

Unified Streaming and Non-streaming Two-pass End-to-end Model for Speech Recognition

Binbin Zhang¹, Di Wu¹, Zhuoyuan Yao², Xiong Wang², Fan Yu², Chao Yang¹, Liyong Guo¹, Yaguang Hu¹, Lei Xie², Xin Lei¹

¹Mobvoi Inc., Beijing, China

²Audio, Speech and Language Processing Group (ASLP@NPU), School of Computer Science, Northwestern Polytechnical University, Xi'an, China

binbinzhang@mobvoi.com

Abstract

In this paper, we present a novel **two-pass approach to unify streaming and non-streaming end-to-end (E2E) speech recognition in a single model**. Our model adopts the hybrid CTC/attention architecture, in which the conformer layers in the encoder are modified. We propose a **dynamic chunk-based attention strategy** to allow arbitrary right context length. At inference time, the CTC decoder generates n-best hypotheses in a streaming way. **The inference latency** could be easily controlled by only changing the chunk size. The CTC hypotheses are then rescored by the attention decoder to get the final result. This efficient rescoring process causes very little sentence-level latency. Our experiments on the open 170-hour AISHELL-1 dataset show that, the proposed method can unify the streaming and non-streaming model simply and efficiently. On the AISHELL-1 test set, our unified model achieves 5.60% relative character error rate (CER) reduction in non-streaming ASR compared to a standard non-streaming transformer. The same model achieves 5.42% CER with 640ms latency in a streaming ASR system.

Index Terms: streaming speech recognition, two-pass, dynamic chunk, U2

1. Introduction

End-to-end (E2E) models have gained more and more attention to speech recognition over the last few years. E2E models combine the acoustic, pronunciation and language models into a single neural network, and show competitive results compared to conventional ASR systems. There are mainly three popular end to end approaches, namely CTC[1, 2], recurrent neural network transducer (RNN-T)[3, 4] and attention based encoder-decoder (AED)[5, 6, 7]. Both of them has its advantages and limitations in terms of recognition accuracy, application scenario, and many efforts have been paid for comparison of these models[8] or joint some of them into one model[9, 10].

While these models have great performance in a none streaming application, it usually requires a lot of work or a lot of accuracy degradation to make the model work in a streaming way, and a lot of works have been done for that. For RNN-T, a two pass[10, 11] method was proposed to close the accuracy gap to none streaming *Listen, Attend, Spell*(LAS) model. For AED, *Hard Monotonic Attention*[12] is first proposed for monotonically align the input and output of the AED model, then it could work in a streaming way, with the same idea, *Monotonic Chunk-wise Attention*(MoChA)[13] and *Monotonic Multihead Attention*[14] are proposed to further improve performance and stability of monotonic attention.

Recently there are also increasing interests in unifying none streaming and streaming speech recognition model into one model. Some transducer based models such as Y-model [15] and UNIVERSAL ASR[16] have been designed for this goals and achieve good performance. The unified model not only reduces the accuracy gap between the streaming model and none streaming model, but also alleviates the burden of model development, training and deployment.

In this work, we propose a new framework U2 to unify none streaming and streaming speech recognition. Our framework is based on the hybrid CTC/attention architecture with conformer blocks. The training process is simple and avoids the RNN-T model's complicated tricks and instability issues. To support streaming, We modify the conformer block while bringing just little performance degradation. In further, by using a dynamic chunk training strategy, our framework allows users to control the latency at inference time. Our results shows U2 achieves state-of-the-art streaming accuracy on Aishell-1.

2. Related Works

Hybrid CTC/attention end-to-end ASR in [9] uses both CTC and attention decoder loss in training to achieve fast convergence and improve robustness of the AED model. However, during decoding, it combines attention score and CTC score and performs joint decoding. Both of the scores can only be computed until full speech utterance is available, which makes it a none stream model.

Two pass[10] based model was proposed on RNN-T and achieves comparable accuracy to a LAS model. However, RNN-T training is very memory consuming[17] so that we can not use a large batch for training, which results in a very slow training speed as well as poor performance. Besides, RNN-T training is also unstable, CTC pre-training was proposed in [18] and *Cross Entropy*(CE) pre-training was proposed in [19] to assist RNN-T training, which is also tricky and complicated. These two reasons increase the difficulty of using RNN-T in speech recognition, especially for small companies, which lack computing and research resources. And it was pointed that training directly from scratch is unstable using combined RNN-T loss and LAS loss, so a three-step training was proposed in [10] to solve the problem, which further complicates the training pipeline.

For unified none streaming and streaming model, Y-model uses variable context at training and can use several optional contexts at inference. However, the optional contexts are pre-defined at the training stage, and the contexts are carefully designed in terms of the number of encoder layers, the kernel size

of the convolution operation. Dual-mode only has one streaming configuration in both training and inference, if we want another streaming model with different latency at inference, the model needs to be totally retrained. Besides, both Y-model and Dual-mode are RNN-T based models, they have the same drawbacks as RNN-T.

Our proposed U2, a CTC and AED based joint model, jointly trained by combined CTC and AED loss and dynamic chunk attention, not only unified the none streaming and streaming model, gives a promising result, but also significantly simplify the training pipeline, as well as dynamically control the trade-off between latency and accuracy in streaming application.

3. U2

3.1. Model architecture

The proposed two-pass architecture is shown in Figure 1. It contains three parts, a *Shared Encoder*, a *CTC Decoder* and a *Attention Decoder*. The *Shared Encoder* consists of multiple Transformer[20] or Conformer[21] encoder layers. The *CTC Decoder* consists of a linear layer and a log softmax layer, The CTC loss function is applied over the softmax output in training. The *Attention Decoder* consists of multiple Transformer decoder layers. We can make the *Shared Encoder* only see limited right contexts, then CTC decoder could run in a streaming mode in the first pass. In the second pass, the output of the *Shared Encoder* and *CTC Decoder* can be used in different ways. The training and decoding processes are detailed in the following.

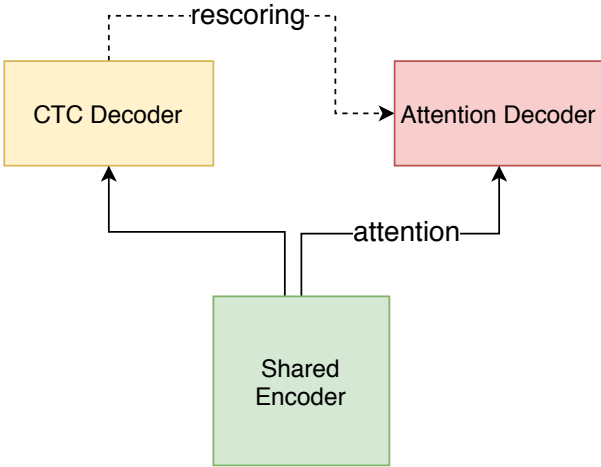


Figure 1: Two pass CTC and AED joint architecture

3.2. Training

3.2.1. Combined Loss

The training loss is combined with CTC loss and AED loss as listed in 1, where \mathbf{x} is the acoustic feature, \mathbf{y} is the corresponding annotation, $\mathbf{L}_{\text{CTC}}(\mathbf{x}, \mathbf{y})$, $\mathbf{L}_{\text{AED}}(\mathbf{x}, \mathbf{y})$ are the CTC and AED loss respectively, λ is a hyperparameter which balance the importance of CTC and AED loss. Unlike RNN-T based two pass in [10] where a three step process was used to stable the training, we can directly train our model by the combined loss from scratch, which significantly simplify our training pipeline. And As shown in [9], the combined loss also help the model con-

verge faster and have better performance.

$$\mathbf{L}_{\text{combined}}(\mathbf{x}, \mathbf{y}) = \lambda \mathbf{L}_{\text{CTC}}(\mathbf{x}, \mathbf{y}) + (1 - \lambda) \mathbf{L}_{\text{AED}}(\mathbf{x}, \mathbf{y}) \quad (1)$$

3.2.2. Dynamic Chunk Training

A Dynamic chunk training technique is proposed in this section to unify the none streaming and streaming model and enable latency control.

As described before, our U2 could only be streaming when the *Shared Encoder* is streaming. Full self attention is used in standard Transformer encoder layers, as shown in Figure 2 (a), every input at time t depends on the whole inputs, green means there is a dependency, while white means there is no dependency. The simplest way to stream it is to make the input t only see itself and the input before t , namely left attention, seeing no right context, as shown in Figure 2 (b), but there is very big degradation compared to full context model. Another common technique is to limited input t only see a limited right context $t+1, t+2, \dots, t+W$, where W is the right context for each encoder layer, and the total context is accumulated through all the encoder layers, for example, if we have N encoder layers, each has W right context, the total context is $N * W$. Right context usually improves performance compared to pure left attention, however, we should carefully design the number of layers and every right context for each layer to control the final right context for the whole model, and things get more difficult when we use conformer encoder layer, in which convolution through time with right context is used. We adopt a chunk attention in

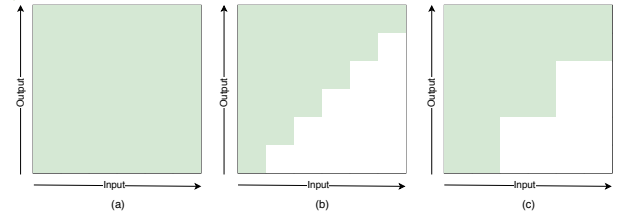


Figure 2: Full attention, Left attention, Chunk Attention

this work, as shown in Figure 2 (c), we split the input to several chunks by a fixed chunk size C , the dark green is for the current chunk, for each chunk we have inputs $[t+1, t+2, \dots, t+C]$, every chunk depends on itself and the all the previous chunks. Then the whole latency of the encoder depends on the chunk size, which is easy to control and implement. We can train the model using a fixed chunk size, we call it static chunk training, and decoding with the same chunk.

Motivated by the idea of unified E2E model, we further propose a dynamic chunk training. We can use dynamic chunk size for different batches in training, the dynamic chunk size range is a uniform distribution from 1 to max utterance length, namely the attention varies from left context attention to full context attention, and the model captures different information on various chunk size, and learns how to do accurate prediction when different limited right context provided. We call the chunks which sizes are from 1 to 25 as streaming chunk for streaming model and size which is max utterance length as none streaming chunk for none streaming model. However, the results of this method is not good enough, so next we change the distribution of chunk size during training process as follows.

$$\text{chunksize} = \begin{cases} l_{\text{max}} & x > 0.5 \\ l \sim U(1, \min(25, l_{\text{max}} - 1)) & x \leq 0.5 \end{cases} \quad (2)$$

As shown in Equation 2, x is sampled from 0 to 1.0 in each batch during the training process, l_{max} is the max utterance length of current batch, and U is a uniform distribution. So the distribution of chunk size changed, half is full chunk for none streaming, and the other half from 1 to 25 is used for streaming.

Our later experiments will show, this is a simple but efficient way, the model trained by dynamic chunk size has a comparable performance compared to static chunk training.

Besides this batch level method, we also tried epoch level - using full chunk for the first half epochs and streaming chunk for the second half or in turn. But these strategies do not work.

3.2.3. Causal Convolution

The convolution units in conformer consider both left and right context. The total right context depends on convolution layer's context and the stack number of conformer layers. So this structure not only brings in additional latency, but also ruin the benefits of chunk-based attention, that the latency is independent on the network structure and could be just controlled by chunk at inference time. To overcome this issue, we use casual convolution[15] instead.

3.3. Decoding

The *Shared Encoder* consumes the audio feature chunk by chunk. The larger chunk size usually means higher latency and better accuracy and the maximum latency is proportional to the frame number of one chunk. The proper decoding chunk size depends on specific task requirements.

The *CTC Decoder* outputs first pass hypotheses in a streaming way. At the end of the input, the *Attention Decoder* uses full context attention to get better results. Two different modes are explored here:

- Attention Decoder mode. The CTC results are ignored in this mode. *Attention Decoder* generate outputs in an auto-regressive way with the attention of the output of *Shared Encoder*.
- Rescoring mode. The n-best hypotheses from CTC are scored by the *Attention Decoder* with the output of the *Shared Encoder* in a teacher-forcing mode. The best rescored hypothesis is used as the final result. This mode avoids the auto-regressive process and achieves better real-time factor(RTF). Besides, the CTC scores could be weighted combined to get a better result in a simple way.

$$SCORES_{final} = \lambda * SCORES_{CTC} + SCORES_{attention} \quad (3)$$

In order to get a better result, ctc weighted score was added during rescoring mode decoding as shown in Equation 3, and our later experiments will show that it is always beneficial to decoding results.

4. Experiments

In order to evaluate our proposed U2, we demonstrate our experiments on the open source Chinese Mandarin speech corpus AISHELL-1[22], which contains 150 hours training set, 10 hours dev set and 5 hours test set, the test set contain 7176 utterances in total. And we use wenet¹ end-to-end speech recognition toolkit for all experiments.

¹<https://github.com/mobvoi/wenet>

We use Conformer the state-of-the-art ASR networks as our shared encoder, and the decoder part is the same as the traditional transformer decoder. Conformer adds convolution module on the basis of transformer so that it can model both local and global context capture and results in better results on different ASR tasks. As for dynamic chunk training of conformer model, causal convolution is used to instead of conventional convolution in the experiment making our en-coder is independent to right context.

4.1. AISHELL-1

For AISHELL-1, we use 80 dimensional log-mel filter bank (FBank) splice 3 dimensional pitch computed on 25ms window with 10ms shift as feature. And we do speed perturb with 0.9, 1.0 1.1 on the whole data to generate 3-fold speed changes. SpecAugment[23] is applied with 2 frequency masks with maximum frequency mask ($F = 50$), and 2 time masks with maximum time mask($T = 50$). Two convolution sub-sampling layers with kernel size 3*3 and stride 2 is used in the front of the encoder, namely 4 times sub-sampling in total. For encoder, we use 12 conformer layers with 4 multi head attention. For the *Attention Decoder*, we use 6 transformer layers with 4 multi head attention. Both transformer and conformer are use 256 attention dimension and 2048 feed forward dimension. Accumulating grad was also used to stabilize training, and we update parameters every four steps. Attention dropout, feed forward dropout and label smoothing regularization are applied in each encoder and decoder layer in order to prevent over-fitting. We use Adam optimizer and transformer learning rate schedule with 25000 warm-up steps to train models. Moreover, we get our final model by averaging the top 10 best models which have a lower loss on the dev set at the training.

Table 1: Decoding method comparison

decoding method	CTC weight	RTF	CER
attention decoder	/	0.197	5.00
ctc prefix beam search	/	/	5.02
attention rescoring	0.0	/	4.77
attention rescoring	0.5	0.082	4.72

4.1.1. Decoding Method

First, we explore different decoding methods on a none streaming model, in which full context and a conformer with standard convolution kernel size 15 are used in training, to ensure both CTC and AED decoder give a reasonable result. For AED decoder, we use beam 10 for decoding. We use prefix beam search for CTC, which is used for generating top-n different hypotheses for later rescoring.

As shown in the Table 1, the attention rescoring result outperforms both CTC prefix beam search and attention decoder results, which is out of our expectation.

After analyzing the decoding results of CTC prefix beam search and attention rescoring, we found that a lot of wrong results generated by CTC prefix beams search could be corrected by attention rescoring. However some good cases in were false corrected after attention rescoring, which means CTC plays an important role in some cases. So we added CTC weight during attention rescoring as Equation 3. And we tested different CTC weights from 0.1 to 0.9, all of them helps attention rescoring in our experiments, and 0.5 is the most stable one. Here

Table 2: *Dynamic vs Static chunk training*

training method	decoding mode	decoding chunk size				
		full	16	8	4	1
static chunk training, static chunk inference	attention decoder	5.35	5.95	5.99	6.15	6.36
	ctc prefix beam search	5.18	6.30	6.50	6.69	6.73
	attention rescoring	4.86	5.55	5.78	6.06	6.02
dynamic chunk training, static chunk inference	attention decoder	5.82	5.99	6.13	6.29	6.60
	ctc prefix beam search	5.51	6.23	6.57	6.92	7.83
	attention rescoring	5.05	5.48	5.66	5.85	6.40

we just show the result when CTC weight is 0.5, as we see, when combining with CTC weight, the CER can be further reduced to 4.72. To our knowledge, it’s the best published result on AISHELL-1. And 0.5 is the default CTC weight of attention rescoring mode in our later experiments.

Since standard attention decoder is running in an autoregressive fashion, which is time consuming, while attention rescoring just uses attention decoder for rescoring, it can be processed in parallel, and it should be faster in theory. So here we also investigate the RTF of both attention decoder and attention rescoring method, and single thread is used during decoding in Pytorch. As expected in Table 1, we got 2.40 times speed up by attention rescoring compared to attention decoder in decoding.

To conclude here, we can see the attention rescoring is both faster and more accurate.

4.1.2. Dynamic chunk evaluation

As mentioned before, causal convolution is used in dynamic chunk training to unify none streaming and streaming model, and a kernel size of 8 is used, which is half of the previous experiment since the model is limited to see left context only here.

In order to compare with static chunk training, we trained the five different models with different static chunk size full/16/8/4/1, then decode with the same chunk size as our baseline. And we trained only one unified model with the aforementioned dynamic chunk strategy as in Equation 2. The result is shown in Table 2, and we mainly pay our attention to the attention rescoring result here since it’s the final performance of our system. As we can see from the table, dynamic chunk trained model has a little degradation on full chunk and chunk 1, which are the two boundary points of the dynamic chunk with infinite latency and no latency respectively. We guess it’s more difficult to learn boundary information in the unified model. However, we see a slight gain over static chunk trained model when chunk size is 16/8/4, which means dynamic chunk strategy benefits the unified model by varying chunk training in this case.

Overall, the dynamic chunk trained model are comparable static chunk trained models, so we can easily unify the none streaming model and streaming model into one single model by our U2 framework via the two pass decoding and dynamic chunk training.

4.1.3. Comparison to other solution

Table 3 lists several published streaming solutions on AISHELL-1 test set, including Sync-Transformer[24], SCAMA[25], and MMA[14]. Δ is the additional latency introduced by attention rescoring at the end of decoding in our U2, but it’s fast enough as we have talked before it can be paralleled into one batch computing, and it’s 50-100ms as analysed in [10, 15]. We can see our U2 has far surpassed other solutions with a small additional latency.

Table 3: *Comparison to other streaming solution*

model	params(M)	latency(ms)	CER
Sync-Transformer[24]	/	400	8.91
SCAMA[25]	43	600	7.39
MMA[14]	/	640	6.60
our U2	47	640+ Δ	5.42

5. Conclusions

We propose a framework to train a single model which can do recognition in both streaming and full context way. This framework can be trained directly and stably without complicated training process. A fast weighted re-score method is used to get full-context performance with little additional latency. We also propose a dynamic chunk based strategy to improve the model performance and enable trading off the latency and accuracy conveniently at inference time.

6. References

- [1] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.
- [2] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, “Deep speech 2: End-to-end speech recognition in english and mandarin,” in *International conference on machine learning*, 2016, pp. 173–182.
- [3] A. Graves, “Sequence transduction with recurrent neural networks,” *arXiv preprint arXiv:1211.3711*, 2012.
- [4] A. Graves, A.-r. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.
- [5] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio, “End-to-end continuous speech recognition using attention-based recurrent nn: First results,” *arXiv preprint arXiv:1412.1602*, 2014.
- [6] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, “Listen, attend and spell,” *arXiv preprint arXiv:1508.01211*, 2015.
- [7] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, “Attention-based models for speech recognition,” in *Advances in neural information processing systems*, 2015, pp. 577–585.
- [8] R. Prabhavalkar, K. Rao, T. N. Sainath, B. Li, L. Johnson, and N. Jaitly, “A comparison of sequence-to-sequence models for speech recognition,” in *Interspeech*, 2017, pp. 939–943.
- [9] S. Kim, T. Hori, and S. Watanabe, “Joint ctc-attention based end-to-end speech recognition using multi-task learning,” in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2017, pp. 4835–4839.
- [10] T. N. Sainath, R. Pang, D. Rybach, Y. He, R. Prabhavalkar, W. Li, M. Visontai, Q. Liang, T. Strohman, Y. Wu *et al.*, “Two-pass end-to-end speech recognition,” *arXiv preprint arXiv:1908.10992*, 2019.
- [11] T. N. Sainath, Y. He, B. Li, A. Narayanan, R. Pang, A. Bruguier, S.-y. Chang, W. Li, R. Alvarez, Z. Chen *et al.*, “A streaming on-device end-to-end model surpassing server-side conventional model quality and latency,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6059–6063.
- [12] C. Raffel, M.-T. Luong, P. J. Liu, R. J. Weiss, and D. Eck, “Online and linear-time attention by enforcing monotonic alignments,” *arXiv preprint arXiv:1704.00784*, 2017.
- [13] C.-C. Chiu and C. Raffel, “Monotonic chunkwise attention,” *arXiv preprint arXiv:1712.05382*, 2017.
- [14] H. Inaguma, M. Mimura, and T. Kawahara, “Enhancing monotonic multihead attention for streaming asr,” *arXiv preprint arXiv:2005.09394*, 2020.
- [15] A. Tripathi, J. Kim, Q. Zhang, H. Lu, and H. Sak, “Transformer transducer: One model unifying streaming and non-streaming speech recognition,” *arXiv preprint arXiv:2010.03192*, 2020.
- [16] J. Yu, W. Han, A. Gulati, C.-C. Chiu, B. Li, T. N. Sainath, Y. Wu, and R. Pang, “Universal asr: Unify and improve streaming asr with full-context modeling,” *arXiv preprint arXiv:2010.06030*, 2020.
- [17] J. Li, R. Zhao, H. Hu, and Y. Gong, “Improving rnn transducer modeling for end-to-end speech recognition,” in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 114–121.
- [18] K. Rao, H. Sak, and R. Prabhavalkar, “Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer,” in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 193–199.
- [19] H. Hu, R. Zhao, J. Li, L. Lu, and Y. Gong, “Exploring pre-training with alignments for rnn transducer based end-to-end speech recognition,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7079–7083.
- [20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, 2017, pp. 5998–6008.
- [21] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, “Conformer: Convolution-augmented transformer for speech recognition,” *arXiv preprint arXiv:2005.08100*, 2020.
- [22] H. Bu, J. Du, X. Na, B. Wu, and H. Zheng, “Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline,” in *2017 20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA)*. IEEE, 2017, pp. 1–5.
- [23] 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,” *arXiv preprint arXiv:1904.08779*, 2019.
- [24] Z. Tian, J. Yi, Y. Bai, J. Tao, S. Zhang, and Z. Wen, “Synchronous transformers for end-to-end speech recognition,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7884–7888.
- [25] S. Zhang, Z. Gao, H. Luo, M. Lei, J. Gao, Z. Yan, and L. Xie, “Streaming chunk-aware multihead attention for online end-to-end speech recognition,” *arXiv preprint arXiv:2006.01712*, 2020.