CUSIDE: Chunking, Simulating Future Context and Decoding for Streaming ASR

Keyu An[1], *Huahuan Zheng*[1], *Zhijian Ou*[1†], *Hongyu Xiang*[2], *Ke Ding*[2], *Guanglu Wan*[2]

Speech Processing and Machine Intelligence (SPMI) Lab, Tsinghua University, China[1]
Meituan, China[2]

{aky19,zhh20}@mails.tsinghua.edu.cn, ozj@tsinghua.edu.cn,
{xianghongyu, dingke02, wanguanglu}@meituan.com

Abstract

History and future contextual information are known to be important for accurate acoustic modeling. However, acquiring future context brings latency for streaming ASR. In this paper, we propose a new framework - Chunking, Simulating Future Context and Decoding (CUSIDE) for streaming speech recognition. A new simulation module is introduced to recursively simulate the future contextual frames, without waiting for future context. The simulation module is jointly trained with the ASR model using a self-supervised loss; the ASR model is optimized with the usual ASR loss, e.g., CTC-CRF as used in our experiments. Experiments show that, compared to using real future frames as right context, using simulated future context can drastically reduce latency while maintaining recognition accuracy. With CUSIDE, we obtain new state-of-the-art streaming ASR results on the AISHELL-1 dataset.

Index Terms: Streaming ASR, Multi-task learning, Conformer.

1. Introduction

Recently, self-attention based neural networks such as Transformer [1] and Conformer [2] have shown excellent performance in automatic speech recognition (ASR) [2, 3, 4], particularly for being used as acoustic encoders to extract high-level representations from speech. Self-attention architectures capture temporal dependencies in a sequence by computing pairwise attention weights, and thus are superior in leveraging contextual information for acoustic encoders. However, the full-sequence attention mechanism involves sequence-level fully-connected computation, and thus is unsuitable for streaming ASR, where each word must be recognized shortly after it was spoken. For streaming ASR, chunk-based self-attention networks, which use chunk-level input to control the latency, are adopted in many previous works [5, 6, 7, 8, 9, 10] and show better ASR performance when compared to other streaming schemes such as causal self-attention [11].

In chunk-based latency controlled models, a certain number of left and right contextual frames are often spliced to each chunk, which is found to benefit the acoustic encoder and produce improved ASR performance [6, 7, 12, 9]. For example, Dong *et al.* [6] reported a 10%∼15% relative character error rate reduction (CERR) on HKUST dataset when splicing left and right contextual frames to each chunk, and similar results are reported in other end-to-end models [7, 9, 10]. However, the use of right contextual frames (i.e., look-ahead frames) needs to wait until the arrival of the right contextual frames, and thus brings additional latency.

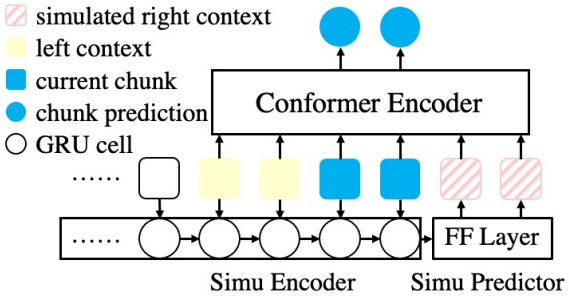

Figure 1: *The CUSIDE framework. We split the utterance into non-overlapping chunks, and iteratively run simulation encoder and predictor to simulate a certain number of future frames. Current chunk is spliced with some history frames and the simulated future frames, and they are together fed to the acoustic encoder to produce the prediction for the current chunk. The simulation encoder, predictor, and acoustic encoder are implemented in our experiments as, but generally not limited to, GRU, feed-forward (FF) layer, and Conformer respectively.*

In this paper, we introduce a new framework - Chunking, Simulating Future Context and Decoding (CUSIDE) for streaming ASR, which is illustrated in Figure 1. Inspired from predictive coding [13, 14], a new simulation module, consisting of lightweight simulation encoder and predictor, is introduced to recursively simulate the future contextual frames, based only on history frames. Thus, the ASR model does not need to wait for future context. The CUSIDE framework can be easily applied to existing ASR models via multi-task training. The simulation module is jointly trained with the ASR model using a self-supervised loss; the ASR model is optimized with the usual ASR loss, e.g., CTC-CRF [15] as used in our experiments. The CUSIDE framework can be further enhanced with several recently developed techniques for streaming ASR, such as weight sharing and joint training of a streaming model with a full-context model [16], chunk size jitter [9, 17] and stochastic future context [18] in training. Experiments demonstrate that, compared to using real future frames as right context, using simulated future context can drastically reduce latency while maintaining recognition accuracy. With CUSIDE, we obtain new state-of-the-art streaming ASR results on the AISHELL-1 dataset [19].

The paper is organized as follows. Section 2 outlines related work. Section 3 details the components of CUSIDE. Experiments are shown in Section 4. Section 5 gives the conclusion. The code will be released upon the acceptance of the paper.

† Corresponding author, also affiliated with Beijing National Research Center for Information Science and Technology, China. This work is supported by NSFC 61976122 and Tsinghua University - Meituan Joint Institute for Digital Life.

2. Related work

2.1. Low latency ASR

Building ASR models with low latency has been studied in conventional DNN-HMM hybrid systems [12]. Recently, three classes of end-to-end ASR models, namely CTC [20], RNN-T [21] and attention-based encoder-decoder (AED) [22], have demonstrated excellent performance, which raises the demand to build streaming end-to-end ASR. *For CTC and RNN-T*, both can use frame-synchronous decoding which involves no latency, thus the challenge for streaming ASR is to control the latency of the underlying acoustic encoder while maintaining recognition accuracy being close to a full-context model. Streaming acoustic encoders can be built with uni-directional recurrent networks and causal self-attention networks, which do not use any future context, but with accuracy sacrifice. Chunk-based acoustic encoders are attractive and adopted in many previous works, where bi-directional recurrent networks or fully-connected self-attention networks can be used within a chunk [6, 9, 11], realizing full-context utilization in a chunk. Truncated self-attention and time restricted self-attention [3, 23] are proposed to limit the contexts available for self-attention. *For AED*, the previously mentioned methods for low-latency acoustic encoders can be applied as well, but an additional challenge is that the decoding in AED is label-synchronous, which needs to attend to the entire sequence of representations extracted by the acoustic encoder. To address this, many efforts have been made to convert full sequence soft attention into local attention [5, 24, 25].

2.2. Unifying streaming and non-streaming ASR

Recently, there has been a growing interest in unifying streaming and non-streaming ASR, with initial motivation to simplify the workflow of training and deploying streaming and full-context ASR. Interestingly, dual-mode ASR [16], based on RNN-T, finds that weight sharing and joint training of a streaming model with a full-context model can benefit the low-latency and accuracy of streaming ASR. U2 [17], based on hybrid CTC-AED, adopts dynamic chunk sizes in training, which are uniformly drawn from 1 to some maximum size. In this manner, different AED models, varying from causal attention to full attention, are jointly trained with shared weights. Multi-mode Transformer Transducer [18] proposes to consider stochastic future contexts during training, so that the trained model is capable to serve in various latency budget scenarios without significant accuracy deterioration.

Additionally, different two-pass decoding strategies have been used to unify streaming and non-streaming ASR, such as in Universal ASR [26], U2++ [17] and RNN-T with Cascaded encoders [27]. The main idea is that during inference, a streaming model is used first to generate intermediate results (features or hypotheses), which are then further processed by a non-streaming model. The encoders can be shared [26, 17] or the decoders are shared [27]. In all the unified models mentioned above, the non-streaming model introduces extra parameters on top of the streaming model.

2.3. Predictive coding

The idea of predicting future frames has been explored in recent self-supervised speech representation learning such as autoregressive predictive coding (APC) [13] and contrastive predictive coding (CPC) [14]. Beyond predicting the next few frames by exploiting the local smoothness of speech signal, it is shown in [13, 14] that it is possible to learn more global structures in speech by neural networks so that the prediction can be as far as 200ms. This success of predictive coding motivates the introduction of the simulation module in CUSIDE to simulate future context. The difference is that these predictive coding methods adopt much larger encoders, which are trained on large amounts of unlabeled speech and used as speech representation learners for various downstream tasks such as speaker identification and speech translation. In CUSIDE, the purpose of predicting future is to provide context information for acoustic encoders for streaming ASR.

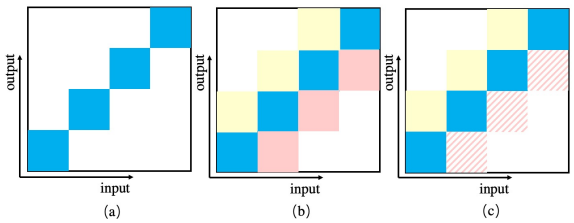

Figure 2: *The dependency map of different types of chunking used in inference. (a) Chunking w/o context. (b) Context sensitive chunking with left and right contexts, which are colored with yellow and red, respectively. (c) The proposed chunking with simulated future context. Colored [i, j] denotes whether output[j] depends on input[i]. White means there is no dependency. Dashed means that for output[j], input[i] is simulated. Here the sizes of left and right context are plotted to be equal to the sizes of the chunk, which is not necessary in practice.*

3. Method

The CUSIDE framework consists of context sensitive chunking, simulating future context and decoding. It can be easily applied to existing ASR models via multi-task training as shown below. Different neural network architectures (e.g., Bi-LSTM, Transformer and Conformer) can be used as acoustic encoders, and we can use different decoding methods, such as WFST based decoding for CTC or CTC-CRF [15], beam-search for RNN-T or AED [5, 24]. To be specific, we adopt state-of-the-art Conformer encoder [2] for acoustic modeling and WFST based decoding for CTC-CRF in our experiments.

3.1. Context sensitive chunking

Full-sequence self-attention based encoders such as Transformer and Conformer require the entire speech utterance as input, thus unsuitable for streaming ASR. To realize streaming ASR, we adopt context sensitive chunking (CSC) [12, 9] for the Conformer encoder, as illustrated in Figure 2(b). Specifically, the entire utterance is firstly split into non-overlapping chunks. For each chunk, a certain number of frames to the left and right of the chunk are spliced as contextual frames. The spliced frames are collectively called a context sensitive chunk, which together is fed into the Conformer encoder. Note that the output from contextual frames is discarded at the output layer of the Conformer. Though with some extra computation, CSC is attractive and has been shown with promising results for streaming ASR [12, 9, 6, 7], due to its use of both past and future context. However, acquiring future context brings additional latency, as the model needs to wait until the right context frames come before it can produce current output during inference. To address this drawback, we propose CSC with simulated future context for low latency ASR.

3.2. Simulating future context

As illustrated in Figure 1, in CUSIDE, the new arriving chunk is firstly encoded by a simulation encoder, which is a unidirectional GRU whose hidden state is initialized by the last hidden state from the previous chunk. The GRU hidden state $h \in \mathbb{R}^{\mathrm{d_{GRU}}}$ at the right boundary of the current chunk is then fed into a simulation predictor, which is a simple feed-forward (FF) layer, to simulate the right context frames $\widetilde{\mathbf{X}} = (\widetilde{\mathbf{x}}_1, \cdots, \widetilde{\mathbf{x}}_N)$, i.e., the log Mel-spectrograms:

$$\widetilde{\mathbf{x}}_i = h\mathbf{A}_i + b_i \tag{1}$$

where $\widetilde{\mathbf{x}}_i \in \mathbb{R}^{\mathrm{d_{Mel}}}, i = 1, \cdots, N$, N is the number the simulated right context frames, and $\mathrm{d_{Mel}}$ is the dimension of the log Mel-spectrograms. $\mathbf{A}_i$ is of shape ($\mathrm{d_{GRU}}$, $\mathrm{d_{Mel}}$), and b_i is of shape ($\mathrm{d_{Mel}}$). In this paper, to reduce the overhead of parameters and computation due to the introduction of simulating future context, lightweight GRU and feed-forward layer are used as simulation encoder and predictor, respectively.

In both training and recognition, the simulated future context $\widetilde{\mathbf{X}}$ is spliced to the chunk with the left context, as shown in Figure 2(c), which together are fed to the acoustic encoder as in usual CSC. During training, the L1 loss between the simulated future frames $\widetilde{\mathbf{X}}$ and the real future frames $\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_N)$ is added to the ASR loss with a scaling factor α. Thus, the simulation module is trained by the usual (supervised) ASR loss and the (self-supervised) simulation loss.

3.3. Weight sharing and joint training with full-context model

Inspired by dual-mode ASR [16], we train a single ASR model with shared weights for both streaming and non-streaming ASR. Notably, the forward calculation in the streaming mode of CUSIDE is actually in full-context within each context sensitive chunk, thus can be performed using exactly the same parameters and operations as in non-streaming ASR. In contrast, the forward calculation in the streaming mode of dual-model ASR uses causal operators such as causal attention and convolution, which are different from non-causal operators used in the non-streaming mode. And the non-streaming model introduces extra parameters on top of the streaming model. Moreover, we adopt chunk size jitter for different batches in training, which can improve the generalization capability of the streaming ASR model as shown in [9, 17]. Specifically, the chunk size for every batch is sampled from a uniform distribution $\mathcal{U}$(C-A, C+A), where C is the default chunk size and A is the dynamic range.

The whole training in CUSIDE as shown in Algorithm 1 is in multi-task training, which involves three tasks - simulation future context, streaming and non-streaming ASR. We simulate future context for all chunks at the same time, and use the simulated frames and the ground-truth future frames (obtained by shifting the real frames backward) to calculate the simulation loss. For the streaming ASR loss, after chunk-wise forward calculation, we reshape the chunk-wise posterior predictions to construct the utterance-level posterior predictions, which are used to compute the streaming ASR loss. The non-streaming ASR loss is calculated with Conformer encoder as usual.

Algorithm 1 Pseudocode of model training in CUSIDE.

```
# Requires: data_loader, AcousticEncoder, SimuEncoder and SimuPredictor.
for X, y in data_loader do # Load the speech feature X and the label y.
    non_streaming_pred = AcousticEncoder(X) # Perform full-context recognition.
    non_streaming_loss = asr_loss(non_streaming_pred, y) # Calculate utterance-level non-streaming loss.

    simulated_context = SimuPredictor(SimuEncoder(X)) # Simulate the future context.
    simu_loss = L1Loss(simulated_context, Shift(X)) # Calculate simulation loss. Shift(X) outputs the ground-truth future context, by shifting X backward.

    [chunk_k]_1^K = format_chunk(X, simulated_context) # Construct K context sensitive chunks with simulated future context. K = ⌈len(X)/chunk_size⌉. The chunk size can be randomly drawn. We use zero padding for the last chunk.
    [chunk_pred_k]_1^K = AcousticEncoder([chunk_k]_1^K) # Perform batch forward for all chunks and discard output from contextual frames.
    streaming_pred = format_utt([chunk_pred_k]_1^K) # Construct the utterance-level posterior prediction for the streaming model.
    streaming_loss = asr_loss(streaming_pred, y) # Calculate utterance-level streaming loss.

    total_loss = non_streaming_loss + streaming_loss + α*simu_loss # Compute total loss.
    total_loss.backward() # Update parameters.
end for
```

4. Experiments

4.1. Experiment settings

We use the CTC-CRF based ASR Toolkit - CAT [9] to conduct the experiments. CTC-CRF [15] is a CRF with CTC topology, which eliminates the conditional independence assumption in CTC and performs significantly better than CTC. Experiments are conducted on Aishell-1 [19], which consists of a total of 178 hours of labeled speech. The official lexicon [1] is used.

All experiments use 80-dim filterbank features with 3-dim pitch features, which are extracted with a 25ms sliding window at a 10ms frame rate. We apply online mean and variance normalization to the input feature. 3-fold speed perturbation and SpecAugment [28] are used for data augmentation. Unless otherwise specified, the acoustic model is a 12-layer Conformer, and the number of attention heads, attention dimension, and feed-forward dimension are 4, 256, and 2048 respectively. For simulating the future context, we use a 3-layer GRU with 256 hidden units, and 1 feed forward layer. The GRU and feed forward layer have about 5% of the total model size ($\sim$ 37M). Thus, the parameter overhead is negligible.

In training, we use the Adam optimizer and follow the Transformer learning rate scheduler [1]. The learning rate will decay with a factor of 0.1 if the loss does not decrease on the validation set. We stop training when the learning rate is less than a given threshold. The final model averages the top 5 checkpoints with the best validation loss on the development set. Gradient clipping is applied to stabilize the training. We further adopt a curriculum learning strategy in the first several training epochs, i.e., we sort the input utterances from short to long, as model training tends to be easier over short sequences. The scaling factor α for the simulation loss is set to 100. In decoding, a word-level 3-gram language model (LM) trained on the transcripts is used to build the decoding WFST.

[1] https://www.openslr.org/resources/33/resource_aishell.tgz

Table 1: *CER and average time cost per chunk on AISHELL-1 test set. For the right context, [ ] denotes that these frames are simulated. For all experiments, the chunk size is 400ms by default. ctx is shorthand for context.*

Exp	left ctx (ms)	right ctx at training (ms)	right ctx at inference (ms)	CER
1	400	400	400	6.09
2	400	0	0	7.27
3	400	[400]	[400]	7.28
4	400	400 or 0	0	7.15
5	400	400 or 0 or [400]	[400]	**6.14**
6	800	400	400	6.05
7	800	0	0	6.82
8	800	[400]	[400]	6.73
9	800	400 or 0	0	5.94
10	800	400 or 0 or [400]	[400]	**5.83**

4.2. Experiment results

The Character Error Rates (CERs) and average time costs per chunk when using different contextual information are shown in Table 1. From the table we observe that: 1) For the streaming ASR model, the lack of right context leads to a significant degradation in recognition accuracy (Exp 1 vs Exp 2, and Exp 6 vs Exp 7), and the accuracy degradation can not be compensated with more left context (Exp 1 vs Exp 7); 2) Using stochastic future context at training yields better performance at inference for the models without right context (Exp 4 and Exp 9) and with simulated right context (Exp 5 and Exp 10). 3) With stochastic future context, CUSIDE (Exp 5 and Exp 10) outperforms the model without right context (Exp 4 and Exp 9) consistently, with only a small overhead over parameter and time cost[2], especially when the CSC uses fewer left frames as history information. Notably, CUSIDE obtains equally good, or even better results compared with the model using real right context.

The results of CUSIDE with chunk size jitter, weight sharing and joint training with a full-context model are shown in Table 2. During training, the chunk size is randomly sampled from $\mathcal{U}$(200ms, 600ms) with C=400ms and A=200ms. The results demonstrate the effectiveness of these two training strategies. Notably, the performance of our streaming model is very close to the full-context model (5.47 vs 5.28). It is found that weight sharing and joint training also improve the non-streaming model (results in the parentheses). Presumably, this is because that the full-context model tends to overfit the training data, which could be mitigated by joint training with a chunk based model.

We further compare the accuracy and latency of CUSIDE with other models from literature. Notably, some existing streaming models use two-pass decoding. For comparison, a 3-layer Transformer LM on the training transcripts in AISHELL-1 is used for rescoring. The results are shown in Table 3. CUSIDE obtains state-of-the-art results on AISHELL-1 with minimal overall latency.

4.3. Discussion

To observe the quality of the simulated frames, we visualize the simulated log Mel-spectrogram in Fig 3. It can be seen that the simulated result roughly follows the spectrogram patterns, e.g., the different magnitudes of the speech and non-speech segments, but may not be able to capture finer details, e.g., the formant peaks in middle and high frequencies. As we only use limited data to train the simulation module, further improving the simulation module such as self-supervised pretraining on larger datasets would be an interesting extension of our work.

[2]2ms per chunk in our experiments when testing on a GeForce GTX 1080 GPU.

Table 2: *Effect of chunk size jitter, weight sharing and joint training with a full-context model. The numbers in the parentheses are the full-context recognition results of the unified model.*

model configuration	CER
Exp 5 in Table 1	6.14
+ chunk size jitter in training	5.96
+ weight sharing & joint training	5.83 (4.98)
Exp 10 in Table 1	5.83
+ chunk size jitter in training	5.72
+ weight sharing & joint training	5.47 (5.01)
full-context model	5.28

Table 3: *Comparison with other streaming models from literature on AISHELL-1 test set. Following [17], the latency is defined as the chunk size plus the right context (if any). Δ is the additional latency introduced by rescoring the first-pass hypotheses, which is typically less than 100ms for a utterance.*

model	latency (ms)	CER
SCAMA	600	7.39
MMA narrow chunk	960	7.5
MMA wide chunk	1920	6.6
HS-DACS Transformer	1280	6.8
U2++	640	5.81
U2++ w/ rescoring	640 + Δ	5.05
WNARS w/ rescoring	640 + Δ	5.22
CUSIDE	400 + 2	5.47
+ NNLM rescoring	400 + 2 + Δ	4.79

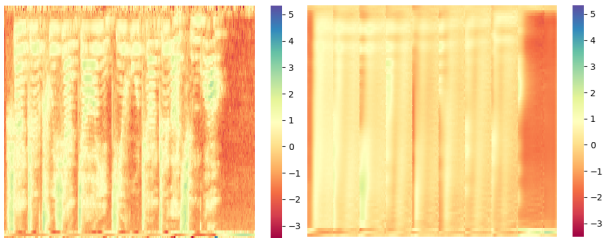

Figure 3: *The real spectrogram (left, with mean and variance normalized) and the simulated spectrogram (right) for the utterance (BAC009S0764W0121) in the AISHELL-1 test set.*

5. Conclusions

In this paper, we introduce the CUSIDE framework, which uses simulated future context to improve the recognition accuracy of chunk-based streaming ASR. With CUSIDE, we achieve new state-of-the-art streaming ASR results on AISHELL-1 in terms of accuracy and latency. While we use Conformer based CTC-CRF as our ASR model, our approach can be applied to other end-to-end ASR models such as RNN-T and attention based encoder-decoder, which will be interesting future work.

6. References

[1] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[2] A. Gulati, C.-C. Chiu, J. Qin, J. Yu, N. Parmar, R. Pang, S. Wang, W. Han, Y. Wu, Y. Zhang, and Z. Zhang, "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. INTERSPEECH*, 2020, pp. 5036–5040.

[3] C.-F. Yeh, J. Mahadeokar, K. Kalgaonkar, Y. Wang, D. Le, M. Jain, K. Schubert, C. Fuegen, and M. Seltzer, "Transformer-transducer: End-to-end speech recognition with self-attention," *arXiv preprint arXiv:1910.12977*, 2019.

[4] L. Dong, S. Xu, and B. Xu, "Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition," in *Proc. ICASSP*, 2018, pp. 5884–5888.

[5] 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," in *Proc. INTERSPEECH*, 2020, p. 2142–2146.

[6] L. Dong, F. Wang, and B. Xu, "Self-attention aligner: A latency-control end-to-end model for asr using self-attention network and chunk-hopping," in *Proc. ICASSP*, 2019, pp. 5656–5660.

[7] H. Miao, G. Cheng, C. Gao, P. Zhang, and Y. Yan, "Transformer-based online CTC/attention end-to-end speech recognition architecture," in *Proc. ICASSP*, 2020, pp. 6084–6088.

[8] E. Tsunoo, Y. Kashiwagi, T. Kumakura, and S. Watanabe, "Transformer asr with contextual block processing," in *Proc. ASRU*, 2019, pp. 427–433.

[9] K. An, H. Xiang, and Z. Ou, "CAT: A CTC-CRF based ASR toolkit bridging the hybrid and the end-to-end approaches towards data efficiency and low latency," in *Proc. INTERSPEECH*, 2020, p. 566–570.

[10] C. Wu, Y. Wang, Y. Shi, C.-F. Yeh, and F. Zhang, "Streaming Transformer-Based Acoustic Models Using Self-Attention with Augmented Memory," in *Proc. INTERSPEECH*, 2020, pp. 2132–2136.

[11] N. Moritz, T. Hori, and L. J. Roux, "Dual causal/non-causal self-attention for streaming end-to-end speech recognition," in *Proc. INTERSPEECH*, 2021, pp. 1822–1826.

[12] K. Chen and Q. Huo, "Training deep bidirectional lstm acoustic model for lvcsr by a context-sensitive-chunk bptt approach," in *Proc. INTERSPEECH*, 2015.

[13] Y.-A. Chung and J. Glass, "Generative pre-training for speech with autoregressive predictive coding," in *Proc. ICASSP*. IEEE, 2020, pp. 3497–3501.

[14] A. van den Oord, Y. Li, and O. Vinyals, "Representation Learning with Contrastive Predictive Coding," *arXiv preprint arXiv:1807.03748*, 2018.

[15] H. Xiang and Z. Ou, "CRF-based single-stage acoustic modeling with CTC topology," in *Proc. ICASSP*, 2019, pp. 5676–5680.

[16] J. Yu, W. Han, A. Gulati, C.-C. Chiu, B. Li, T. N. Sainath, Y. Wu, and R. Pang, "Dual-mode ASR: Unify and improve streaming ASR with full-context modeling," in *Proc. ICLR*, 2021.

[17] D. Wu, B. Zhang, C. Yang, Z. Peng, W. Xia, X. Chen, and X. Lei, "U2++: Unified two-pass bidirectional end-to-end model for speech recognition," *arXiv preprint arXiv:2106.05642*, 2021.

[18] K. Kim, F. Wu, P. Sridhar, K. J. Han, and S. Watanabe, "Multi-Mode Transformer Transducer with Stochastic Future Context," in *Proc. INTERSPEECH*, 2021, pp. 1827–1831.

[19] 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 *Proc. O-COCOSDA*, 2017, pp. 1–5.

[20] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006, p. 369–376.

[21] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[22] 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*, 2016, pp. 4960–4964.

[23] N. Moritz, T. Hori, and J. Le, "Streaming automatic speech recognition with the transformer model," in *Proc. ICASSP*, 2020, pp. 6074–6078.

[24] C.-C. Chiu and C. Raffel, "Monotonic chunkwise attention," in *Proc. ICLR*, 2018.

[25] L. Dong and B. Xu, "CIF: Continuous integrate-and-fire for end-to-end speech recognition," in *Proc. ICASSP*, 2020, pp. 6079–6083.

[26] Z. Gao, S. Zhang, M. Lei, and I. Mcloughlin, "Universal ASR: Unifying streaming and non-streaming ASR using a single encoder-decoder model," *arXiv preprint arXiv:2010.14099*, 2020.

[27] A. Narayanan, T. N. Sainath, R. Pang, J. Yu, C.-C. Chiu, R. Prabhavalkar, E. Variani, and T. Strohman, "Cascaded encoders for unifying streaming and non-streaming ASR," in *Proc. ICASSP*, 2021, pp. 5629–5633.

[28] 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*, 2019, pp. 2613–2617.