修 士 論 文 の 和 文 要 旨

研究科・専攻	大学院　情報理工学研究科　情報・ネットワーク工学専攻　博士前期課程		
氏　　名	Teo Wen Shen	学籍番号	2131178
論 文 題 目	Compressing Long Acoustic Vectors to Short Acoustic Tokens by Incorporating Continuous Integrate-and-Fire in RNN-Transducers for Automatic Speech Recognition (長い音響系列を短い音響トークンに圧縮する Continuous Integration-and-Fire を RNN-Transducer に組み込む音声認識)		

要　　旨

本論文では、Continuous Integrate-and-Fire (CIF) フレームワークを用いて音響圧縮する音声認識の研究を行った。CIF は長い音響系列を短く整列させ、適切なサイズに圧縮する能力を持つ。具体的には、出力記号の数に対して音響系列の長さを削減することを目指し、RNN-Transducer（RNN-T）モデルに CIF を組み込み、トレーニングメモリの使用量とデコード時間を大幅に削減し、性能の低下を最小限に抑える音声認識を実現した。この手法には、新しい attention-based のダウンサンプリングサブモジュールや効果的な重み予測サブモジュールなども加えた。さらに、パラメーターを追加せずに、かつ、デコードステップを増やさずに、CIF の性能を向上させるための摂動戦略を実施し、その結果も報告する。これに加え，異なる CIF 構築とデータセットの組み合わせに対する包括的な実験も行いその結果も報告する。

令和 5 年度 修士論文

Compressing Long Acoustic Vectors to Short Acoustic Tokens by Incorporating Continuous Integrate-and-Fire in RNN-Transducers for Automatic Speech Recognition

指導教員　南 泰浩　教授

電気通信大学情報理工学研究科

情報・ネットワーク工学専攻

学籍番号 2131178

Teo Wen Shen

Contents

Chapter 1

Introduction

1.1 Background

The Continuous Integrate-and-Fire (CIF) framework [1] has garnered significant attention in various speech-related applications due to its capability to align and compress longer sequences into shorter ones. Briefly, it predicts a weight for each element in the input sequence, and, for each group of elements whose weights sum to a hyperparameter threshold, it integrates and fires them into one output element. CIF has found applications in speaker change detection [2] and simultaneous speech translation [3, 4]. Moreover, within the field of automatic speech recognition (ASR) where CIF was first implemented, it has been utilized as an optimization technique for ASR training [5] and as a means to link acoustic and linguistic representations [6, 7, 8].

However, when CIF is used in its original form as a sequence-to-sequence (Seq2Seq) training objective, its per-token Cross-Entropy (CE) loss falls short compared to other Seq2Seq losses, such as the Connectionist Temporal Classification (CTC) [9] loss and Recurrent Neural Network Transducer (RNN-T) loss. This performance gap can be attributed, in part, to the aggressive downsampling in CIF, which allows for faster decoding but may lead to degradation in performance, as noted in [10]'s findings.

On the other hand, RNN-Transducer (RNN-T) models [11] have emerged as powerful Seq2Seq architectures for streaming ASR. Unlike the conventional Connectionist Temporal Classification (CTC) framework [9], RNN-Ts can learn dependencies between output symbols and sequentially produce multiple symbols given an input acoustic vector, accomplished by considering the probability distribution over sequences of all alignments and all lengths. However, the computational demands of RNN-T models arise from their 4-dimensional loss matrix (B, T, U, V) per batch, where B represents

the batch size, T the length of the input acoustic vector, U the length of the output symbols, and V the vocabulary size. Consequently, training RNN-T models requires immense memory capacity.

In practice, the number of acoustic vectors T often greatly surpasses the number of output symbols U, suggesting a potential redundancy in the acoustic sequence. Indeed, [12] demonstrated that their RNN-T model works equally well even when an input acoustic vector is limited to output only one symbol, and they could reduce the decoding operation steps from $\mathcal{O}(U+T)$ to $\mathcal{O}(T)$.

1.2 Motivation

To investigate this intuition, I propose leveraging the CIF framework to reduce the length of the acoustic sequence from T to a more proportionate size M relative to the number of output symbols U. Specifically, I introduce CIF into the RNN-T model, utilizing it as an aggressive downsampling module instead of a Seq2Seq training objective, to aggressively reduce the number of acoustic vectors down to the number of linguistic morphemes. Subsequently, the model learns to draw multiple symbols from the compressed acoustic sequence, referred to as acoustic tokens, by training on the RNN-T loss. Some degradation in performance is expected compared to decoding from the full acoustic sequence of length T. Nonetheless, my objective is to minimize this degradation while reducing training memory footprint and decoding time.

To achieve my goals, I explore the performances of various combinations of CIF architectures in the context of Japanese and English ASR. Additionally, I propose a new CIF architecture that outperforms other CIF candidates in my experiment evaluations. The contributions of my work can be summarized as follows:

- I propose a new attention-based, ragged (variable-length) downsampling submodule that outperforms the conventional weighted sum mechanism.
- I propose a new lean weight prediction submodule that yields competitive results.
- I explored various intuitive perturbation strategies and demonstrate their effects on the CIF framework.
- I offer insights drawn from my experiment results and evidences, furthering the understanding of CIF in ASR.

Chapter 2

Related Works

2.1 Continuous Integrate-and-Fire

Given a sequence of target output symbols $\mathbf{Y}^* = (y_0^*, ..., y_u^*, ..., y_{U-1}^*)$, the Continuous Integrate-and-Fire (CIF) module compresses acoustic vectors, i.e. the output of an acoustic encoder $\mathbf{H} = (\mathbf{h}_0, ..., \mathbf{h}_t, ..., \mathbf{h}_{T-1})$, into shorter acoustic tokens $\mathbf{C} = (\mathbf{c}_0, ..., \mathbf{c}_m, ..., \mathbf{c}_{M-1})$, where $U \simeq M < T$. This process includes predicting a weight sequence $\alpha = (\alpha_0, ..., \alpha_t, ..., \alpha_{T-1})$ for each time step $\mathbf{T} = (0, ..., t, ..., T-1)$. In more detail, the CIF module can be broken down into the following five steps:

1. **Weight Prediction Submodule (ω)**: Given acoustic vectors $\mathbf{H}$ that spans across time steps $\mathbf{T}$, CIF predicts scalar, non-negative weights α for all vectors in the sequence. These weights will be used in the ragged downsampling to determine firing points. Mathematically, this can be represented as:
$$\alpha_t = \omega(\mathbf{h}_t) \tag{2.1}$$
where ω denotes this weight prediction function.

2. **Weight Scaling**: During training when a target length M^* (e.g. the number of output symbols) is known, the weights α are scaled to α' such that they sum up to βM^*. Here, β is a hyperparameter firing threshold whose purpose will be explained in the Ragged Downsampling step. During inference when the target length M^* is unknown, the weights are used as-is.

3. **Quantity Loss ($\mathcal{L}_{qua}$)**: To ensure that the predicted weights can be directly used when the target length M^* is unknown, CIF incorporates an additional training objective to minimize the difference between the predicted length $\hat{M}$ and the target length M^* during training. This

objective, commonly referred to as the Quantity Loss $\mathcal{L}_{qua}$, can be expressed as follows:

$$\hat{M} = \frac{\sum_{t \in T} \alpha_t}{\beta} \tag{2.2}$$

$$\mathcal{L}_{qua} = \quad | \, M^* - \hat{M} \, | \tag{2.3}$$

4. **Ragged Downsampling (ϕ)**: CIF defines firing points τ as time steps where the cumulative sum of weights exceeds an integer multiple of β. The range of time steps from the previous firing point τ_{m-1} to the current τ_m constitutes a segment $\mathbf{m}$. Vectors in the same segment are then pooled (or *integrated*) to produce acoustic tokens $\mathbf{c}_m$, a vector in the output sequence $\mathbf{C}$. This process can be described as:

$$\text{Given } \tau_{m-1}, \quad \tau_m = \tau \quad \text{if} \quad \sum_{t=\tau_{m-1}}^{\tau} \alpha'_t \geq \beta \tag{2.4}$$

$$\mathbf{T} \supseteq \mathbf{m} = (\tau_{m-1}, ..., t_m, ..., \tau_m)$$
$$\mathbf{c}_m = \phi(\mathbf{h}_{t_m}, \alpha_{t_m} \mid t_m \in \mathbf{m}) \tag{2.5}$$

where ϕ denotes the ragged downsampling function.

5. **Cross Entropy (CE) Loss**: Since the number of acoustic tokens M are adjusted during training to match the number of target output symbols U, the acoustic tokens $\mathbf{C}$ are trained using a per-symbol Cross Entropy (CE) loss against the target output symbols $\mathbf{Y}^*$.

Because the hyperparameter firing threshold β is usually set to 1, it is occasionally omitted in some CIF notations. The following subsections will introduce past approaches proposed for weight scaling, weight prediction, ragged downsampling, and cross entropy loss.

2.2 Weight Scaling

A lack of diversity in CIF implementations becomes apparent, as the literature reviewed [2, 3, 4, 5, 6, 7, 8, 10, 13, 14] invariably adheres to the conventional weight scaling method in the original proposal by [1]. This process can be expressed as:

$$\alpha'_t = \begin{cases} \dfrac{M^*}{\hat{M}} \alpha_t & \text{if } M^* \text{ is known} \\ \alpha_t & \text{otherwise} \end{cases} \tag{2.6}$$

where $\hat{M} = \sum_{t \in T} \alpha_t / \beta$ denotes the predicted length.

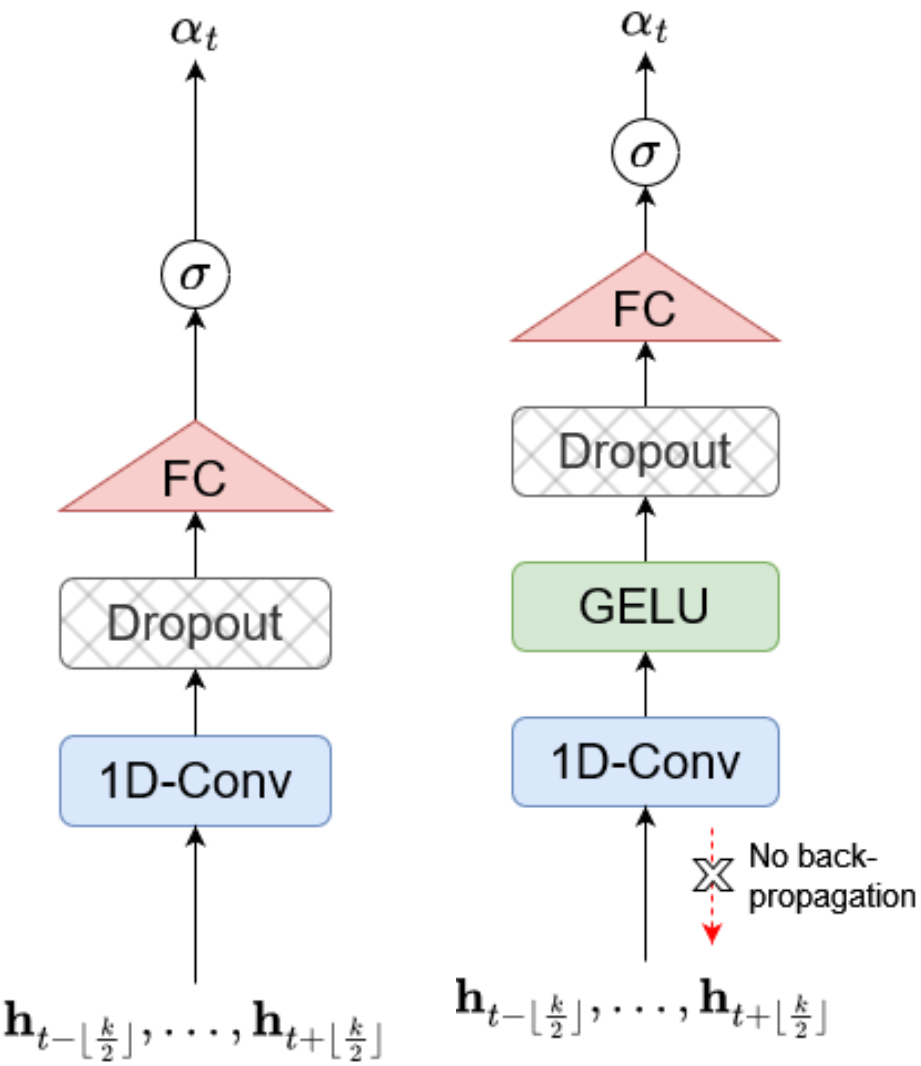
Two interesting points arise from this straightforward method. Firstly, since the target lengths M^* are discrete integers, it would be near impossible to expect the unscaled weights which consist of continuous floating point values to accurately and consistently sum up to M^*. This requires inference-only mechanisms like tail handling to cater for scenarios where acoustic information accumulates in the final segment but does not reach the firing threshold, arguably exacerbating the train-test mismatch. Secondly, although the weights as outputs from a sigmoid activation function are expected to assume the range $0 < \alpha_t < 1$, it is possible for weights to exceed 1 via scaling during training, when the predicted length $\sum_{t \in T} \alpha_t$ is smaller than the target length M^*. This does not happen during inference time.

2.3 Weight Prediction (ω)

Most implementations of CIF [5, 6, 7, 8, 13, 14] adopt the conventional weight prediction submodule proposed by [1]. This design, illustrated in Figure 2.1a as "ConvFc", consists of a 1-dimensional convolution network followed by a dropout layer and a fully-connected layer with a single output neuron, activated by a final sigmoid function.

For simultaneous speech translation, [3] introduced a more sophisticated design, illustrated in Figure 2.1b. It sequentially consists of a 1-dimensional convolutional layer, a LayerNorm, a Gaussian Error Linear Unit (GELU) activation layer, a Dropout layer, a fully-connected layer, and a final sigmoid activation layer. Gradients from the submodule are not allowed to backpropagate into the encoder. In this thesis, I term this design "ConvActFc".

There are other designs beyond my comparison scope. For example, [4] proposed a submodule for simultaneous speech translation that consumes the last dimension of the output of an encoder pretrained with self-supervised learning (SSL). In my work, I have a different objective since I jointly optimize the encoder and the CIF module. Additionally, [2] introduced "DifferNet" for speaker change detection that incorporates a mechanism to compare the current vector with a pooled representation of past vectors in the same segment. I excluded this design from my comparison because the additional operations, albeit natural for speaker change detection, adds computation overhead without providing intuitive utility in the ASR context.



(a) ConvFc [1] (b) ConvActFc [3]

Figure 2.1: Weight prediction submodules $\alpha = \omega(\mathbf{H})$ described in the literature, where k refers to the kernel size of the convolutional layer.

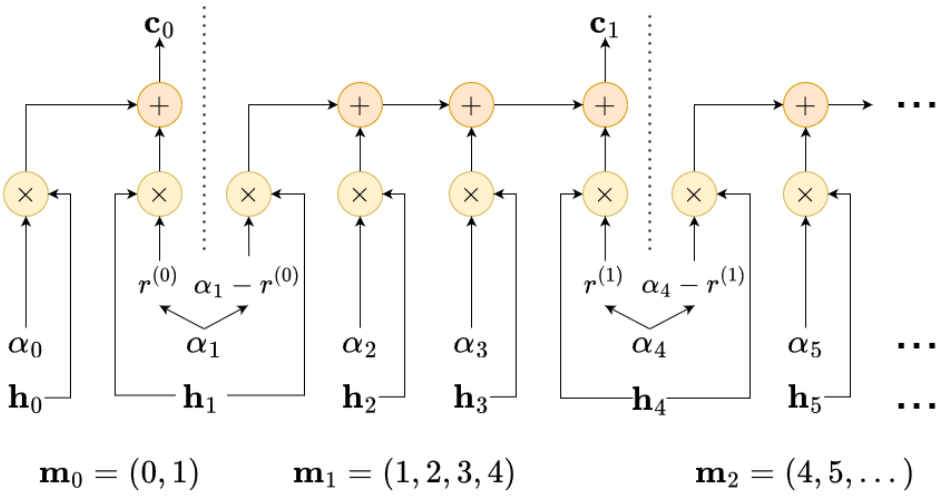


Figure 2.2: The conventional downsampling method, which I term "Cascade" downsampling, where the weight α_τ at the firing time step τ is split between the preceding and succeeding segments.

2.4 Ragged Downsampling Submodule (ϕ)

Most implementations [2, 3, 4, 5, 6, 7, 8, 10, 13, 14] in the literature follow the conventional integration process proposed by [1] and represented in Figure 2.2, suggesting a scarcity of investigations into alternative integration strategies.

In this method, input vectors $\mathbf{h}_t$ are scaled by their weights α_t and accumulated in a forward manner. When a segment m reaches its firing point τ_m with the conditions of equation 2.4, the input vector at the firing point $\mathbf{h}_{\tau_m}$ undergoes a complex two-step operation.

First, it is scaled by $r^{(m)}$, the remainder required for the cumulative sum of weights so far to reach the firing threshold β. This scaled vector is then added to the accumulated vector obtained so far, resulting in the pooled vector $\mathbf{c}_m$. Second, the same input vector $\mathbf{h}_{\tau_m}$ is scaled by the difference $\alpha_{\tau_m} - r^{(m)}$ and starts a new pool for the subsequent accumulation of the next segment $m + 1$. I refer to this conventional method as "Cascade Downsampling".

2.5 Cross Entropy (CE) Loss

In the traditional context of ASR before the advent of Seq2Seq models, the Cross Entropy (CE) loss is commonly associated with framewise classification, where explicit alignment between the longer input acoustic vectors and shorter output symbols is required. However, the CIF framework, by being trained to produce one acoustic token per symbol, implicitly learns a soft alignment between its inputs and outputs through its weight prediction and ragged downsampling submodules. As a result, it can optimize its outputs directly using the CE training objective without the need for explicit alignment.

However, directly decoding the compressed acoustic tokens from CIF may yield inferior results, which necessitates additional architectural reinforcement, such as the autoregressive decoder [15], as proposed in the original CIF framework [1]. In this work, I choose not to consider the autoregressive decoder as part of the CIF Seq2Seq framework. The reasoning behind this decision is that the autoregressive decoder can be added to any other architecture to realize the same Character Error Rate (CER) improvements theoretically.

Chapter 3

Methodology

Figure 3.1 provides an overview of my proposed architecture, where I insert CIF between the encoder and joiner modules of the RNN-T as a aggressive downsampling module. Specifically, I detach the per-symbol CE training objective from CIF, leveraging its temporal compression capability purely as a segment-wise pooling mechanism with dynamic boundaries along the time dimension.

This modification offers several advantages. Firstly, by detaching the training objective from CIF, I can independently define the desired number of acoustic tokens M and the number of output symbols U, with the joiner module determining the alignment between them. Secondly, since $M < T$, I can reduce the RNN-T loss matrix from (B, T, U, V) to (B, M, U, V), and the decoding operations from $\mathcal{O}(T + U)$ to $\mathcal{O}(M + U)$.

This chapter continues with several changes I made to the CIF, addressing the aspects outlined earlier.

3.1 Ragged Downsampling (ϕ)

I proposed two more ragged downsampling designs in my work to be compared against the conventional Cascade downsampling method. Figure 3.2 illustrates my proposals.

3.1.1 Sozu Downsampling

To simplify the pooling process and avoid the two-step operation of the Cascade downsampling method, I propose a straightforward modification. Instead of splitting the acoustic vector $\mathbf{h}_\tau$ at the firing time step τ between the preceding and subsequent segments, I assign the entire vector to the preceding segment. This adjustment allows the accumulation of the next

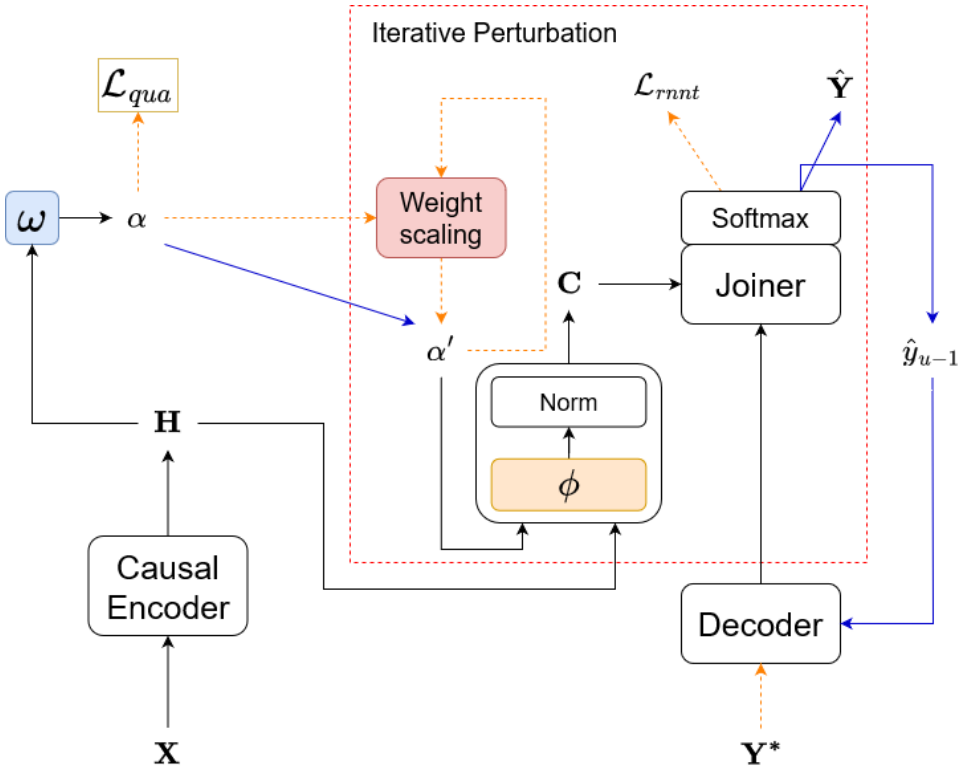


Figure 3.1: My proposed RNN-T model incorporating the CIF module. The colored portions highlight my CIF module formulation. as detailed in Section 2.1. and serve as grounds for comparison among different designs to evaluate the impact of various choices on error rate, memory footprint, and decoding speed. During training, the model follows the dotted orange arrows, evaluating the region indicated by the dotted red frame iteratively with weights perturbed to varying degrees. At inference time, I follow the solid blue arrows instead.

segment to begin with an empty vector, simplifying the process. Then, the accumulation of the next segment starts with an empty vector.

It is worth noting that this modification leads to fluctuations around β in the sum of weights within each segment, causing inconsistently scaled acoustic tokens. To address this issue, I introduce an optional normalization step, shown within a green frame in the diagram. Specifically, I scale the compressed output of each segment m by the sum of weights $\sum_{t \in \mathbf{m}} \alpha_t$ within that segment to ensure more consistent magnitudes in the resulting acoustic tokens. I will thoroughly investigate the impact of this normalization step

in my experiments.

I name this modified method "Sozu (添水)" [1], also known as "Shishi-odoshi (鹿威し)" in Japanese. The term "Sozu" refers to a bamboo tube with an open end that empties all of its water once the accumulated water crosses its tipping point. The analogy highlights the similarity between the behavior of the bamboo tube and the way this modification works, where the entire acoustic vector is assigned to (or *flushed out for*) the preceding segment before moving on to the next segment.

3.1.2 Ragged (Multi-head) Attention

Drawing inspiration from the AttentionDownsample module in Icefall [2], this integration process employs multi-head attention to generate a segment-level representation using a learnable query vector $\mathbf{q}_\theta$. The analogy to how the `[CLS]` token captures sentence-level representation in [16] provides an intuitive understanding of my integration process. Shown in Figure 3.2b, I divide the T dimension into M segments based on the firing points defined in Equation 2.4. Unlike Icefall's fixed downsampling factor, each segment in my process has varying length $|\mathbf{m}|$ (i.e., being *ragged*), as the firing points happen at irregular intervals.

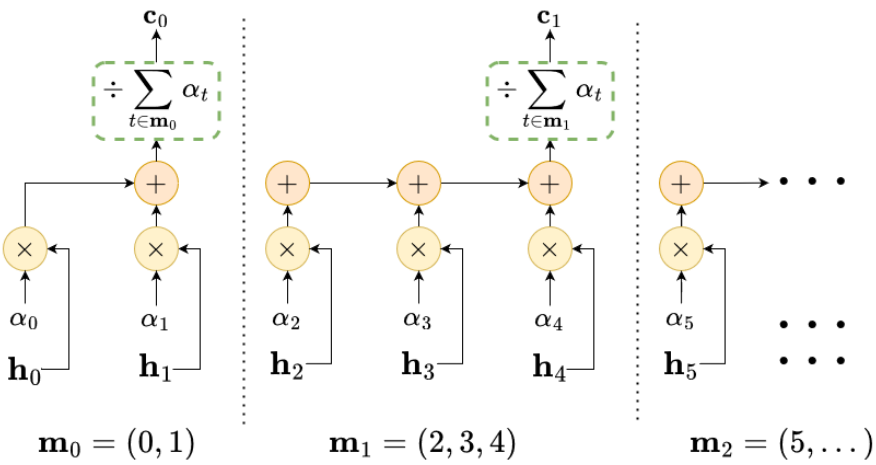
I adopt the bullet $\bullet$ to represent a ragged relation between dimensions. For instance, $\mathbf{H}^{M\bullet|\mathbf{m}|\times d}$ represents the acoustic vectors in 3 dimensions, where M denotes the number of segments, $|\mathbf{m}|$ the number of time steps within each segment, and d the dimension along which attention is calculated. This shape, where elements along dimensions $|\mathbf{m}|$ and d have regular length while elements along dimension M vary in length, allows the query vector to attend to each segment separately, akin to how the `[CLS]` token is appended to each sentence separately in BERT. Thus, attention operations are performed only within each segment, along dimension $|\mathbf{m}|$, ensuring mutual independence among segments for streaming inference.

The ragged multi-head attention downsampling mechanism is summarized by the following equations, where I notate the interacting dimensions following PyTorch's broadcasting semantics [3] and omit other implicit dimensions such as the batch size B and number of heads h for brevity. I use a subscript like $f_{|\mathbf{m}|}$ to represent function f applied along the dimension $|\mathbf{m}|$.

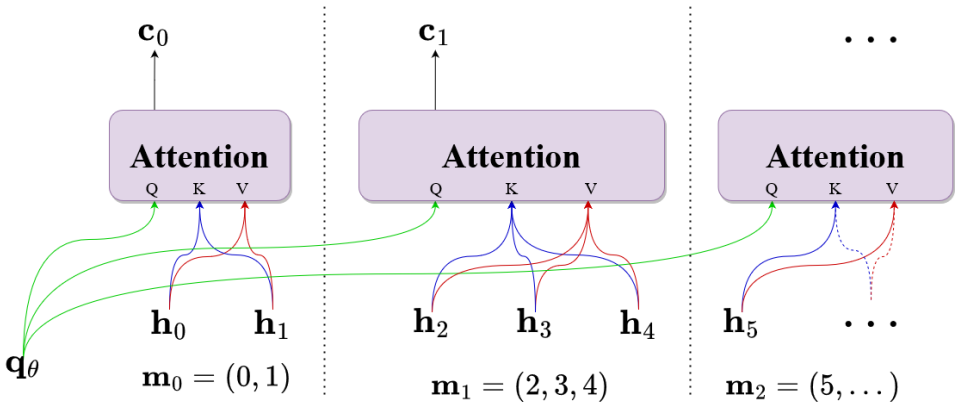
[1] https://en.wikipedia.org/wiki/Shishi-odoshi

[2] https://github.com/k2-fsa/icefall/egs/librispeech/ASR/pruned_transducer_stateless7_streaming/zipformer.py

[3] https://pytorch.org/docs/stable/notes/broadcasting.html



(a) Sozu Downsampling



(b) Ragged (Multi-head) Attention Downsampling

Figure 3.2: Ragged downsampling methods explored in this paper. In Figure 3.2a, I present a new design that ablates the splitting of α_τ, assigning the firing time step entirely to the previous segment. Finally, Figure 3.2b showcases my proposed ragged downsampling method, featuring a multi-head attention mechanism.

$$
\begin{aligned}
\mathbf{K}^{M \bullet |\mathbf{m}| \times d} &= \mathbf{V}^{M \bullet |\mathbf{m}| \times d} = \mathbf{H}^{M \bullet |\mathbf{m}| \times d} + \mathbf{P}^{M \bullet |\mathbf{m}| \times d} \\
\mathbf{A}^{M \bullet |\mathbf{m}|} &= \mathrm{softmax}_{|\mathbf{m}|} \left[\sum_{d} \left(\frac{\mathbf{q}_\theta^{1 \times 1 \times d}}{\sqrt{d}} \odot \mathbf{K}^{M \bullet |\mathbf{m}| \times d} \right) \right] \\
\mathbf{C}^{M \times d} &= \sum_{|\mathbf{m}|} \left(\mathbf{A}^{M \bullet |\mathbf{m}| \times 1} \odot \mathbf{V}^{M \bullet |\mathbf{m}| \times d} \right)
\end{aligned}
$$

where $\odot$ represents the element-wise product and $\mathbf{P}$ non-learnable positional encodings [17] scaled according to the model dimension.

3.2 Weight Prediction Submodule (ω)

To ensure the strictly positive nature of weights in the weight prediction submodule (ω), I introduce the Exponential Rectified Linear Unit (eReLU). This piece-wise activation function combines an exponential function for negative values and a Rectified Linear Unit (ReLU) for positive values, ensuring that all predicted weights are positive. The eReLU activation function is defined as follows:

$$\mathrm{eReLU}(x;\varepsilon) = \begin{cases} x & \text{if } x \geq \varepsilon \\ \varepsilon e^{x} & \text{otherwise} \end{cases} \tag{3.1}$$

where $0 < \varepsilon \ll 1$.

The eReLU activation offers two essential advantages over the conventional ReLU activation function in the weight prediction submodule. Firstly, weights exactly equal to zero will remain unaffected by the weight scaling mechanism, causing an uneven distribution of scaled weights across time steps. Secondly, the eReLU activation ensures that gradients from the quantity loss can still backpropagate into the model, even for time steps whose weights are close to zero.

In pursuit of a leaner weight prediction submodule, I propose three new designs. My early experiments revealed that using a fully-connected layer to project a high-dimensional vector down to a scalar value resulted in abnormally large gradients in the weight prediction submodule. To address this, all of my proposed designs, explained below, utilize a simple mean-pooling technique to obtain the scalar weight.

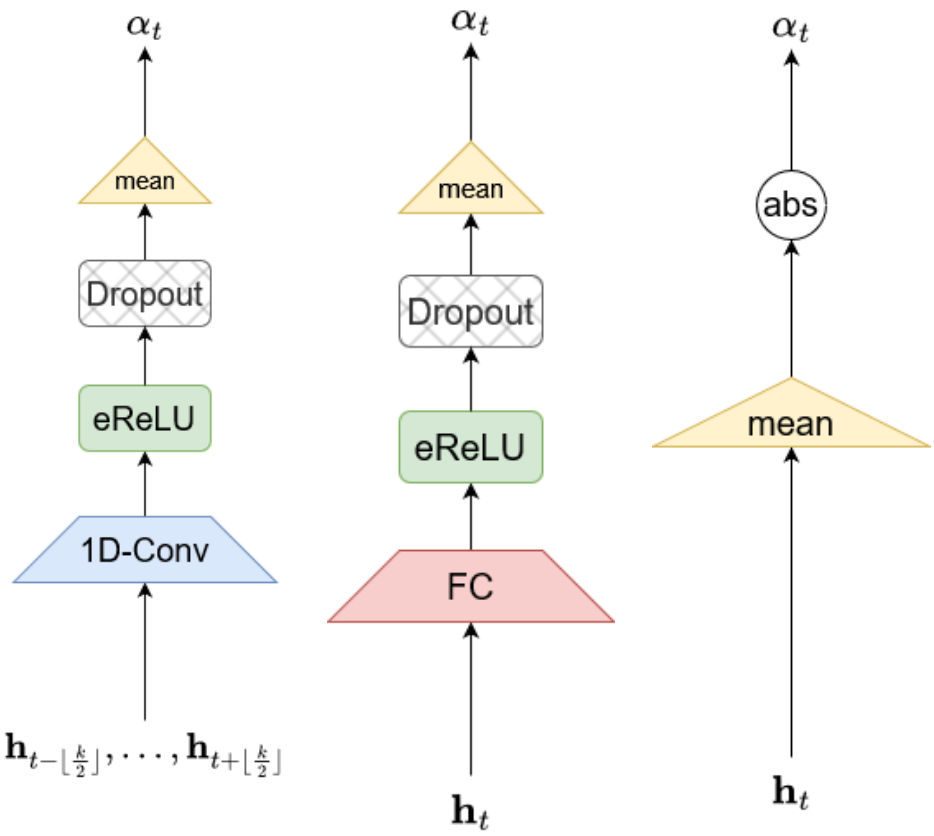
1. **ConvActMean** builds upon the intuition of previous approaches and leverages neighboring vectors through a convolutional network but produces a dimensionally low output with d_ω channels. It then continues with the aforementioned eReLU function, a Dropout layer, and a mean pooling operation.

2. **FcActMean** replaces the convolutional network in ConvActMean with a simpler fully-connected layer that also converts the input to d_ω channels.

3. **MeanAbs** represents a minimalistic, parameter-less design that I explored, consisting only a mean and a magnitude operation. Mathematically, it can be summarized as:

$$\alpha_t = \mid \overline{\mathbf{h}_t} \mid \tag{3.2}$$

Newly proposed ω functions



(a) ConvActMean (b) FcActMean (c) MeanAbs

Figure 3.3: Weight prediction submodules $\alpha = \omega(\mathbf{H})$ proposed in this work, where k refers to the kernel size of the convolutional layer. Narrower widths indicate smaller dimensions.

3.3 Weight Scaling and Perturbation

To address the challenges discussed in Section 2.2, I propose the following training strategies, which are summarized in Algorithm 1. In the algorithm, $\mathcal{U}(0,1)$ represents a random value sampled from the Uniform distribution between 0 and 1, and $\mathcal{N}(1, 0.1^2)$ a random value sampled from the Normal distribution with mean 1 and variance 0.1^2.

1. **Target length perturbation**: With a probability of perturbation rate ρ, I introduce slight perturbations to the target length M^* to

Algorithm 1: Weight scaling and perturbation

Input: Weights $\alpha = (\alpha_0, ..., \alpha_t, ..., \alpha_{T-1})$, target length M^*, perturb rate $0 \leq \rho \leq 1$, and perturb times (N_{reset}, N)

Output: Perturbed, scaled weights $\tilde{\alpha}' = (\tilde{\alpha}'_0, ..., \tilde{\alpha}'_t, ..., \tilde{\alpha}'_{T-1})$

```
for i ← 1 to N_reset do
    α̃ ← α
    for j ← 1 to N do
        if U(0,1) < ρ then
            M̃* ← M* × min(N(1, 0.1²), 0.9)
        else
            M̃* ← M*
        if U(0,1) < ρ then
            forall α̃_t do  α̃_t ← α̃_t × N(1, 0.1²)
        M̂̃ ← Σ_{t∈T} α̃_t
        if M̃* > 50M̂̃ then
            forall α̃_t do α̃_t' ← M̃*/T
        else
            forall α̃_t do  α̃_t' ← min(M̃*/M̂̃ α̃_t, 0.99)
        yield α̃'
```

simulate incomplete last segments that do not reach the firing threshold β. This is achieved by multiplying M^* with $\mathcal{N}(1, 0.1^2)$ that is clamped to a minimum of 0.9 to prevent excessively short output sequences $\mathbf{C}$. This perturbation propagates to the individual weights through scaling.

2. **Weight clamping**: I clamp the scaled weights to a maximum value of 0.99 to ensure that each input vector $\mathbf{h}_t$ is represented by at most one element in the output sequence $\mathbf{C}$, bringing the training conditions closer to inference. Moreover, if the predicted length $\hat{M}$ is disproportionately small compared to M^*, e.g. with a ratio of 50 : 1, I redistribute weights equally to all time steps, similar to fixed-length downsampling.

3. **Weight perturbation**: To enhance the model's generalization across different compressed outputs, I slightly perturb the individual weights with a probability of ρ.

4. **Iterative perturbation**: Using a schedule defined by two integers

(N_{reset}, N), I create multiple sets of weights for each batch of data to evaluate an RNN-T loss $\mathcal{L}_{RNNT}^{(i,j)}$, where $i \leq N_{reset}$ and $j \leq N$. The final RNN-T loss for backpropagation is obtained by averaging those evaluated RNN-T losses.

Therefore, my final training objective bears the form:

$$\mathcal{L} = \frac{\lambda_{rnnt}}{N_{reset}N} \sum_{i=1}^{N_{reset}} \sum_{j=1}^{N} \mathcal{L}_{rnnt}^{(i,j)} + \lambda_{qua}\mathcal{L}_{qua} \tag{3.3}$$

where λ_{rnnt} and λ_{qua} represent the weights for the RNN-T loss and Quantity loss, respectively.

Chapter 4

Experiment Setup

4.1 Datasets

I conduct experiments on two languages: English and Japanese. For data preprocessing, I extract acoustic features using 80-channel filter banks with a window of 25ms and a stride of 10ms. To augment the data, I apply spec-augmentation [18] and noise augmentation with [19], excluding speed perturbation to save time and not triple the training data.

For each language dataset, I consider two definitions of the target length M^*. In the "Sym" case, M^* is defined as the number of output symbols, i.e., $M^* = U = |\mathbf{Y}^*|$, whereas for "Mor", M^* is set to be the number of morphemes, where Sym $\geq$ Mor. My definition for morphemes will be explained separately for each language.

Librispeech (English)

The English dataset consists of the 100-hour and 360-hour 'clean' subsets of Librispeech [20] for training, and the full 'clean' and 'others' sets in the development and test subsets. The number of morphemes is defined as the number of space-demarcated words in the transcript, and I use a set of 500 tokens obtained from Byte Pair Encoding (BPE) [21] as the output symbol set.

Table 4.1 shows the statistics of audio lengths for samples in the training, validation, and evaluation sets. During training, samples that are shorter than 0.3 seconds or longer than 20 seconds are excluded for better training efficiency.

Subset	Train	Valid	test-clean	test-other
Sample count	132,553	5,567	2,620	2,939
Tot. dur. (hh:mm:ss)	464:11:48	10:30:33	5:24:13	5:20:30
Mean (s)	12.6	6.8	7.4	6.5
Std (s)	3.6	4.5	5.2	4.4
Min (s)	1.1	1.1	1.3	1.2
25% (s)	11.4	3.6	3.7	3.4
50% (s)	13.9	5.5	5.8	5.2
75% (s)	15.1	8.6	9.6	8.2
99% (s)	16.7	23.6	25.5	21.4
99.5% (s)	16.8	28.0	28.4	23.8
99.9% (s)	17.1	32.4	32.8	33.5
Max (s)	29.7	35.2	35.0	34.5

Table 4.1: Audio lengths of samples in the Librispeech training, validation, and evaluation subets. 'Sample count' shows the number of samples in each subset, and 'Tot. dur' displays the total duration of the subset in (hh:mm:ss) format. The subsequent rows provide detailed audio length statistics in seconds (s) for each subset, including the mean, standard deviation, minimum, 25th percentile, 50th percentile, 75th percentile, 99th percentile, 99.5th percentile, 99.9th percentile, and maximum audio lengths among samples in the corresponding subset.

Corpus of Spontaneous Japanese (Japanese)

The Japanese dataset is the Corpus of Spontaneous Japanese (CSJ) [22], which I partition into training, validation, and evaluation subsets with the Lhotse recipe [23]. To focus the weight prediction module on meaningful utterances rather than just human voice, I work with a fluent transcript version where fillers (F tags) and partial words (D and D2 tags) are removed. The number of morphemes is determined by the labeled morphemes in the CSJ transcript, and I use an output symbol set of 3266 characters.

Table 4.2 provides the statistics of the audio lengths for samples in the training, validation, and evaluation subsets. Similar to Librispeech, I exclude samples that are shorter than 0.3 seconds or longer than 20 seconds during training. Notably, samples in CSJ display a more uniform distribution across subsets. This is characteristic is a result of the samples being segmented heuristically from long continuous recordings.

Subset	Train	Valid	eval1	eval2	eval3
Sample count	304,717	3,743	1,023	1,025	865
Tot. dur. (hh:mm:ss)	561:23:05	6:40:15	1:55:40	2:02:07	01:26:44
Mean (s)	6.6	6.4	6.8	7.1	6.0
Std (s)	2.8	3.0	2.7	2.5	3.0
Min (s)	0.1	0.1	0.2	0.1	0.3
25% (s)	4.6	3.9	4.9	5.9	3.3
50% (s)	7.6	7.4	7.7	7.9	6.8
75% (s)	9.0	9.0	9.0	9.1	8.7
99% (s)	10.0	10.0	10.0	10.0	10.0
99.5% (s)	10.0	10.0	10.0	10.0	10.0
99.9% (s)	10.2	10.1	10.0	10.0	10.0
Max (s)	16.2	11.8	10.0	10.0	10.0

Table 4.2: Audio lengths of samples in the CSJ training, validation, and evaluation subsets. 'Sample count' shows the number of samples in each subset, and 'Tot. dur' displays the total duration of the subset in (hh:mm:ss) format. The subsequent rows provide detailed audio length statistics in seconds (s) for each subset, including the mean, standard deviation, minimum, 25th percentile, 50th percentile, 75th percentile, 99th percentile, 99.5th percentile, 99.9th percentile, and maximum audio lengths among samples in the corresponding subset.

4.2 Framework

I conduct my experiments using the Next Generational Kaldi (NGK) framework [24], specifically, the "pruned transducer stateless7 streaming" recipe in Icefall [1]. My encoder module follows the design of the streaming "zipformer" with the original recipe's hyperparameters, which has demonstrated superior results [2] over the prior streaming Conformer [25]. However, I remove the final downsampling layer since I are already relying on an external CIF module to aggressively downsample the encoder output. My stateless decoder module [26] extends the context size to 4 while otherwise maintaining the original Icefall design.

To ensure a fair comparison of past and new methods on the same encoder, I implement the ConvFc and ConvActFc weight prediction submodules in Icefall. For the conventional downsampling method, I use torch_cif [3]

[1] https://github.com/k2-fsa/icefall
[2] https://github.com/k2-fsa/icefall/blob/master/egs/librispeech/ASR/RESULTS.md
[3] https://github.com/George0828Zhang/torch_cif

that parallelizes the for-loop along the time dimension. I observed through early experiments that my RNN-T models with the conventional Cascade method accumulated huge gradients and failed to converge. I attribute this issue to the substantial difference between the source length T and the target length M, as I do not use a final fixed-length downsampling layer in the encoder, unlike [1]. To address this, I stop the gradients originating from the RNN-T loss from flowing back into the encoder via the weights.

4.3 Hyperparameters

For training, I use RNN-T loss pruning [27] with a prune range of 8 for Japanese and 16 for English, unless otherwise mentioned. These values are roughly estimated based on the number of output symbols in a morpheme. All models were trained for 40 epochs. Both decoding uses a beam size of 4 and a chunk size of 640ms. I use the modified beam search method [12] for the CIF-less model, which has an operation complexity of $\mathcal{O}(T)$. For the CIF models, I use a batched beam search method that advances each hypothesis in either the utterance U or time M directions, and do not use any inference-only techniques like tail-handling [1]. Chosen empirically, each acoustic token $\mathbf{c}_m$ of CIF models can output at most 5 symbols for Japanese and 9 symbols for English. For training and decoding, I use a maximum duration of 240 seconds and add a 30-frame pad at the end of each utterance.

Moreover, I set the firing threshold β to 1, the perturb times (N_{reset}, N) to $(4, 2)$, the ε in the eReLU function to 0.01, the number of channels d_ω in ConvActMean and FcActMean to 4, and the number of heads h in the ragged attention downsampling submodule to 8.

Chapter 5

Results and Discussion

In this chapter, I present the results of my experiments and provide in-depth discussions to explain my decisions and insights. The chapter is structured as follows. I begin by analyzing and comparing the effects of incorporating CIF into RNN-Ts using two downsampling methods: the conventional Cascade downsampling and my proposed Ragged Attention downsampling. Through my results, I demonstrate the superior performance of my proposed method over the conventional approach. Next, I report the results of my ablation study, conducted on my proposed weight scaling and perturbation strategies. By evaluating different configurations, I gain insights into the effectiveness of these techniques and their contributions to the overall performance. Then, I compare Cascade downsampling against Sozu downsampling, investigating how the difference impacts the model's efficiency and accuracy. Finally, I perform a thorough architectural comparison among various notable configurations of the weight prediction (ω) and ragged downsampling (ϕ) submodules.

Among different combinations of averaged checkpoints from previous epochs up to epoch 40, only the models which yielded the best CERs on the validation set were selected to evaluate the test sets and reported here. Through these analyses, I aim to provide a comprehensive understanding of my proposed methods and their impact on the performance of RNN-T models. My discussions will highlight the key findings and potential areas for further improvement and research.

5.1 Cascade vs R.Attn and Sym vs Mor

In this section, I present a detailed comparison of different downsampling methods and target length (M^*) definitions for RNN-T models with incorporated CIF, compared against the CIF-less RNN-T model. I analyze the

results with a focus on accuracy, memory reduction, and decoding speedup.

	Model	CIF-less	Cascade		R.Attn	
	Add. Param.	0	446,850		447,234	
	M^* Type	-	Sym	Mor	Sym	Mor
Librispeech	test-clean (CER)	4.26	4.60	5.34	4.39	4.63
	test-other (CER)	13.02	13.83	14.79	13.41	13.52
	Mem. Red. (%)	100	99	100	098	102
	Speedup (%)	100	128	154	123	149
CSJ	eval1 (CER)	4.03	4.37	4.98	4.28	4.44
	eval2 (CER)	3.42	3.71	4.22	3.45	3.87
	eval3 (CER)	3.58	3.88	4.30	3.82	4.27
	Mem. Red. (%)	100	149	180	148	179
	Speedup (%)	100	115	139	111	134

Table 5.1: Comparison of ragged downsampling submodules and target length M^* definitions for Librispeech and CSJ. CERs are reported for each test set. The row labeled 'Add. Param.' shows the number of additional parameters compared to the original CIF-less model. 'Mem. Red.' represents the reduction in memory footprint achieved by the proposed models, calculated as the ratio between the training memory footprints of the CIF-less model and the corresponding model in each column. 'Speedup' refers to the improvement in decoding time, calculated as the ratio between the decoding times of the CIF-less model and the corresponding model in each column.

Character Error Rates (CERs)

Across all downsampling methods and target length M^* definitions, setting M^* to the symbol count "Sym" consistently yielded better results compared to the morpheme count "Mor". This outcome is expected, as reducing the acoustic length to the symbol count provides more acoustic context to the RNN-T model.

My proposed Ragged Attention (R.Attn) downsampling submodule consistently outperformed the conventional Cascade method, showing the superiority of the proposed method. However, it is important to note that the best performing CIF design did not surpass the performance of the CIF-less model.

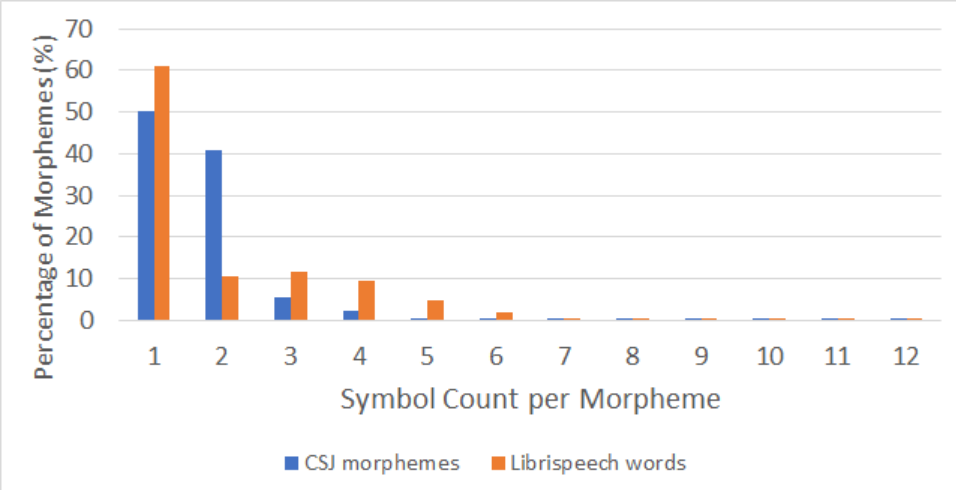


Figure 5.1: Distribution of symbol count per morpheme in the training subsets of Librispeech and CSJ.

Memory Reduction

CIF achieved significant memory reduction for the CSJ dataset but not for Librispeech. This discrepancy can be attributed to the larger vocabulary size V in CSJ compared to Librispeech. The reduction in acoustic length from T to the target length M provided more savings in the RNN-T loss matrix for CSJ due to the larger V. However, the additional parameters introduced by CIF offset these savings, resulting in negligible impact for Librispeech.

When setting M^* to "Mor", a lower training memory footprint was observed compared to setting it "Sym". This trend was more prominent for CSJ than Librispeech. Figure 5.1, illustrating the distribution of symbol count per morpheme across both languages, supports this finding. In CSJ, while 50% of the morphemes consist of 1 symbol (i.e., character), there is still potential for savings for the remaining 50% that have more than 1 symbol. On the other hand, in Librispeech, a significant portion (61%) of words already consists of 1 symbol (i.e., BPE token), limiting the potential savings by setting M^* to "Mor". This aligns with the nature of BPE tokenization, which aims to represent the language with the least amount of tokens.

Decoding Speedup

The simpler Cascade downsampling method, with fewer operations, achieved higher speedups compared to my proposed Ragged Attention downsampling. Additionally, setting M^* to "Mor" resulted in higher speedups compared to

Dataset	Librispeech		CSJ	
	Sym	Mor	Sym	Mor
Mean	10.1	19.3	6.1	9.5
Std	3.2	5.6	2.5	3.9
Min	4.8	9.9	0.2	0.3
25%	8.2	15.8	4.3	6.8
50%	9.4	18.3	6.9	10.9
75%	11.1	21.3	8.1	12.7
99%	20.2	40.2	9.0	14.1
99.5%	23.7	46.0	9.0	14.1
99.9%	32.5	62.3	9.0	14.2
Max	78.5	87.0	9.0	14.2

Table 5.2: Comparison of the number of acoustic vectors per acoustic token for datasets Librispeech and CSJ, using target length M^* definitions "Sym" and "Mor". These values are estimated by dividing the length of the acoustic vector by the acoustic token length for each data sample.

"Sym". Surprisingly, Librispeech exhibited overall higher speedups than CSJ. Table 5.2, which shows the statistics of the number of acoustic vectors per acoustic token, explains this observation. on average, acoustic tokens in Librispeech represent more acoustic vectors and occupy more temporal length than in CSJ. This indicates more temporal compression for Librispeech when reducing the acoustic vector length T to the acoustic token M, leading to greater savings in decoding operation steps from $\mathcal{O}(T + U)$ to $\mathcal{O}(M + U)$.

Furthermore, the gap between "Sym" and "Mor" for Librispeech is bigger than for CSJ, explaining the greater gains achieved in Lirbispeech by setting the target length to the shorter "Mor".

Overall Observations

The results presented here confirm two observations that I will not repeat in the following experiments:

1. "Sym" consistently outperforms "Mor" in terms of accuracy, but it leads to lower memory reductions and speedups.

2. CSJ exhibits significant memory reductions during training, whereas Librispeech shows only negligible reductions.

5.2 Weight Perturbation Strategies

Due to the higher symbol count per morpheme in Librispeech, the prune range for Librispeech is increased to 16 for all subsequent experiments. Additionally, I focus on the "Mor" definition of target lengths to maximize training memory reduction and decoding speedups.

Training strategies	Librispeech			CSJ		
	CERs	Sum	Impr.	CERs	Sum	Impr.
No perturbation	5.28, 14.74	20.02	-	4.98, 4.22, 4.30	13.50	-
+ α perturb.	5.31, 14.81	20.12	-0.10	4.94, 4.27, 4.24	13.45	0.05
+ M^* perturb.	**5.01**, 14.62	19.63	0.49	4.63, **3.99**, **3.94**	12.56	0.89
+ iter. perturb.	5.08, **13.98**	**19.06**	0.57	**4.43**, 4.06, 4.01	**12.50**	0.06
+ α clamping	5.14, 14.12	19.26	-0.20	4.50, 4.09, 3.96	12.55	-0.05
		Total Impr.	0.76		Total Impr.	0.95

(a) ConvFc + Cascade

Training strategies	Librispeech			CSJ		
	CERs	Sum	Impr.	CERs	Sum	Impr.
No perturbation	4.57, 13.46	18.03	-	4.44, 3.87, 4.27	12.58	-
+ α perturb.	4.59, 13.78	18.37	0.34	4.39, 3.76, 4.15	12.30	0.28
+ M^* perturb.	4.68, 13.50	18.18	0.19	4.43, 3.68, 4.07	12.18	0.12
+ iter. perturb.	**4.43**, **13.28**	**17.71**	0.47	4.43, 3.78, 4.32	12.53	-0.35
+ α clamping	**4.43**, 13.31	17.74	-0.03	**4.33**, **3.67**, **3.96**	**11.96**	0.57
		Total Impr.	0.29		Total Impr.	0.62

(b) ConvFc + R.Attn

Training strategies	Librispeech			CSJ		
	CERs	Sum	Impr.	CERs	Sum	Impr.
No perturbation	4.35, 13.29	17.64	-	4.57, 4.04, 4.40	13.01	-
+ α perturb.	4.42, **13.10**	**17.52**	0.12	4.39, 3.93, 4.14	12.46	0.55
+ M^* perturb.	4.41, 13.24	17.65	-0.13	4.29, 4.01, 4.20	12.50	-0.04
+ iter. perturb.	4.38, 13.23	17.61	0.04	4.34, **3.70**, **3.93**	**11.97**	0.53
+ α clamping	**4.32**, 13.25	17.57	0.04	**4.19**, 3.94, 3.98	12.11	-0.14
		Total Impr.	0.07		Total Impr.	0.90

(c) MeanAbs + R.Attn

Table 5.3: Ablation study results for weight scaling and perturbation strategies, comparing three CIF designs: "ConvFc+Cascade" (Table 5.3a), "ConvFc+R.Attn" (Table 5.3b), and "MeanAbs+R.Attn" (Table 5.3c) on both Librispeech and CSJ. Bold values are the best results in each column.

Table 5.3 summarises the ablation study results of my weight perturbation strategies, performed on three CIF designs: "ConvFc+Cascade", "Con-

vFc+R.Attn", and "MeanAbs+R.Attn". The first two designs were introduced in Section 5.1, while the third design combines my proposed Ragged Attention downsampling submodule with the parameter-less weight prediction submodule, MeanAbs.

Each row in Table 5.3 corresponds to a training strategy, progressively added along with training strategies in previous rows. In sequence, the strategies added are weight perturbation, target length perturbation, iterative perturbation, and weight clamping. CERs are shown for 'test-clean' and 'test-other' subsets for Librispeech and 'eval1', 'eval2', and 'eval3' subsets for CSJ. The CER sums for each subset are shown on the right. The improvement 'Impr.' achieved with each added strategy is calculated by deducting the CER sum of the previous row from the CER sum of the current row, where positive numbers indicate improvement and negative numbers indicate degradation. The total improvement ('Total Impr.') for each dataset achieved is indicated at the last row.

From the results, I observe that all models exhibit performance improvements when all perturbation strategies are applied compared to no perturbation. However, the impact of each strategy varies significantly across datasets and model designs. The improvements are greater for CSJ than Librispeech, indicating that the effectiveness of the perturbation may depend on characteristics of the dataset. Moreover, my proposed Ragged Attention which uses attention instead of the weights for acoustic vector compression achieved overall less improvements than the conventional Cascade downsampling which directly relies on the weights for compression.

1. **Weight perturbation (α perturb.)**: Weight perturbation shows negligible impact on "ConvFc+Cascade", but significant improves "ConvFc+R.Attn" and "MeanAbs+R.Attn".

2. **Target length perturbation (M^* perturb.)**: "ConvFc+Cascade" benefits the most from target length perturbation, with the largest single-step improvement at 0.89. However, the performance for "MeanAbs+R.Attn" deteriorated with target length perturbation.

3. **Iterative perturbation (iter. perturb.)**: Almost all models benefit from iterative perturbation, with "ConvFc+Cascade" on Librispeech realizing the greatest improvement of 0.57. "ConvFc+R.Attn" on CSJ, however, does not share this trend and instead saw a significant deterioration of -0.35.

4. **Weight clamping (α clamping)**: The impact of weight clamping varies among setups. The greatest improvement is realized by "ConvFc+R.Attn" on CSJ at 0.57, whereas "ConvFc+Cascade" on Lib-

rispeech experiences a slight accuracy degradation of -0.20 with weight clamping.

Overall, applying these perturbation strategies together improves the performance of all models, even though the strategies exhibit differing impacts depending on the dataset and design. I acknowledge that further research is needed to fully understand the observed tendencies in this study. Nonetheless, with the efficacy of the combination of perturbation strate I proceed with all strategies applied indiscriminately to all models in my subsequent experiments.

5.3 Cascade vs Sozu

In this section, I conduct experiments to compare the conventional Cascade downsampling approach with the Sozu downsampling method. The main objectives are to analyze the impact of removing the two-step operation on the acoustic vector $\mathbf{h}_\tau$ at the firing time step τ and to investigate the effects of the normalization step on the output acoustic tokens. I use the "ConvFc" weight prediction submodule for this comparison.

Ablating the Two-Step Operation

"Sozu ($\times$)" removes the two-step operation and assigns $\mathbf{h}_\tau$ entirely to the previous segment. As expected, this modification results in a slight improvement in decoding speedup (3%) compared to the conventional Cascade downsampling. Surprisingly, "Sozu ($\times$)" also demonstrates a pronounced advantage in accuracy. This establishes the superiority of Sozu downsampling over the conventional Cascade downsampling method, both in terms of decoding time and CER.

Impact of Normalization

I explore the effect of adding a normalization step to the output acoustic tokens for Sozu downsampling, labeled as "Sozu ($\checkmark$)". This step attempts to scale the acoustic tokens to maintain consistency in the sum of weights within each segment. The additional scaling operation leads to noticeably slower decoding when compared against both "Sozu ($\checkmark$)" and Cascade downsampling. However, in exchange, further CER improvements are realized.

It is important to note that the normalization step has a potential weakeness in dealing with short, insignificant utterances. These utterances often contain lightly uttered interjections or unclear partial words, and are labeled as fillers in CSJ. They are represented as empty transcript, $\mathbf{Y}^* = \varnothing$.

ϕ	Librispeech			CSJ		
	CERs	Sum	Speedup	CERs	Sum	Speedup
Cascade	5.14, 14.12	19.26	1.54	4.50, 4.09, 3.96	12.55	1.39
Sozu (×)	4.77, 13.99	18.76	**1.57**	4.51, **3.81**, 3.88	**12.20**	**1.42**
Sozu (✓)	**4.60**, **13.55**	**18.15**	1.50	**4.46**, 3.92, **3.77**	12.15	1.37

Table 5.4: Comparison between Cascade and Sozu downsampling methods. "Sozu (×)" removes the normalization step and "Sozu (✓)" uses the normalization step. Bold values are the best results in each column.

In such cases, the normalization obscures the original weights assigned by the weight prediction submodule and magnifies the acoustic tokens, making them indistinguishable from fully uttered words. This effect is particularly evident in the CSJ dataset's "excluded" subset, which contains many dialogue utterances not used in training, validation, or testing. The CER for "Sozu (✓)" in this subset is higher (7.60) compared to "Sozu (×)" (CER of 4.67), indicating a clear distinction between the two.

I acknowledge this disadvantage of the normalized Sozu as a special case, where a prior voice activity detection model could be incorporated to alleviate the issue. As a result, for the subsequent comparisons, I use the results obtained with the normalized version to represent the Sozu downsampling approach.

5.4 Weight Prediction Submodules

Table 5.5 summarises the results of my architectural comparison experiments, with setups numbered from 0 to 8. Experiment 0 corresponds to the CIF-less RNN-T model, following the original design by Icefall in the "pruned transducer stateless7 streaming" recipe. Experiments 1 to 3 represent designs that do not utilize my proposed Ragged Attention downsampling method. Experiment 1 was proposed by [1], while Experiment 2 was introduced by [3]. Experiments 4 to 8, on the other hand, are designs that incorporate my proposed Ragged Attention downsampling method. These setups serve as a basis for comparison for the weight prediction submodule.

These results corroborate several tendencies that were observed in previous experiments, and thus, instead of reiterating those discussions, I will focus on the comparison of weight prediction submodules in this section. As a recap, the following observations were established:

Exp. No.	Design ϕ	Design ω	Add. Params	Librispeech CERs	Librispeech Sum	Librispeech Speedup	CSJ CERs	CSJ Sum	CSJ Speedup
0	CIF-less		0	4.26, 13.02	17.28	1.00	4.03, 3.42, 3.58	11.03	1.00
1	ConvFc	Cascade [1]	446,850	5.14, 14.12	19.26	**1.54**	4.50, 4.09, 3.96	12.55	1.39
2	ConvActFc	Cascade [3]	447,618	8.30, 20.18	28.48	**1.54**	4.71, 3.94, 4.49	13.14	**1.40**
3	ConvFc	Sozu	446,850	4.60, 13.55	18.15	1.50	4.46, 3.92, 3.77	12.15	1.37
4	ConvFc	R.Attn	447,234	4.43, 13.31	17.74	1.49	4.33, **3.67**, 3.96	11.96	1.34
5	ConvActFc	R.Attn	448,002	10.77, 24.95	35.72	1.52	4.45, 3.92, 4.10	12.47	1.32
6	ConvActMean	R.Attn	8,709	4.45, 13.36	17.81	1.50	4.30, 3.70, **3.65**	**11.65**	1.27
7	FcActMean	R.Attn	5,637	4.47, **13.22**	17.69	1.45	4.27, 3.80, 4.10	12.17	1.30
8	MeanAbs	R.Attn	4,097	**4.32**, 13.25	**17.57**	1.50	**4.19**, 3.94, 3.98	12.11	1.36

Table 5.5: Architectural comparison for combinations of ragged downsampling (ϕ) and weight prediction (ω). The 'Params' column indicates the additional parameters compared to the CIF-less model. The 'Speedup' column shows the RTF improvement rate, calculated as the decoding time ratio between the CIF-less model and the corresponding model in each row. CERs are provided for 'test-clean' and 'test-other' in the Librispeech column, and 'eval1', 'eval2', and 'eval3' in the CSJ column. The CER sums offer a simple indicator of the models' overall performance. The best values among models with CIF are displayed in bold.

1. Librispeech achieves significantly greater decoding speedups compared to CSJ across all setups.
2. The Sozu downsampling method consistently outperforms the Cascade downsampling in terms of accuracy (lower CER values). However, it experiences a slight degradation in decoding speed due to the final normalization operation.
3. The proposed Ragged Attention (R.Attn) downsampling method demonstrates remarkable superiority over other downsampling methods, achieving consistently lower CER values. Similarly, a slight degradation in decoding speed is observed.
4. The best-performing CIF designs, indicated by bold CER values, do not surpass the performance of the CIF-less model.

ConvActFc

ConvActFc, proposed by [3] for simultaneous speech translation, surprisingly performed the worst among its counterparts in the same downsampling method category. Comparing ConvActFc to ConvFc and ConvActMean, we observe a potential reason for its lower performance. Despite [3]'s intention to empower the weight prediction submodule with more parameters to allow the acoustic encoder to specialize in content encoding, stopping the gradient from flowing from the weight prediction submodule into the acoustic encoder has put the weight prediction submodule in passive, reactive role. Consequently, this deprived the acoustic encoder of the valuable signals provided by the Quantity Loss training objective, hindering its learning process.

ConvFc vs ConvActMean

Despite having significantly more additional parameters than ConvActmean, ConvFc did not exhibit substantial advantages in performance. Surprisingly, ConvActMean even outperformed ConvFc by 0.31 in CER sum. This observation suggests that, rather increasing the number of parameters in the weight prediction submodule, there might be an overabundance of parameters in this submodule.

FcActMean

FcActmean replaces the 1D convolutional layer of ConvActmean with a fully-connected layer, effectively reducing the context available to the weight prediction submodule. This change resulted in performance improvements for Librispeech; however, CSJ experienced a degradation. This finding suggests

that the weight prediction submodule in CSJ benefits more from a broader context than that of Librispeech.

MeanAbs

Despite its minimalist, parameter-less design, MeanAbs surprisingly performed remarkably well compared to other weight prediction submodules. The additional 4,097 parameters come from the query vector $\mathbf{q}_\theta$ of the Ragged Attenion downsampling submodule and the extended 4-context window in the RNN-T decoder module.

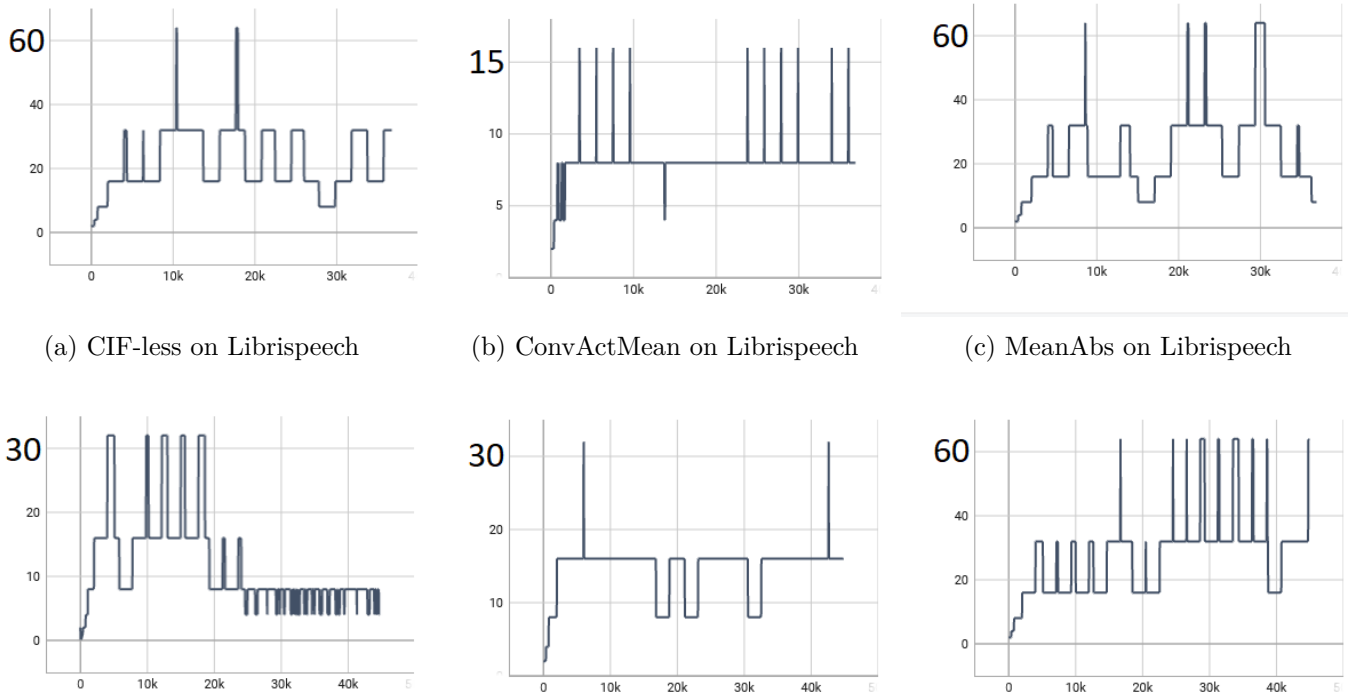
In Librispeech, MeanAbs achieved the best CER for 'test-clean', with only a 0.06 gap compared to the CIF-less model. However, for 'test-other', its CER ranked second after FcActMean. It is essential to note that the training data consists entirely of utterances from the 'clean' subset, and the slight degradation for 'test-other' could indicate MeanAbs's relative weakness in adapting to out-of-domain audio. Nevertheless, the difference (0.03) is too minor to draw definitive conclusions.

In CSJ, MeanAbs only achieved the best CER for 'eval1', falling behind other convolutional weight prediction submodules like ConvActMean and ConvFc for 'eval2' and 'eval3' by a noticeable margin. The differences in performance among these subsets may be attributed to their distinct characteristics. While 'eval1' consists solely of academic presentation recordings by male presenters, 'eval2' is a mix of academic presentations by both genders, and 'eval3' comprises spontaneous, non-academic public speaking sessions. This observation aligns with the previous finding in Librispeech, suggesting that MeanAbs's simplicity may have limited the acoustic encoder's ability to generalize effectively to out-of-domain audio.

5.4.1 Overall Trends and Future Directions

The remarkable success of the MeanAbs weight prediction submodule invites further research to optimize and explore its potential.

One intriguing avenue for improving the weight prediction submodule lies in the choice of activation function. Given that the same acoustic vector is supplied to both the ragged downsampling and weight prediction submodules, it plays a dual role: encoding the content of the utterance and encoding its weight. Incorporating an even function (symmetric around the y-axis) as the activation function for weight prediction may have helped to decouple these two tasks. Consequently, this could have enhanced the content encoding capabilities of the acoustic encoder, providing more room for expressivity in the representation of the acoustic vectors.



(a) CIF-less on Librispeech (b) ConvActMean on Librispeech (c) MeanAbs on Librispeech

(d) CIF-less on CSJ (e) ConvActMean on CSJ (f) MeanAbs on CSJ

Figure 5.2: Gradient scales as calibrated by PyTorch logged throughout 40 epochs of training.

A closer examination of the gradient scales as calibrated by PyTorch reveals a second explanation to MeanAbs's success. These gradient scales are calibrated by PyTorch's automatic mixed precision package [1], utilized in both the original Icefall recipe and all my experiments. PyTorch's gradient scaler prevents underflow (flushing small gradients to zero) and overflow (wrapping large gradients to infinity), by calibrating a scale that strikes a balance between these requirements.

Figure 5.2 illustrates the gradient scale trends compared across Librispeech and CSJ for Experiments 0 (CIF-less) on the left, 6 ("ConvActMean+R.Attn") in the middle, and 8 ("MeanAbs+R.Attn") on the right. MeanAbs exhibits noticeably larger gradient scales, indicating that more refined gradients are successfully salvaged for backpropagation, and they are less prone to explode into infinity. Conversely, ConvActMean shows smaller gradient scales when compared to the CIF-less model, suggesting that it experiences more frequent occurrences of infinity gradients. It appears that relying entirely on the acoustic encoder to embed both weight and content information while using a parameter-less operation to extract the weight information from its outputs (as done in MeanAbs) might be a promising direction for further exploration. This finding challenges conventional assumptions about the necessity of complex weight prediction submodules, and imply that simpler designs may yield superior performance in certain contexts.

5.5 Librispeech vs CSJ

Comparing the results between Librispeech and CSJ provides valuable insights into potential directions for future research, revealing two significant observations. Firstly, in Librispeech, the best performing CIF model, "MeanAbs+R.Attn", achieves a CER sum with only a marginal difference of 0.29 compared to the CIF-less model. On the other hand, the best performing CIF model in CSJ, "ConvActmean+R.Attn", falls behind the CIF-less model by a larger margin of 0.62 in CER sum. Secondly, lighter, smaller weight prediction submodules tend to perform better in Librispeech, while convolutional weight prediction submodules fare better in CSJ. I attempt to provide explanations for these observations with two distinct factors.

Effect of Fluent Transcripts

CSJ comprises mostly spontaneous, unscripted speech containing lightly uttered sounds, which may not be reflected in the fluent transcript. In contrast,

[1] https://pytorch.org/docs/stable/notes/amp_examples.html

Librispeech is a speech corpus collected from audio books, featuring calm and steady recitations, and exhibiting a more one-to-one correspondence between the transcript and the uttered words. Using a fluent transcript in CSJ can make weight prediction a more challenging task. In order to accurately decide the correct weight, the weight prediction submodule must be capable of interpreting the semantic meaning embedded within the acoustic vector. As a result, contextualizing the acoustic vector by considering neighboring acoustic vectors in CSJ proves beneficial, as it helps capture the full context of the spontaneous speech. On the other hand, the benefits from added contexts in Librispeech are outweighed by the negative impacts associated with more complex weight prediction submodules (as explained in Subsection 5.4.1.

Linguistic Differences

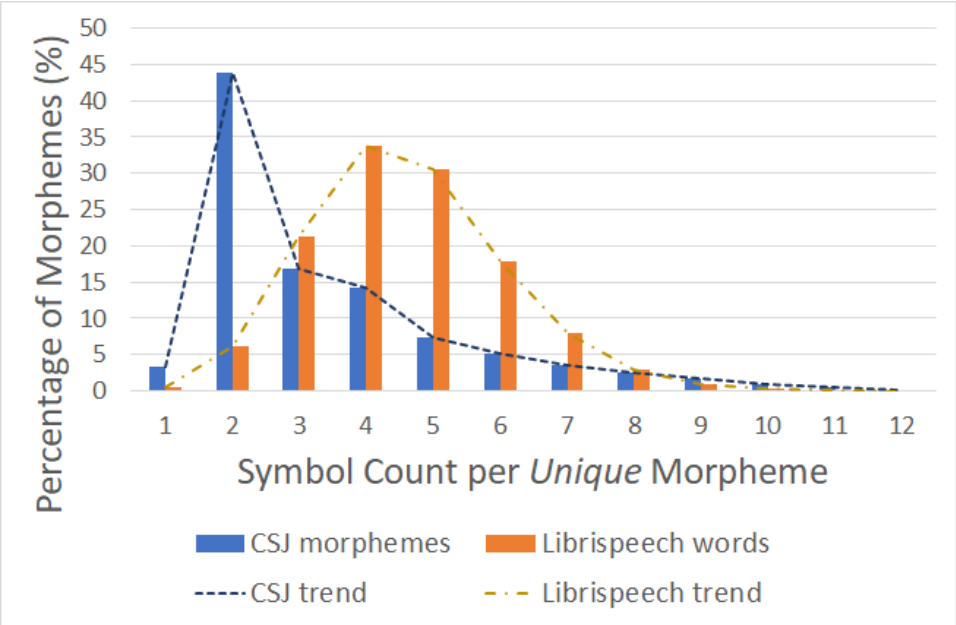


Figure 5.3: Distribution of symbol count per *unique* morpheme in the training subsets of Librispeech and CSJ

Examining the linguistic differences between Librispeech and CSJ offers insights from a different perspective. Figure 5.3 compares the number of output symbols per unique morpheme in Librispeech and in CSJ, where Librispeech uses BPE tokens, and CSJ uses characters. The chart shows that a prominent majority (44%) of CSJ morphemes have 2 characters, giving its

distribution a sharper peak, and more morphemes have a higher number of symbols (>9). In contrast, Librispeech features a more balanced distribution with a peak around 4 symbols per morpheme and tails that taper off more evenly.

To quantify the irregularity of the distribution when compared to a normal distribution, we can use statistics concepts such as skewness and kurtosis. Skewness measures the degree of asymmetry in a distribution. A positive skewness value indicates that the distribution has a longer right tail, while a negative skewness value indicates a longer left tail. Kurtosis, on the other hand, measures the heaviness of the tails in comparison to a normal distribution. Higher kurtosis values imply heavier tails, and lower values suggest lighter tails.

In the case of CSJ, its skewness of 3.85 and kurtosis of 16.47 point to a highly skewed and heavy-tailed distribution. Conversely, Librispeech exhibits a skewness of 2.36 and kurtosis of 4.64, indicating a less skewed and lighter-tailed distribution compared to CSJ. Although both datasets demonstrate sharper peaks and skewness towards higher numbers of symbols per morpheme, Librispeech's more regular distribution, as evidenced by its smaller gap with the CIF-less model, suggests greater consistency within the symbol counts per unique morpheme. Addressing this regularity may serve as a promising research direction to narrow the gap further and potentially surpass the CIF-less model's performance.

Chapter 6

Conclusion

6.1 Summary of Contributions

In this thesis, I investigated the potential of leveraging the Continuous Integrate-and-Fire (CIF) framework to compress the acoustic vectors into acoustic tokens. By introducing CIF's alignment and compression capabilities into the RNN-Transducer (RNN-T) architecture, I aimed to reduce the length of the acoustic sequence T to a more proportionate size M relative to the expected number of output symbols U. The incorporation of CIF downsampling with RNN-T training yielded significant reductions in training memory footprint and decoding speed, while minimizing performance degradation.

My contributions to this field can be summarized as follows:

1. I proposed a novel attention-based, ragged (variable-length) downsampling submodule, "Ragged Attention", which outperformed the conventional weighted sum Cascade mechanism. This submodule demonstrated superior performance in reducing acoustic sequence length while preserving essential acoustic information for ASR tasks.

2. Among several weight prediction submodules that I introduced, I highlighted the advantages of the simple, parameter-less MeanAbs over other candidates with parameters. Despite its minimalist design, MeanAbs performed remarkably well in accuracy, showcasing its potential as an effective and efficient weight prediction design.

3. I conducted comprehensive experiments with various weight perturbation strategies, investigating their impact on different CIF architectures and datasets. These strategies provided valuable insights into the effectiveness of perturbations in optimizing CIF-based architectures.

4. I reported various architecture- and language-specific tendencies observed from my results, providing explanations, supporting evidences and statistical analyses gathered from the datasets.

6.2 Future Directions

My investigations into combining the CIF framework with RNN-T models have yielded promising results and highlighted potential directions for future research in ASR. The successful integration of CIF downsampling and RNN-T training demonstrates its capability to reduce training memory and decoding time without sacrificing substantial performance. The proposed Ragged Attention downsampling submodule, in particular, proved highly effective against other downsampling submodules.

However, the performance gap between CIF-less and CIF-incorporated models remains as a notable research goal. To address this gap, I propose the following future directions:

1. **Weight Perturbation**: Further exploration of different weight perturbation strategies and their effects on various CIF architectures and datastes is necessary. Understanding the underlying dynamics that resulted in the observed tendencies will significantly propel CIF architectures to achieve better accuracy performance, not only in ASR but also in other tasks.

2. **Weight Prediction Design**: Finding the optimal design for the weight prediction submodule is crucial. Striking the right balance between simplicity and complexity, equipped with the appropriate context length and parameters, will lead to improved performance of CIF.

3. **Morpheme Definition**: Investigating statistically motivated morpheme definitions that result in a more regular symbol count per morpheme distribution could alleviate the burden on the weight prediction submodule and acoustic vector. This research direction aims to identify definitions applicable across diverse datasets and languages.

Overall, this thesis opens up exciting avenues for future research in acoustic compression, harnessing the potential of the CIF framework to optimize computational efficiency across a wide range of applications.

Chapter 7

Acknowledgement

I extend my heartfelt gratitude to my esteemed professor, Minami-sensei, for accepting me as a student, and introducing me into the rewarding, exciting field of speech processing. His unwavering belif in my abilities and relentless encouragement have been instrumental in empowering me to tackle challenging topics head-on. Under his mentorship, I gained the self-confidence to explore the most convoluted and complicated theoretical problems, making this thesis possible. The opportunities that he offered to me has been transformative, to say the least. I am eternally grateful for the positive impact he has had on my life.

I am deeply indebted to the Artificial Intelligence eXploration research centre (AIX) for providing me access to their GPU servers. Without their generous support, none of the experiments and research work presented in this thesis would have been possible. Their powerful computational resources played a vital role in carrying out extensive experiments, leading to the valuable insights achieved in my research.

Lastly, I want to express my sincere appreciation to my family in Malaysia for their unwavering support and encouragement throughout my academic pursuit for knowledge. Their love and understanding have been the cornerstone of my motivation, propelling me forward to continue my studies.

To all those who have contributed to my academic and personal growth, thank you for your guidance, encouragement, and support. Your influence has been invaluable, shaping me into the person I am today. I am profoundly grateful for the opportunities and experiences this academic endeavor has provided me, and I look forward to continuing my pursuit of knowledge in the field of speech processing.

Bibliography

[1] Linhao Dong and Bo Xu. Cif: Continuous integrate-and-fire for end-to-end speech recognition. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6079–6083, 2020.

[2] Zhiyun Fan, Linhao Dong, Meng Cai, Zejun Ma, and Bo Xu. Sequence-level speaker change detection with difference-based continuous integrate-and-fire. *IEEE Signal Processing Letters*, Vol. 29, pp. 1551–1554, 2022.

[3] Chih-Chiang Chang and Hung yi Lee. Exploring Continuous Integrate-and-Fire for Adaptive Simultaneous Speech Translation. In *Proc. Interspeech 2022*, pp. 5175–5179, 2022.

[4] Qian Dong, Yaoming Zhu, Mingxuan Wang, and Lei Li. Learning when to translate for streaming speech. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 680–694, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[5] Keyu An, Xian Shi, and Shiliang Zhang. Bat: Boundary aware transducer for memory-efficient and low-latency asr. *arXiv preprint arXiv:2305.11571*, 2023.

[6] Linhao Dong, Zhecheng An, Peihao Wu, Jun Zhang, Lu Lu, and Zejun Ma. Cif-pt: Bridging speech and text representations for spoken language understanding via continuous integrate-and-fire pre-training. *arXiv preprint arXiv:2305.17499*, 2023.

[7] Minglun Han, Linhao Dong, Shiyu Zhou, and Bo Xu. Cif-based collaborative decoding for end-to-end contextual speech recognition. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6528–6532, 2021.

[8] Minglun Han, Feilong Chen, Jing Shi, Shuang Xu, and Bo Xu. Knowledge transfer from pre-trained language models to cif-based speech recognizers via hierarchical distillation. *arXiv preprint arXiv:2301.13003*, 2023.

[9] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376, 2006.

[10] Yen Meng, Hsuan-Jui Chen, Jiatong Shi, Shinji Watanabe, Paola Garcia, Hung-yi Lee, and Hao Tang. On compressing sequences for self-supervised speech models. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pp. 1128–1135, 2023.

[11] Alex Graves. Sequence transduction with recurrent neural networks. *CoRR*, Vol. abs/1211.3711, , 2012.

[12] Wei Kang, Liyong Guo, Fangjun Kuang, Long Lin, Mingshuang Luo, Zengwei Yao, Xiaoyu Yang, Piotr Żelasko, and Daniel Povey. Fast and parallel decoding for transducer. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023.

[13] Fan Yu, Haoneng Luo, Pengcheng Guo, Yuhao Liang, Zhuoyuan Yao, Lei Xie, Yingying Gao, Leijing Hou, and Shilei Zhang. Boundary and context aware training for cif-based non-autoregressive end-to-end asr. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 328–334, 2021.

[14] Linhao Dong, Cheng Yi, Jianzong Wang, Shiyu Zhou, Shuang Xu, Xueli Jia, and Bo Xu. A comparison of label-synchronous and frame-synchronous end-to-end models for speech recognition. *arXiv preprint arXiv:2005.10113*, 2020.

[15] Linhao Dong, Feng Wang, and Bo Xu. Self-attention aligner: A latency-control end-to-end model for asr using self-attention network and chunk-hopping. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5656–5660. IEEE, 2019.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[17] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics.

[18] Daniel S Park, Yu Zhang, Chung-Cheng Chiu, Youzheng Chen, Bo Li, William Chan, Quoc V Le, and Yonghui Wu. Specaugment on large scale datasets. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6879–6883. IEEE, 2020.

[19] David Snyder, Guoguo Chen, and Daniel Povey. Musan: A music, speech, and noise corpus. *arXiv preprint arXiv:1510.08484*, 2015.

[20] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 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, 2015.

[21] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[22] Kikuo Maekawa, Hanae Koiso, Sadaoki Furui, and Hitoshi Isahara. Spontaneous speech corpus of japanese. In *International Conference on Language Resources and Evaluation (LREC)*, Vol. 2, pp. 947–952, 2000.

[23] Piotr Żelasko, Daniel Povey, Jan Trmal, and Sanjeev Khudanpur. Lhotse: a speech data representation library for the modern deep learning ecosystem. 2021.

[24] Daniel Povey, Piotr Żelasko, and Sanjeev Khundanpur. Speech recognition with next-generational kaldi (k2, lhotse, icefall), 2021.

[25] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

[26] Mohammadreza Ghodsi, Xiaofeng Liu, James Apfel, Rodrigo Cabrera, and Eugene Weinstein. Rnn-transducer with stateless prediction network. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7049–7053, 2020.

[27] Fangjun Kuang, Liyong Guo, Wei Kang, Long Lin, Mingshuang Luo, Zengwei Yao, and Daniel Povey. Pruned RNN-T for fast, memory-efficient ASR training. In *Proc. Interspeech 2022*, pp. 2068–2072, 2022.