original paper], a jointly trained CTC and Neural Transducer model framework is adopted. The shared encoder is a 12-layer Conformer [34] with dimension 512. The Neural Transducer decoder is a stateless decoder [26] with dimension 512 and context length 2. A convolutional downsampling module with stride 4 before the encoder reduces the frame rate to 25 Hz. The model has a total of 78.9M parameters.

Training
The loss function is the sum of 0.2 times the CTC loss [1] and the pruned RNN-T loss [35]. Considering that the CTC branch's prediction of blank symbols is inaccurate in early stages, the frame skipping mechanism is only used after 4000 update steps. This paper designs a series of experiments in the following sections to explore the impact of hyperparameters (β, β , λ, K). All models are trained using 4 NVIDIA V100 GPUs.

Evaluation
Performance is evaluated on the LibriSpeech test sets (test-clean and test-other). In addition to Word Error Rate (WER), frame reduction ratio (defined as

$|f_{skipped}|/T$) and Real Time Factor (RTF) are measured as inference speed metrics. Furthermore, to evaluate the ability of language models to combine with Neural Transducers using frame skipping technology, WER with external language model (LM) decoding is also reported. This paper investigates language model integration using both shallow fusion and low-order density ratio. The external language model consists of three Long Short-Term Memory (LSTM) [36] layers trained on the LibriSpeech language model corpus. The low-order n-gram language model in low-order density ratio is a bi-gram trained on the 960-hour transcript text of LibriSpeech. The ratios of LSTM language model and bi-gram language model during decoding are tuned through grid search on the LibriSpeech development sets (dev-clean and dev-other).

4.2.1 Impact of Blank Regularization Strategies on Performance

Figure 4-2 [Figure 4: see original paper] shows the blank labeling of the same acoustic input frames by blank-regularized CTC under different strategies. From the continuity of white portions, it can be seen that the proposed regularization methods effectively encourage CTC to label non-blank repeated frames as blank frames, making the overall proportion of dark portions larger, which means fewer frames are involved in Neural Transducer loss computation.

Figure 4-2 [Figure 4: see original paper] Comparison of blank labeling by blank-regularized CTC for the same acoustic input frames under different strategies. Dark portions represent blank frames, white portions represent non-blank frames. 0 represents CTC without regularization, 1 represents CTC with soft restriction regularization, and 2 represents CTC with hard restriction regularization.

Results of applying different strategies to regularize blank symbols are shown in Table 4-1 . The baseline model is a Neural Transducer and CTC joint training system without frame skipping technology, where CTC loss serves as an auxiliary loss with weight 0.2. For comparison with existing frame skipping methods that do not modify CTC topology, Table 4-1 lists three models using different preset thresholds for frame skipping during training and inference, referred to as Threshold. Table 4-1 also shows the average frame reduction ratios over test sets. As a reference, we calculate the theoretically maximum possible frame reduction ratio $\gamma_{max} = 1 - \frac{S}{T} = 78.61\%$, where S is the total length of output symbol sequences and T is the total length of acoustic input frame sequences in the test set.

The following observations can be made: * Larger frame reduction ratios can be achieved after applying soft or hard restrictions on the CTC topology compared to existing methods, indicating that a larger proportion of frames are recognized as blank frames by the CTC head. * Trade-offs between WERs and RTF can be tuned by adjusting λ or K. A larger penalty λ or a smaller K encourages the shared encoder to better discriminate between blank and non-blank frames,

leading to more confident predictions for blank frames. By further increasing λ to 5 or reducing K to 1, the frame reduction ratio approaches γ_{max} while maintaining reasonable accuracy. * The proposed blank regularization method achieves 4 times speedup while yielding even slightly lower WERs than the baseline model without blank skipping. This indicates that consecutively repeated non-blank frames contribute little to the decoding results with proper regularization during training, and discarding them during inference does not affect WERs.

Figure 4-3 [Figure 4: see original paper] visualizes the relationship between frame reduction ratio and aggregated WER, which is the summation of WER on test-clean and test-other. The aggregated WER of the baseline model (horizontal loosely dashed line) and γ_{max} (vertical densely dashed line) are also plotted for reference. Data points for each configuration are collected by varying the decoding blank threshold β (values: 0.8, 0.85, 0.9, 0.95, 0.99, 0.999). As can be seen, the trade-off between aggregated WER and frame reduction ratio can be achieved by tuning β during decoding. A smaller β results in a larger proportion of blank frames while having higher WERs. Our proposed method achieves a much larger frame reduction ratio than existing methods (Threshold-x in Figure 4-3) without sacrificing accuracy. By tuning λ or K, the Neural Transducer with blank-regularized CTC even outperforms the baseline without blank skipping while achieving over 75% frame reduction ratio.

4.2.2 Impact of Blank Regularization Strategies on External Language Model Integration

Table 4-2 shows WERs with external language model decoding and the relative performance improvement percentages compared to baseline models without external language models from Table 4-1.

Compared to the baseline model, blank-regularized Neural Transducer models achieve larger relative WER reduction when combined with external language models, regardless of whether shallow fusion or low-order density ratio is used. This suggests that discarding more blank frames enables better fusion with external language models.

Conclusions and Future Work

Inspired by the observation that blank symbols play similar roles in CTC and Neural Transducer, this paper proposes two novel blank regularization methods to further increase the proportion of predicted blank symbols for the co-trained CTC model. By discarding blank frames before going through the joint network, the blank-regularized CTC can greatly accelerate the inference of Neural Transducer. Notably, the proposed regularization methods enable the frame reduction ratio of Neural Transducer to approach the theoretical boundary. Experiments show that the Neural Transducer guided by blank-regularized CTC achieves 4 times speedup during inference without sacrificing performance. Our

approaches achieve better trade-offs between WER and inference real-time factor than existing methods. Additionally, a further gain can be observed when decoding with external language models.

This work reveals the feasibility of Neural Transducers with almost no blank symbols. Future work will attempt to redefine the blank symbols of Neural Transducers to achieve a more efficient end-to-end autoregressive framework.

References

[1] Graves A, Fernández S, Gomez F J, et al. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks [C]. In Proc. ICML, Pittsburgh, 2006.
[2] Graves A, Mohamed A, Hinton G E. Speech recognition with deep recurrent neural networks [C]. In Proc. ICASSP, Vancouver, 2013.
[3] Chan W, Jaitly N, Le Q, et al. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition [C]. In Proc. ICASSP, Shanghai, 2016.
[4] Chen Z, Zheng W, You Y, et al. Label-synchronous decoding algorithm and its application in speech recognition [J]. Chinese Journal of Computers, 2019, 42 (1511-1523).
[5] Xu H, Jia F, Majumdar S, et al. Multi-blank Transducers for Speech Recognition [C]. In Proc. ICASSP, Rhode Island, 2023.
[6] Chen Z, Deng W, Xu T, et al. Phone Synchronous Decoding with CTC Lattice [C]. In Proc. Interspeech, San Francisco, 2016.
[7] Zhang Y, Sun S, Ma L. Tiny Transducer: A Highly-Efficient Speech Recognition Model on Edge Devices [C]. In Proc. ICASSP, Toronto, 2021.
[8] Tian Z, Yi J, Bai Y, et al. FSR: Accelerating the Inference Process of Transducer-Based Models by Applying Fast-Skip Regularization [C]. In Proc. Interspeech, Brno, 2021.
[9] Wang Y, Chen Z, Zheng C, et al. Accelerating RNN-T Training and Inference Using CTC guidance [C]. In arXiv preprint arXiv:2210.16481, 2022.
[10] Povey D, Zelasko P, Khudanpur S. Speech recognition with next-generation kaldi (k2, lhotse, icefall) [C]. In Interspeech: tutorials, 2021.
[11] Miao Y, Gowayyed M, Metze F. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding [C]. In Proc. ASRU, Scottsdale, 2015.
[12] Laptev A, Majumdar S, Ginsburg B. CTC Variations Through New WFST Topologies [C]. In Proc. Interspeech, Incheon, 2022.
[13] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need [C]. In Proc. NIPS, Long Beach, 2017.
[14] Povey D, Ghoshal A, Boulianne G, et al. The Kaldi speech recognition toolkit [C]. In Proc. ASRU, Waikoloa, 2011.
[15] Watanabe S, Hori T, Karita S, et al. Espnet: End-to-end speech processing toolkit [C]. In arXiv preprint arXiv:1804.00015, 2018.
[16] Yao Z, Wu D, Wang X, et al. WeNet: Production oriented Streaming

and Non-streaming End-to-End Speech Recognition Toolkit [C]. In Proc. Interspeech, Brno, 2021.
[17] Zhang B, Wu D, Peng Z, et al. WeNet 2.0: More Productive End-to-End Speech Recognition Toolkit [C]. In Proc. Interspeech, Incheon, 2022.
[18] Ott M, Edunov S, Baevski A, et al. fairseq: A Fast, Extensible Toolkit for Sequence Modeling [C]. In Proc. NAACL, Minneapolis, 2019.
[19] Hadian H, Sameti H, Povey D, et al. End-to-end Speech Recognition Using Lattice-free MMI. [C]. In Proc. Interspeech, Hyderabad, 2018.
[20] Mikolov T, Karafiát M, Burget L, et al. Recurrent neural network based language model [C]. In Proc. Interspeech, Chiba, 2010.
[21] Chorowski J, Jaitly N. Towards better decoding and language model integration in sequence to sequence models [C]. In Proc. Interspeech, Stockholm, 2017.
[22] Kannan A, Wu Y, Nguyen P, et al. An Analysis of Incorporating an External Language Model into a Sequence-to-Sequence Model [C]. In Proc. ICASSP, Calgary, 2018.
[23] Meng Z, Kanda N, Gaur Y, et al. Internal language model training for domain-adaptive end-to-end speech recognition [C]. In Proc. ICASSP, Toronto, 2021.
[24] McDermott E, Sak H, Variani E. A density ratio approach to language model fusion in end-to-end automatic speech recognition [C]. In Proc. ASRU, Singapore, 2019.
[25] Variani E, Rybach D, Allauzen C, et al. Hybrid autoregressive transducer (hat) [C]. In Proc. ICASSP, Barcelona, 2020.
[26] Ghodsi M, Liu X, Apfel J, et al. RNN-transducer with stateless prediction network [C]. In Proc. ICASSP, Barcelona, 2020.
[27] Zheng H, An K, Ou Z, et al. An Empirical Study of Language Model Integration for Transducer based Speech Recognition [C]. In Proc. Interspeech, Incheon, 2022.
[28] Panayotov V, Chen G, Povey D, et al. Librispeech: an asr corpus based on public domain audio books [C]. In Proc. ICASSP, South Brisbane, 2015.
[29] Zelasko P, Povey D, Trmal J Y, et al. Lhotse: a speech data representation library for the modern deep learning ecosystem [C]. In arXiv preprint arXiv:2110.12561, 2021.
[30] Ko T, Peddinti V, Povey D, et al. Audio augmentation for speech recognition [C]. In Proc. Interspeech, Dresden, 2015.
[31] Park D S, Chan W, Zhang Y, et al. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition [C]. In Proc. Interspeech, Graz, 2019.
[32] Snyder D, Chen G, Povey D. Musan: A music, speech, and noise corpus [C]. In arXiv preprint arXiv:1510.08484, 2015.
[33] Sennrich R, Haddow B, Birch A. Neural Machine Translation of Rare Words with Subword Units [C]. In Proc. ACL, Berlin, 2016.
[34] Gulati A, Qin J, Chiu C, et al. Conformer: Convolution-augmented Transformer for Speech Recognition [C]. In Proc. Interspeech, Shanghai, 2020.
[35] Kuang F, Guo L, Kang W, et al. Pruned RNN-T for fast, memory-efficient

ASR training [C]. In Proc. Interspeech, Incheon, 2022.
[36] Hochreiter S, Schmidhuber J. Long short-term memory [J]. Neural computation, 1997, 9.

Acknowledgments

As this thesis comes to a close, so does my undergraduate studies. Beginning in 2019 and ending in 2023, I started in telecommunications and later switched to computer science, dabbling in mathematics but only scratching the surface. I am grateful to my teachers and seniors for their generous guidance—though the peach and plum trees do not speak, a path forms beneath them. I am deeply thankful for everyone's support.

Here, I thank my thesis advisor Wang Longbiao, and teachers Liu Xueli and Wang Xiaobao for their revision suggestions. I thank teachers Li Keqiu, Wei Jizeng, Feng Wei, Qu Wenyue, Wan Liang, Lin Di, Shang Fanhua, Xiong Deyi, Wu Hutong, Xu Guangquan, Wang Li, Wang Bo, Wang Qilong, Chen Shizhan, Chen Junjie, Bi Zhongke, Hao Jianye, Zhang Yi, Li Gang, Li Sen, Li Chun, Li Youmeng, Zhang Xiaowang, Jin Di, Liu Shuang, Liu Jian, Sun Yueheng, Qu Ri, Su Ran, Rao Guozheng, and Tang Shanjiang from the College of Intelligence and Computing. I thank my PhD advisors Chen Xie and Yu Kai. I thank my internship mentors at Xiaomi, Daniel Povey and Kuang Fangjun, and my colleagues Yang Xiaoyu, Guo Liyong, Yao Zengwei, Kang Wei, and Lin Long. Finally, I thank my beloved Yuan Fang.

Publications During Bachelor's Studies

Published Papers: [1] Yang Y, Yang X, Guo L, et al. Blank-regularized CTC for Frame Skipping in Neural Transducer [C]. In Proc. Interspeech, Dublin, 2023. (CCF C, First author, accepted May 2023)
[2] Yao Z, Kang Wei, Kuang F, et al. Delay-penalized CTC implemented based on Finite State Transducer [C]. In Proc. Interspeech, Dublin, 2023. (CCF C, Sixth author, accepted May 2023)

Note: Figure translations are in progress. See original paper for figures.