INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Blank-regularized CTC for Frame Skipping in Neural Transducer

Yifan Yang[*,1], *Xiaoyu Yang*[*,1], *Liyong Guo*[1], *Zengwei Yao*[1],
Wei Kang[1], *Fangjun Kuang*[1], *Long Lin*[1], *Xie Chen*[†,2], *Daniel Povey*[†,1]

[1]Xiaomi Corp., Beijing, China [2]Shanghai Jiao Tong University, Shanghai, China
{yangyifan7, yangxiaoyu6, guoliyong, yaozengwei, kangwei1, kuangfangjun, linlong, dpovey}@xiaomi.com, chenxie95@sjtu.edu.cn

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 symbols 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. It is interesting to find that the frame reduction ratio of the neural Transducer can approach the theoretical boundary. Experiments on LibriSpeech corpus show that our proposed method accelerates the inference of neural Transducer by 4 times without sacrificing performance.

Index Terms: speech recognition, neural Transducer, CTC

1. Introduction

End-to-End (E2E) architectures are gaining more and more attraction in the field of automatic speech recognition (ASR). Several prominent E2E architectures are developed in recent years, such as Connectionist Temporal Classification (CTC) [1], Attention-based Encoder-Decoder (AED) [2], and neural Transducer [3]. Among these three models, CTC and neural Transducer share some common characteristics since they are frame-synchronized systems, where each acoustic frame is mapped to one or more output tokens. In contrast, label-synchronized decoding is adopted in AED, where one valid label token is generated at every step. Due to its streaming nature and superior performance in a range of tasks, the neural Transducer model receives increasing attention from both academic research and industry application. However, compared to the AED model, the decoding of the neural Transducer is more computationally expensive since it needs to handle the output of each frame in the frame-synchronized decoding.

As the input sequence of acoustic frames is typically much longer than the target label sequence, a special blank symbol corresponding to "output nothing" is introduced in the frame-synchronous RNN-T and CTC architectures. During inference, a large proportion of the acoustic frames are classified as blank frames, which can be a waste of computation. For the purpose of accelerating decoding, various studies have examined the identification of blank frames and the influence of discarding blank frames on the decoding results. Chen et al. [4] investigated the peaky posterior property of a CTC model and found that blank frames contribute little to decoding performance. Similarly, Zhang et al. [5] skipped blank labels to speed up the neural Transducer decoding process. Tian et al. [6] discarded encoder output frames based on the blank probabilities generated by a co-trained CTC to reduce the number of encoder frames going through the joiner only during inference. Similar to [6], Wang et al. [7] applied frame skipping in the middle of the shared encoder during training and inference.

The inference acceleration of the neural Transducer model is achieved by discarding blank frames the co-trained CTC predicted [6, 7]. If a larger proportion of acoustic frames can be accurately classified as blank frames by the co-trained CTC, the inference speed of the neural Transducer can be further optimized. To this end, we explored various methods to explicitly regularize the blank probabilities the CTC predicted. The CTC branch is encouraged to emit more blank frames by applying a penalty λ on consecutively repeated non-blank labels or constraining the maximum number K of consecutively repeated non-blank labels in the CTC topology. We show that the number of non-blank acoustic frames can approximate the target label tokens by adjusting λ or K. Experiments on the LibriSpeech corpus demonstrate that the neural Transducer with guidance from blank-regularized CTC yields an even lower word-error-rate (WER) than the baseline model without discarding blank frames. Our approaches achieve better trade-offs between WER and inference speed than existing methods.

To summarize, our contributions are three-fold:

- We propose two novel regularization methods to explicitly encourage the blank symbols in the co-trained CTC, which can further speed up the inference of neural Transducer.
- By applying our proposed strategies, the frame reduction ratio of the neural Transducer could even approach the theoretical boundary.
- Experimentally, we achieve an inference speedup of 4 times compared to the standard neural Transducer without sacrificing performance, and a speedup of 1.5 times over a competitive baseline [7] using frame reduction.

This work uses the k2 [8] framework[1] for modifying CTC topology and loss computation. The code is released as part of the open-source project icefall[2].

2. CTC and Neural Transducer

2.1. Connectionist Temporal Classification

CTC [1] is one of the earliest E2E ASR frameworks, which comprises an encoder and a linear decoder. To address the

* stands for equal contribution. † stands for corresponding authors.

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

length mismatch between acoustic frames and output token sequences, the output vocabulary $\mathcal{V}$ is augmented by a blank symbol $\varnothing$ representing no label emission. Given a sequence of input acoustic features $\boldsymbol{x} = (x_1, \cdots, x_T)$ of length T, the encoder produces embeddings $\boldsymbol{f} = (f_1, \cdots, f_T)$. The embeddings are then passed through the CTC decoder to generate T conditionally independent posterior probabilities $p_1, \cdots, p_T$, corresponding to the vocabulary $\mathcal{V} \cup \{\varnothing\}$. Given a ground truth label sequence, $\boldsymbol{y} = (y_1, \cdots, y_U)$ of length U, $y_u \in \mathcal{V}$, the CTC objective function is defined as the probability of all possible alignments between $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$\mathcal{L}(\boldsymbol{y}) = \sum_{\boldsymbol{\pi} \in \mathcal{B}^{-1}(\boldsymbol{y})} \log p\left(\boldsymbol{\pi} \mid \boldsymbol{x}\right), \tag{1}$$

where $\mathcal{B}(\cdot)$ is a many-to-one mapping that removes repetitive labels and blank labels in an alignment.

With the conditional independence assumption of CTC, the objective function Eqn. (1) can be approximated as:

$$\mathcal{L}(\boldsymbol{y}) \approx \sum_{\boldsymbol{\pi} \in \mathcal{B}^{-1}(\boldsymbol{y})} \sum_{t=1}^{T} \log p\left(\pi_t \mid \boldsymbol{x}\right) \tag{2}$$

Weighted Finite-State Transducer (WFST) topologies can be employed to implement CTC-like algorithms efficiently [9, 10]. A training lattice involves three graphs:

- a topology graph ($\mathbf{H}$) that acts as the map $\mathcal{B}(\cdot)$;
- a lexicon graph ($\mathbf{L}$) that maps sequences of lexicon units to words;
- a dense Finite State Acceptor ($\mathbf{Dense.FSA}$) whose weights represent the acoustic log-probabilities.

The CTC lattice can be obtained in two steps. First, the $\mathbf{H}$ and $\mathbf{L}$ are composed to create the $\mathbf{HL}$ graph, which will convert the predicted token sequences into word sequences. Then, the $\mathbf{HL}$ graph is intersected with $\mathbf{Dense.FSA}$ to produce the CTC lattice, which contains all valid alignments $\mathcal{B}^{-1}(\boldsymbol{y})$. Instead of using the traditional forward-backward algorithm [1], we can compute the total score of the CTC lattice with a differentiable dynamic programming method[3] for optimizing the CTC objective function as shown in Eqn. (2).

2.2. Neural Transducer

Neural Transducer [3] is proposed to address the conditional independence assumption in CTC, where the output probability of y_u is conditioned on all the previous tokens $\boldsymbol{y}_{\leq u-1}$. The decoder which functions like a language model is always fed with the previously emitted token. The joint network defines the probability $p(k|t, u)$ of emitting token k at time t after emitting $u - 1$ previous tokens by fusing the acoustic embedding and text embedding. Similar to CTC, neural Transducer models also maximize the probability $p(\boldsymbol{y}|\boldsymbol{x})$ (referred to as RNN-T object function) by summing over all possible alignments:

$$\mathcal{L}(\boldsymbol{y}) = \sum_{\boldsymbol{\pi} \in \mathcal{A}^{-1}(\boldsymbol{y})} \log p\left(\boldsymbol{\pi} \mid \boldsymbol{x}\right), \tag{3}$$

where $\mathcal{A}(\cdot)$ is a many-to-one mapping that only removes blank labels in an alignment.

It is worth noting that the blank symbols in CTC and neural Transducer serve very similar functions by separating two succeeding label tokens and aligning the input acoustic frames and output label tokens. There are still minor differences between these two types of blank symbols. Specifically, repeated label tokens are allowed in the CTC alignments and a blank symbol is also necessary to distinguish two consecutive identical label tokens, while in the neural Transducer, each label token is produced only once and the rest of the output tokens are blank symbols in each alignment. It is expected that the blank symbol predictions of these two models are highly relevant and well synchronized.

[3]Fsa.get_tot_scores() in https://github.com/k2-fsa/k2

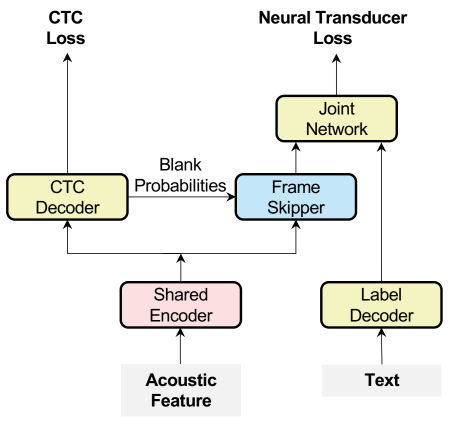


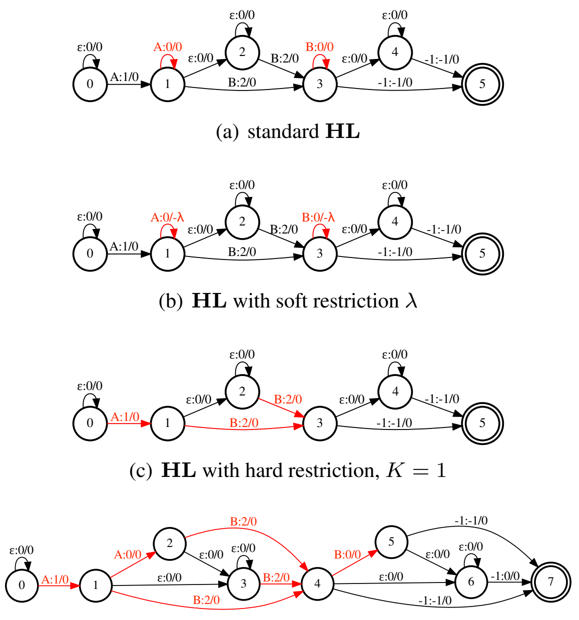
Figure 1: *System architecture of neural Transducer with a co-trained CTC branch for blank skipping.*

2.3. CTC-guided Neural Transducer

Motivated by the similar role of blank symbols in CTC and neural Transducer, several previous works attempted to utilize blank symbols from CTC to guide and simplify the inference of neural Transducer systems. Tian et al. [6] introduced a method to discard encoder output frames based on the blank probabilities generated by the CTC branch. By reducing the number of encoder output frames fed into the joiner, the decoding speed can be largely improved. However, blank skipping is not performed during training. This mismatch between the ways of processing blank frames during training and inference leads to sub-optimal performance. To achieve a coherent frame-skipping behavior between training and inference, Wang et al. [7] applied frame skipping in the middle of the shared encoder during training based on the blank probability predicted by a co-trained CTC model. During the forward pass, the CTC branch computes the blank probabilities and frames with a blank probability higher than the predetermined threshold will be excluded from the RNN-T loss computation. The authors reported a significant speed-up of training and inference with the neural Transducer model, without performance degradation.

3. Methods

Existing methods [6, 7] improve the inference speed of neural Transducers by skipping blank frames with the help of a co-trained CTC. However, few studies have explored the feasibility of skipping non-blank frames. According to the definition of $\mathcal{B}(\cdot)$ in Eqn. 1, the CTC model not only removes blank symbols but also merges consecutively repeated label tokens. Supposing that the frames emitting repeated non-blank symbols can be treated as blank frames by the CTC branch, more encoder output frames can be discarded in the neural Transducer, which will lead to a further inference speedup. Inspired by this observation, two strategies are proposed to narrow the spikes of non-blank posteriors in CTC and force the model to output fewer consecutively repeated non-blank tokens.



(a) standard **HL**

(b) **HL** with soft restriction λ

(c) **HL** with hard restriction, $K = 1$

(d) **HL** with hard restriction, $K = 2$

Figure 2: *WFST representations of CTC **HL** graph outputting the label sequence "AB" w/wo blank regularizations. An arc with the label "**a:b/s**" means the WFST consumes the input token **a** and emits the output token **b** with score **s**. Arcs entering the final state have "**-1:0/0**" as the label. For soft restriction, a penalty λ is applied over the self-loop of all non-blank symbols. For hard restriction, the maximum number K of consecutively repeated non-blank symbols is constrained. Note for $K = 1$, successive repeats of non-blank symbols are not allowed.*

3.1. Soft Restriction

The first proposed strategy employs an additional fixed penalty λ (e.g 0.05) to all non-blank self-loops in the standard **HL** graph of CTC during training, as shown in Fig. 2(a). Consequently, the alignments containing more consecutively repeated non-blank symbols will receive larger penalties, as illustrated in Fig. 2(b), and the CTC model tends to encourage alignments with fewer consecutively repeated non-blank symbols. It is worth noting that we only apply this non-blank self-loop penalty λ during the training. The proportion of the blank frames from the CTC branch can be controlled by tuning the penalty λ. This strategy is referred as *soft restriction*.

3.2. Hard Restriction

In *soft restriction*, the CTC branch yields a larger proportion of skipped frames (i.e. blank frames) by penalizing CTC alignments with consecutively repeated non-blank symbols. However, the alignments having different numbers of repeated non-blank tokens are still optimized during training as these are valid alignments by definition. Another strategy named *hard restriction* can be applied to explicitly limit the maximum number of consecutively repeated non-blank symbols K (including the first non-blank symbol) during training. The resulting WFST topologies for $K = 1$ and $K = 2$ are given in Fig. 2(c) and Fig. 2(d) respectively. As can be seen, the alignments with more than K consecutively repeated non-blank symbols are pruned from the corresponding **HL** graph.

3.3. Frame Skip

During training, the encoder output frames are filtered based on the blank probabilities $p_t^{\varnothing}$ predicted by the co-trained CTC branch. If $p_t^{\varnothing}$ surpasses a pre-defined threshold β (e.g 0.85), the t-th frame is classified as a blank frame and discarded. from the RNN-T loss computation. The set of discarded encoder output frames can be described as follows:

$$\boldsymbol{f}_{skipped} = \left\{ f_t \mid p_t^{\varnothing} \leq \beta \right\}. \tag{4}$$

During inference, a trade-off between frame reduction ratio and recognition accuracy can be tuned by adjusting the threshold β' for blank skipping.

4. Experiments

4.1. Experimental Setups

Dataset The LibriSpeech [11] corpus is used for experiments, containing 960 hours of transcribed audiobook recordings. Lhotse [12] is employed to perform data preparation. Speed perturbation [13] with factors of 0.9 and 1.1 is applied to augment the training data. SpecAugment [14] and noise augmentation [15] are utilized to improve model robustness in training. The input features are 80-channel filter bank features extracted from 25 ms windows with 10 ms shift. The classification units are 500-class Byte Pair Encoding (BPE) [16] word pieces.
System Architecture The co-trained CTC&Transducer model as shown in Fig. 1 is adopted. The shared encoder is a 12-layer Conformer [17] of dimension 512. The label decoder for the neural Transducer is a stateless decoder [18] with a dimension of 512. A convolution subsampling module of stride 4 is placed before the encoder to reduce the frame rate to 25 Hz. The model has 78.9M parameters in total.
Training Pruned RNN-T loss [19] interpolated with CTC loss [1] by factor 0.2 is used as the training objective function. Considering that the blank symbol prediction of the CTC branch is not accurate in the early stage, the frame skipping is applied after 4000 steps. A series of experiments are designed to explore the effect of hyperparameters (β, β', λ, K) in the following subsection. All models are trained with 4 NVIDIA V100 GPUs.
Evaluation The performances are evaluated on the LibriSpeech test-clean and test-other sets. Apart from Word Error Rate (WER), the frame reduction ratio (defined as $\frac{|\boldsymbol{f}_{skipped}|}{T}$) and real-time factor (RTF) are also measured as metrics for inference speed. In addition, to assess the ability of language model integration for the frame-skipped neural Transducer, WERs with external language models (LM) are also reported. Shallow fusion [20, 21, 22] and LODR [23] are investigated for LM fusion respectively. The external LM consisting of three LSTM layers [24] is trained on LibriSpeech LM corpus and the low-order N-gram in LODR is a bi-gram LM trained on 960-hour transcription of LibriSpeech. The scales of the LSTM LM and the bi-gram LM are tuned using grid search on the dev set.

4.2. Experimental Results

The results of applying different strategies to regularize blank are shown in Table 1. The baseline model is a neural Transducer&CTC co-training system without frame skipping, where the CTC loss is used as an auxiliary loss with a factor of 0.2. As a comparison with existing frame skipping methods without modifying CTC topology, three models with different thresholds used in training and inference for frame skipping are also

Table 1: *Comparisons of WER(%), Frame Reduction Ratio(%), and RTF in neural Transducer using different strategies with different $\beta/\lambda/K$.*

Method	WER↓ clean	WER↓ other	Frame Reduction Ratio↑	RTF↓
Baseline	2.45	5.93	0.00	0.0106
Threshold				
$\beta = 0.90$	2.47	6.07	65.66	0.0038
$\beta = 0.85$	2.54	5.95	66.64	0.0036
$\beta = 0.80$	2.59	6.11	68.34	0.0035
Soft Restriction				
$\lambda = 0.03$	2.48	5.91	74.92	0.0026
$\lambda = 0.04$	**2.44**	**5.88**	**75.44**	**0.0026**
$\lambda = 0.05$	2.51	6.00	75.78	0.0024
$\lambda = 5.00$	2.90	6.89	78.25	0.0023
Hard Restriction				
$K = 2$	2.49	5.92	72.35	0.0031
$K = 1$	2.92	6.96	78.23	0.0023

listed, referred as *Threshold*. The average frame reduction ratios over the test sets are also shown in Table 1. As a reference number, we calculate the maximum possible reduction ratio $\gamma_{max} = 1 - \frac{S}{T} = 78.61\%$, which is defined as the 1 minus the length ratio between output tokens and input frames.

The following observations can be made:

- Larger frame reduction ratios can be achieved after applying the 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 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 the 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.

Fig. 3 visualizes the relationship between frame reduction ratio and aggregated WER, 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' from $(0.8, 0.85, 0.9, 0.95, 0.99, 0.999)$. 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 the existing methods (Threshold-x) 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.

Figure 3: *Aggregated WER(%) versus Frame Reduction Ratio (%) curve. Aggregated WER is the summation of WER on test-clean and test-other.*

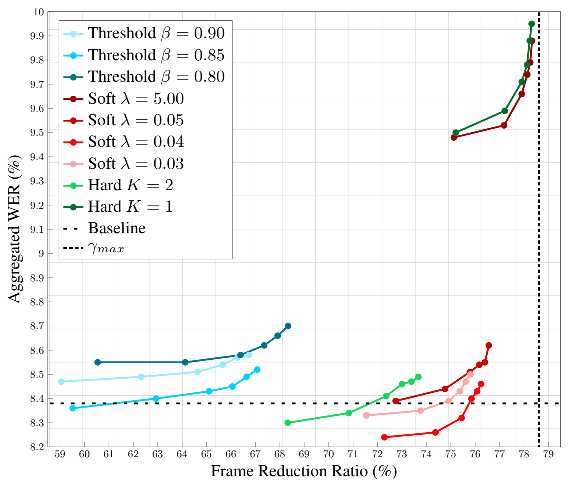


Table 2: *WERs(%) and Relative WER Improvement from LM integration.*

Method	Shallow Fusion↓ clean	Shallow Fusion↓ other	Rel. Imprv.↑	LODR↓ clean	LODR↓ other	Rel. Imprv.↑
Baseline	2.24	5.35	9.43	2.16	5.09	13.48
Threshold						
$\beta = 0.85$	2.23	5.23	12.13	2.12	5.05	15.55
Soft Restriction						
$\lambda = 0.04$	2.19	5.17	11.54	2.09	4.94	15.50
$\lambda = 5.00$	2.55	6.16	11.03	2.44	5.87	15.12
Hard Restriction						
$K = 2$	2.19	5.21	12.01	2.07	5.01	15.81
$K = 1$	2.55	6.10	12.45	2.47	5.78	16.50

The WERs of decoding with external LMs are shown in Table 2. The relative improvements of LM integration are calculated against the WERs in Table 1. Compared to the baseline model, the blank-regularized models achieve larger relative WER reduction with LM integration, suggesting that discarding more blank frames enables better fusion with external LMs.

5. Conclusions

Inspired by the observation that blank symbols in CTC and neural Transducer play similar roles, this paper proposed two novel blank-regularization methods to further boost 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 accelerate the inference of the neural Transducer. One interesting observation is that the frame reduction ratio of the neural Transducer can approach the theoretical boundary. Experiments show that the neural Transducer with guidance from the blank-regularized CTC achieves 4 times speedup during inference without sacrificing performance. Our approaches achieve better trade-offs between WER and inference real-time factors than existing methods. Additionally, a further gain can be observed when decoding with external language models.

6. References

[1] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, Pittsburgh, 2006.

[2] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. ICASSP*, Shanghai, 2016.

[3] A. Graves, A. Mohamed, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*, Vancouver, 2013.

[4] Z. Chen, W. Deng, T. Xu, and K. Yu, "Phone synchronous decoding with CTC lattice," in *Proc. Interspeech*, San Francisco, 2016.

[5] Y. Zhang, S. Sun, and L. Ma, "Tiny transducer: A highly-efficient speech recognition model on edge devices," in *Proc. ICASSP*, Toronto, 2021.

[6] Z. Tian, J. Yi, Y. Bai, J. Tao, S. Zhang, and Z. Wen, "FSR: Accelerating the inference process of transducer-based models by applying fast-skip regularization," in *Proc. Interspeech*, Brno, 2021.

[7] Y. Wang, Z. Chen, C. Zheng, Y. Zhang, W. Han, and P. Haghani, "Accelerating RNN-T training and inference using CTC guidance," in *arXiv preprint arXiv:2210.16481*, 2022.

[8] D. Povey, P. Zelasko, and S. Khudanpur, "Speech recognition with next-generation kaldi (k2, lhotse, icefall)," in *Interspeech: tutorials*, 2021.

[9] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: end-to-end speech recognition using deep RNN models and wfst-based decoding," in *Proc. ASRU*, Scottsdale, 2015.

[10] A. Laptev, S. Majumdar, and B. Ginsburg, "CTC variations through new WFST topologies," in *Proc. Interspeech*, Incheon, 2022.

[11] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *Proc. ICASSP*, South Brisbane, 2015.

[12] P. Zelasko, D. Povey, J. Y. Trmal, and S. Khudanpur, "Lhotse: a speech data representation library for the modern deep learning ecosystem," 2021.

[13] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Proc. Interspeech*, Dresden, 2015.

[14] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," in *Proc. Interspeech*, Graz, 2019.

[15] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," in *arXiv preprint arXiv:1510.08484*, 2015.

[16] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proc. ACL*, Berlin, 2016.

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

[18] M. Ghodsi, X. Liu, J. Apfel, R. Cabrera, and E. Weinstein, "Rnn-transducer with stateless prediction network," in *Proc. ICASSP*, Barcelona, 2020.

[19] F. Kuang, L. Guo, W. Kang, L. Lin, M. Luo, Z. Yao, and D. Povey, "Pruned RNN-T for fast, memory-efficient ASR training," in *Proc. Interspeech*, Incheon, 2022.

[20] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. Interspeech*, Chiba, 2010.

[21] J. Chorowski and N. Jaitly, "Towards better decoding and language model integration in sequence to sequence models," in *Proc. Interspeech*, Stockholm, 2017.

[22] A. Kannan, Y. Wu, P. Nguyen, T. N. Sainath, Z. Chen, and R. Prabhavalkar, "An analysis of incorporating an external language model into a sequence-to-sequence model," in *Proc. ICASSP*, Calgary, 2018.

[23] H. Zheng, K. An, Z. Ou, C. Huang, K. Ding, and G. Wan, "An empirical study of language model integration for transducer based speech recognition," in *Proc. Interspeech*, Incheon, 2022.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, 1997.