INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

DCTX-Conformer: Dynamic context carry-over for low latency unified streaming and non-streaming Conformer

Goeric Huybrechts[1], *Srikanth Ronanki*[1], *Xilai Li*[1], *Hadis Nosrati*[2], *Sravan Bodapati*[1], *Katrin Kirchhoff*[1]

[1]AWS AI Labs, USA
[2]AWS AI Labs, Australia

huybrech@amazon.com, ronanks@amazon.com, lixilai@amazon.com, hnosr@amazon.com, sravanb@amazon.com, katrinki@amazon.com

Abstract

Conformer-based end-to-end models have become ubiquitous these days and are commonly used in both streaming and non-streaming automatic speech recognition (ASR). Techniques like dual-mode and dynamic chunk training helped unify streaming and non-streaming systems. However, there remains a performance gap between streaming with a full and limited past context. To address this issue, we propose the integration of a novel dynamic contextual carry-over mechanism in a state-of-the-art (SOTA) unified ASR system. Our proposed dynamic context Conformer (DCTX-Conformer) utilizes a non-overlapping contextual carry-over mechanism that takes into account both the left context of a chunk and one or more preceding context embeddings. We outperform the SOTA by a relative 25.0% word error rate, with a negligible latency impact due to the additional context embeddings.

Index Terms: end-to-end speech recognition, unified streaming, Conformer, low latency, dynamic context carry-over

1. Introduction

Recently, end-to-end automatic speech recognition (ASR) systems such as attention-based encoder-decoder [1, 2], CTC [3–5] and Transducer [6, 7] have become popular due to their simplicity in combining pronunciation, language and acoustic models into one neural network. Although these state-of-the-art (SOTA) models perform well in full-contextual (i.e. non-streaming) situations, they experience a decline in performance when used in real-time streaming scenarios due to the lack of future context [8–11]. Likewise, performance degrades to an even greater extent when streaming with a limited instead of a full past context [8–11].

These two sources of streaming performance degradation hold both for purely streaming [8–13] and the more recent unified ASR systems [14–20]. The latter systems unify streaming and non-streaming into a single model, which helps reduce development, training and deployment cost. While some studies like [21] have explored solutions to mitigate the lack of future context, we focus on the challenge of limited past context as we are particularly interested in low latency systems. Similarly, we constrain ourselves to unified ASR systems for their advantages highlighted earlier.

A commonly explored solution to overcome the limitation of a restricted past context in block-processing models [8] is to store and propagate the history context. Wu et al. [22] introduce a context-aware inheritance mechanism in the self-attention layers of a Transformer [23]. A context embedding is appended as extra frame to each chunk before the self-attention and is handed over from one chunk/layer to another to help encode not only local acoustic information but also global linguistic, channel and speaker attributes. Related works consider multiple history embeddings with the introduction of memory banks. The self-attention unit of the Augmented Memory Transformer [24] attends on a short segment of the input sequence and a bank of memories that stores the embedding information from all previous processed segments. In Emformer [25], the long-range history context is distilled into an augmented memory bank in between the self-attention and feed-forward layers. While all these works made some great progress, there still exists a gap between streaming with a full and limited past context. Plus, none of these works have considered the unified streaming and non-streaming setting.

In this work, we tackle the streaming performance degradation for unified ASR systems when using a limited chunk's left context. We propose the dynamic context Conformer (DCTX-Conformer) that builds on and enhances the SOTA with next 5 contributions: (1) We incorporate the contextual carry-over (CCO) mechanism of [22] in a SOTA unified ASR Conformer system [20]; (2) We improve upon the CCO mechanism by integrating a dynamic dependency on a chunk's left context; (3) We improve upon the CCO mechanism by adding a dynamic dependency on the number of context embeddings; (4) We conduct experiments using the dynamic CCO mechanism in the lower latency, non-overlapping streaming mode without any look-ahead frames; (5) We conduct an exhaustive experimental study of our model on different chunk sizes, various chunk's left contexts and multiple context embeddings. The results on numerous datasets and many different settings demonstrate the effectiveness and robustness of our proposed model.

2. Approach and related work

In this work, we improve upon the CCO mechanism of [22] and integrate it in a SOTA unified ASR Conformer system [20]. We consider a joint CTC [3]-attention framework [26, 27] for training our unified models.

2.1. End-to-end unified ASR

For unified ASR models to perform well in both streaming and non-streaming settings, they must be exposed to both limited (i.e. streaming) and full (i.e. non-streaming) contexts during training. To accomplish this, [18, 20] propose a dynamic chunk training (DCT) for self-attention layers which involves varying the chunk size dynamically at training time. As in [20], we randomly sample a chunk size between 8 (= 320ms) and 32 (= 1280ms) down-sampled self-attention frames 60% of the time and run a full-contextual training the remaining 40%. Moreover, we use the same dynamic left context mechanism that allows to vary the left context between zero and all preceding chunks so that the model becomes robust to numerous left

10.21437/Interspeech.2023-1129

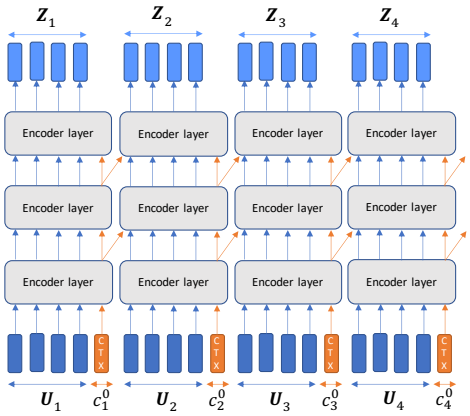

Figure 1: *Contextual carry-over mechanism using non-overlapping chunks.*

context sizes at inference time. The downside of [20] and other (unified) streamable systems is that there remains a non-negligible gap between streaming with a full and limited past context. To overcome this limitation, we integrate our proposed dynamic CCO mechanism that we discuss in next subsection.

2.2. Dynamic contextual carry-over mechanism

The authors of [22] extend the streaming Transformer model with a context-aware inheritance mechanism (Fig. 1). Context embeddings are passed on from one layer/chunk to the next. An experimental study in [22] shows that taking the average of each chunk as context embedding in the first layer provides the best results. As opposed to the original work [22], we integrate the CCO mechanism in a unified Conformer trained with dynamic chunk sizes. The Conformer is generally considered to be a better alternative than the Transformer for (unified) streaming ASR [28], while the DCT approach makes the model more robust to different chunk sizes at inference time. Lastly, unlike [20, 22] that perform block processing with overlapping chunks, we stream in the lower latency non-overlapping manner. Besides these differences in architecture and applications, we make 2 contributions to the mechanism.

Firstly, we propose to keep a dynamic dependency on preceding chunks despite the presence of context embeddings. We demonstrate that this combination leads to significant performance gains for insignificant latency drops. The dynamic dependency consists in varying the chunk's left context size at training time. The context embedding is in that case handed over from the chunk that precedes the left context chunks and no longer from the one directly preceding the current chunk. This dynamic training trick allows to vary the amount of left context needed at inference time depending on the application's latency requirements. Fig. 2 shows the design of the self-attention mask in the use-case of 4 non-overlapping chunks of size 4 with the left context size set to 1 chunk. We formalise the novel dynamic CCO process next. Let $\mathbf{U}_i$ denote the down-sampled chunks that are passed to the first self-attention layer, and c_i denote the corresponding context embeddings. The attention computation takes the embedding dimension d, queries $\mathbf{Q}$, keys $\mathbf{K}$, values $\mathbf{V}$ and *Mask* of Fig. 2 as input:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = Softmax(\frac{Mask(\mathbf{Q}\mathbf{K}^T)}{\sqrt{d}})\mathbf{V}, \quad (1)$$

with the *Mask* adapting $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ as follows:

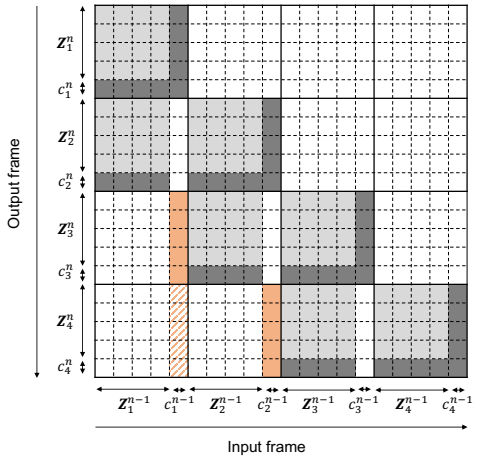

Figure 2: *Contextual carry-over mask. Illustration for 4 non-overlapping chunks of size 4 with left context size set to 1 chunk. Non-white squares = 1, white squares = 0. Light gray = frame to frame dependency, dark gray = context embedding involved. Orange squares represent the context carried over from past context embeddings in the encoder layers $n > 1$.*

For layer 1:

$$\mathbf{Q}_b^1 = [\mathbf{U}_b, c_b^0], \quad (2)$$

$$\mathbf{K}_b^1 = \mathbf{V}_b^1 = [\mathbf{U}_{b-LC:b}, c_b^0], \quad (3)$$

where b denotes the chunk (or block) number, LC denotes the number of left context chunks and $\mathbf{U}_{b-LC:b}$ denotes the concatenation of chunks $\mathbf{U}_{b-LC}$ to $\mathbf{U}_b$ (included).

For layer n > 1:

$$\mathbf{Q}_b^n = [\mathbf{Z}_b^{n-1}, c_b^{n-1}], \quad (4)$$

$$\mathbf{K}_b^n = \mathbf{V}_b^n = [c_{b-LC-1}^{n-1}, \mathbf{Z}_{b-LC:b}^{n-1}, c_b^{n-1}], \quad (5)$$

where $\mathbf{Z}_b^n$ is the output of the n^{th} encoder layer of chunk (or block) b.

Secondly, inspired by the idea of memory banks [24, 25], we show further improvements by relying on more than one preceding context embedding at inference time. For instance, in Fig. 2, output chunk #4 would depend on all frames of chunks #3 and #4 and context embeddings #1 (dashed orange squares), #2 (orange squares) and #4 (dark gray squares). This additional dependency on the preceding context embedding #1 is set at inference time only, as opposed to [24, 25]. The self-attention for chunk $b > LC + 1$ and layer $n > 1$ is still trained relying on one preceding context embedding c_{b-LC-1}^{n-1} only (i.e. the dashed orange squares are set to 0 at training time). If we assume to depend on N_{ctx} preceding (successive) context embeddings, the calculation of the keys and values becomes:

$$\mathbf{K}_b^n = \mathbf{V}_b^n = [c_{b-LC-N_{ctx}:b-LC-1}^{n-1}, \mathbf{Z}_{b-LC:b}^{n-1}, c_b^{n-1}] \quad (6)$$

Unlike [24,25], our queries only depend on the actual chunk and not on any patched left or right context. Plus, our keys and values do not contain any redundant history information as we only include the context embeddings that summarize the context before the chunk's patched left context. Similarly to the keys, they do not contain any patched right context. More importantly, in [24, 25] memory bank entries are recomputed at every layer as an average projection of the chunk. In our CCO mechanism, context embeddings are only initialised as a chunk's average in the first layer. This gives the model more freedom to

learn superior contextual representations in subsequent layers. Every intermediate contextual embedding c_b^n also explicitly depends on context embeddings c_{b-LC-1}^{n-1} and c_b^{n-1}, allowing to better model interactions between them than the memory banks in [24, 25] where this explicit interaction does not exist.

3. Experimental settings

3.1. Datasets

Training We consider 3 different speech corpora varying in size for training our models: (1) The open-source *LibriSpeech* [29] corpus, for which we combine *train-clean-100*, *train-clean-360* and *train-other-500* to have 960 hours of training data; (2) A *small-scale* 1k hour English corpus and (3) a *large-scale* 10k hour superset, sampled from in-house paired audio and text data. Both corpora include audio files with a good mix of accents, speakers, sampling rates and background noise. These 3 data regimes are representative of a wide range of end-to-end ASR systems for various speech applications.

Evaluation For the *LibriSpeech* experiments, we evaluate our models on *test-clean* and *test-other*, whose average utterance length is 7s. For the *small-* and *large-scale* experiments we use the following diverse public test sets: (1) *MTDialogue*[1]: Collection of movie and Twitter data. The dataset is 1.2h long and the average utterance length is 3s; (2) *Wall Street Journal (WSJ)*: We use WSJ's eval_test92 [30], prepared using Kaldi's [31] WSJ recipe. The dataset is 0.7h long and the average utterance length is 8s; (3) *Voxpopuli* [32]: We use the English test partition. The dataset is 4.9h long and the average utterance length is 10s.

3.2. Setup

Training We use a Conformer as the encoder and a shallow single-layer Transformer as the attention-based decoder. The inputs are 80 dimensional log-mel features extracted with 25ms FFT windows and 10ms frame shifts. For the *LibriSpeech* experiments, we use a Conformer-12x512x8, which consists of 12 encoder layers with 512 feature dimensions and 8 self-attention heads. We use ESPnet's pre-trained 24-layered Transformer-based neural LM on the *LibriSpeech-train* dataset for rescoring. For the *small-scale* experiments, we use a Conformer-16x256x4 and a BPE embedding of size 1024. For the *large-scale* experiments, we use a Conformer-16x512x8 and a BPE embedding of size 2048. We train a 4-gram LM on the training text for shallow fusion. The kernel size of our convolution modules is 31. We optimise our model via the hybrid CTC and attention losses. All of our models are trained for 60 epochs with the Adam optimizer [33] and a warm-up learning rate scheduler. The unified ASR models are fine-tuned with DCT for 30 epochs from a full-contextual model trained for 30 epochs, as is done in [20]. We make use of ESPnet [34] and p4de.24xlarge Amazon EC2 instances that consist of 8 NVIDIA A100 Tensor Core GPUs.

Evaluation We discard the attention-based decoder and use the CTC decoder to generate outputs with a CTC prefix beam search and beam size of 20 for the *LibriSpeech* experiments and of 50 for the *small-* and *large-scale* experiments. A CTC decoder optimises the real-time factor compared to the attention-based decoder, as the latter is non-autoregressive and needs triggered attention [35] for streaming inference. All of our streaming results are obtained via non-overlapping streaming without any look-ahead context, unless otherwise stated.

[1] https://github.com/Phylliida/Dialogue-Datasets

4. Results

4.1. Performance impact of chunk size

In Table 1, we analyze the impact of the chunk size on the word error rate (WER) by comparing 3 types of models on 5 test sets: (A) A purely non-streamable model trained in a full-contextual setting (i.e. no DCT); (B) A unified SOTA model [20] without CCO and with full (B.1) and no (B.2) left context at inference time; (C) Our unified model with dynamic CCO and no left context at inference time. All models share the same architecture and were trained on the *small-scale*, *large-scale* and *LibriSpeech* datasets. The results show that our model (C) roughly maintains the full-contextual performance of models (A) and (B), except for the LibriSpeech *test-other* dataset, while significantly reducing the streaming performance gap between (B.1) and (B.2). We observe that the relative improvements increase as the chunk size decreases because context embeddings are increasingly impactful for smaller and therefore inferior acoustic representations. For the *large-scale* model with inference chunk size 320ms, we even exceed the gap. Our hypothesis is that context embeddings provide a more effective way of incorporating past context than utilizing all frames from a full past context input. Moreover, we believe that the larger training set leads to a better modelling of those extra embeddings. Their use also significantly reduces the computational memory over non-contextual models that stream with full past context as the number of keys and values in the self-attention calculations decreases substantially. Overall, we notice an average gap reduction of 24.3%, 49.5% and 109.2% across all datasets and models between no and full past context when streaming with a chunk size of 1280ms, 640ms and 320ms respectively.

4.2. Performance impact of left context size

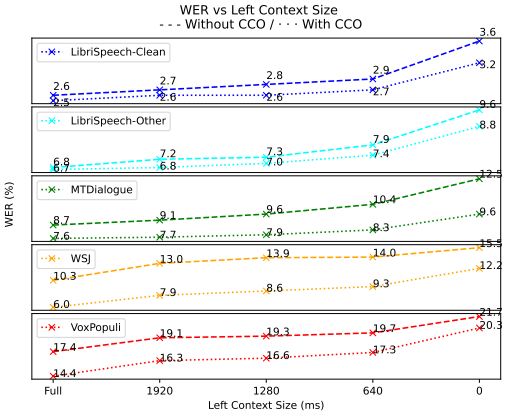

Figure 3: *WER in function of left context (ms) for a chunk size of 640ms without look-ahead frames for model without and with context carry-over.*

As can be observed in Table 1, the streaming gap between the model without CCO and with full left context (B.1) and our model with dynamic CCO and with no left context (C) can still be further reduced in most settings. In Fig. 3, we analyze the impact of adding a chunk's left context on top of the CCO mechanism for the *LibriSpeech* and *large-scale* models when running inference with a chunk size of 640ms. The graphs indicate that the aforementioned gap not only further decreases as we add more left context, but also that with only 1 left context chunk we now outperform the *large-scale* model (B.1) with full left con-

Table 1: *WER for full-contextual setting and for non-overlapping streaming without look-ahead frames in function of chunk size (ms).*

Model	Test set	LibriSpeech				Train set	MTDialogue				WSJ				VoxPopuli			
		Full	1280	640	320		Full	1280	640	320	Full	1280	640	320	Full	1280	640	320
(A) Full-contextual, LC=0		2.1	22.5	49.4	89.0		9.5	14.8	23.9	46.8	7.8	18.3	31.9	57.9	14.5	26.4	42.8	68.8
(B.1) Conformer w/o CCO, LC=Full		2.1	2.4	2.6	3.1		9.3	9.8	10.4	16.6	8.2	9.0	11.3	35.4	14.7	16.8	21.4	56.0
(B.2) Conformer w/o CCO, LC=0	Test-clean	2.1	2.8	3.6	5.4	Small	9.3	10.3	14.2	48.7	8.2	10.7	18.3	80.5	14.7	19.5	34.6	85.9
(C) **DCTX-Conformer, LC=0**		2.3	2.6	3.2	4.4		9.4	10.7	13.1	25.1	8.3	10.5	14.4	38.6	14.8	19.0	25.3	63.5
(A) Full-contextual, LC=0		5.1	29.5	58.2	92.7		7.0	12.2	20.0	39.8	5.2	13.2	24.5	50.9	10.2	19.0	33.9	60.8
(B.1) Conformer w/o CCO, LC=Full		4.9	6.0	6.8	8.2		6.8	7.2	8.7	22.0	5.1	5.7	10.3	43.9	10.1	11.6	17.4	57.2
(B.2) Conformer w/o CCO, LC=0	Test-other	4.9	7.4	9.6	14.8	Large	6.8	8.4	12.5	33.2	5.1	8.4	15.5	58.0	10.1	14.9	21.7	75.0
(C) **DCTX-Conformer, LC=0**		6.0	6.8	8.8	12.8		6.7	7.9	9.6	16.8	5.0	6.5	12.2	27.2	10.1	13.6	20.3	40.5

text (as we already did in Table 1 with our *large-scale* model for a chunk size of 320ms and left context size 0). Taking a closer look at *WSJ* in particular, we observe that taking 3 left context chunks instead of none while still carrying over context leads to a WERR of 35.2% and 49.0% over the model with and without context carry-over respectively. As our models are trained with a dynamic left context, it gives the user the flexibility to easily adjust the left context size at inference time depending on the latency and memory requirements.

4.3. Performance impact of context embeddings

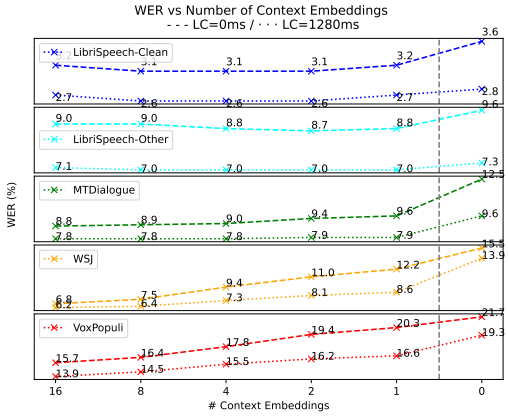

Figure 4: *WER in function of number of context embeddings for a chunk size of 640ms without look-ahead frames and with left context of 0ms and 1280ms. For 0 context embeddings, we consider the baseline model without context carry-over.*

In Fig. 4, we showcase that relying on more than 1 preceding context embedding substantially improves the WER. We observe those improvements despite the fact that the model was trained with 1 preceding context embedding only. The more context embeddings we provide each chunk, the better the model performs, except for the *LibriSpeech* test sets where a minor degradation of 0.9% is observed across the two depicted left context settings when using 16 context embeddings instead of 1. For *WSJ* and *VoxPopuli* on the other hand, we observe an average WERR improvement of 36.1% and 19.5% respectively. Compared to the baseline Conformer model without CCO those numbers even increase to 55.8% and 27.8%. For *MTDialogue*, we notice a faster saturation and only notice slight improvements beyond 4 context embeddings because the average utterance length in that dataset is small (= 3s). Therefore, for most utterances the model can only rely on a few past context embeddings. Overall, we demonstrate that our approach leads to an average 25.0% WERR improvement over the non-contextual baseline model across the datasets and the two depicted left context settings when using 16 context embeddings.

Table 2: *LibriSpeech WER comparison with SOTA look-ahead context overlapping streaming models.*

Model	LC	CC	RC	N_{ctx}	# Params	Clean	Other
AM-Transformer [24]	640	1280	320	Inf	80M	2.8	6.7
Emformer [25]	1280	640	320	4	120M	2.6	**6.0**
Conformer w/o CCO [20]	1280	**320**	320	0	115M	2.5	6.6
DCTX-Conformer, w/o RC	1280	640	**0**	2	115M	2.6	7.0
DCTX-Conformer, w/ RC	1280	**320**	320	2	115M	**2.4**	6.4

4.4. LibriSpeech comparison with SOTA

In Table 2, we compare our model (with 2 context embeddings) to the Augmented Memory Transformer [24] (with infinite memory bank entries and *WAS* [36]), the Emformer hybrid system [25] (with 4 memory bank entries and *SMBR* [37]) and the baseline Conformer model without CCO [20] on the *LibriSpeech* test sets. We provide numbers given in the respective papers in similar settings, where numbers were however given for the higher latency overlapping streaming mode with look-ahead/right context (RC). As demonstrated in [9], while no look-ahead context improves latency, it significantly degrades the WER on the *LibriSpeech* test sets. Despite this, our overall smaller segment (=LC+CC+RC) and our model being trained in a unified fashion, our lower latency model performs equally well on *test-clean* and is competitive on *test-other* wrt the SOTA. When we do use look-ahead context, we even outperform every SOTA model on *test-clean* and are just behind Emformer on *test-other*, even though our model did not see any look-ahead context during training and uses only half of the Emformer center context (CC) size.

4.5. Latency study

In Table 3, we provide some latency results in function of the number of context embeddings using the *small-scale* models, a 640ms chunk, a 1280ms left context and the *VoxPopuli* dataset. The measurements indicate the negligible latency impact of the context embeddings.

Table 3: *Latency in ms in function of number of context embeddings for model without and with context carry-over.*

	Conformer w/o CCO	**DCTX-Conformer**		
# Ctx embeddings	0	1	8	16
Mean	816	820	820	821
P99	954	960	960	961

5. Conclusions

In this work, we incorporate an improved version of the contextual carry-over mechanism in a state-of-the-art unified ASR system. We modify the contextual carry-over mechanism by integrating a dynamic dependency on both the chunk's left context size and preceding context embeddings. With an exhaustive experimental study on many datasets, we show the efficacy and robustness of our proposed approach. The results demonstrate that our DCTX-Conformer model more effectively captures a full past context with reduced latency and computational memory usage in streaming scenarios, without compromising its non-streaming performance.

6. References

[1] 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*. IEEE, 2016, pp. 4960–4964.

[2] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," *Advances in neural information processing systems*, vol. 28, 2015.

[3] 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.

[4] 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*. PMLR, 2016, pp. 173–182.

[5] S. Dingliwal, M. Sunkara, S. Ronanki, J. Farris, K. Kirchhoff, and S. Bodapati, "Personalization of ctc speech recognition models," in *IEEE Spoken Language Technology Workshop (SLT)*, 2023, pp. 302–309.

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

[7] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*. IEEE, 2013, pp. 6645–6649.

[8] 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*. IEEE, 2019, pp. 5656–5660.

[9] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, and S. Kumar, "Transformer Transducer: A streamable speech recognition model with Transformer encoders and RNN-T loss," in *Proc. ICASSP*. IEEE, 2020, pp. 7829–7833.

[10] W. Huang, W. Hu, Y. T. Yeung, and X. Chen, "Conv-Transformer Transducer: Low latency, low frame rate, streamable end-to-end speech recognition," *Proc. Interspeech*, pp. 5001–5005, 2020.

[11] 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*. IEEE, 2020, pp. 6084–6088.

[12] C.-C. Chiu and C. Raffel, "Monotonic chunkwise attention," in *International Conference on Learning Representations*, 2017.

[13] 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," *Proc. Interspeech*, pp. 2773–2777, 2019.

[14] 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.

[15] K. Hu, T. N. Sainath, R. Pang, and R. Prabhavalkar, "Deliberation model based two-pass end-to-end speech recognition," in *Proc. ICASSP*. IEEE, 2020, pp. 7799–7803.

[16] B. Zhang, D. Wu, Z. Yao, X. Wang, F. Yu, C. Yang, L. Guo, Y. Hu, L. Xie, and X. Lei, "Unified streaming and non-streaming two-pass end-to-end model for speech recognition," *arXiv preprint arXiv:2012.05481*, 2020.

[17] 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 *International Conference on Learning Representations*, 2021.

[18] Z. Yao, D. Wu, X. Wang, B. Zhang, F. Yu, C. Yang, Z. Peng, X. Chen, L. Xie, and X. Lei, "Wenet: Production oriented streaming and non-streaming end-to-end speech recognition toolkit," *arXiv preprint arXiv:2102.01547*, 2021.

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

[20] X. Li, G. Huybrechts, S. Ronanki, J. Farris, and S. Bodapati, "Dynamic chunk convolution for unified streaming and non-streaming Conformer ASR," in *Proc. ICASSP*. IEEE, 2023.

[21] K. An, H. Zheng, Z. Ou, H. Xiang, K. Ding, and G. Wan, "Cuside: Chunking, simulating future context and decoding for streaming ASR," *arXiv preprint arXiv:2203.16758*, 2022.

[22] E. Tsunoo, Y. Kashiwagi, T. Kumakura, and S. Watanabe, "Transformer ASR with contextual block processing," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 427–433.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[24] C. Wu, Y. Wang, Y. Shi, C.-F. Yeh, and F. Zhang, "Streaming Transformer-based acoustic models using self-attention with augmented memory," *Proc. Interspeech*, pp. 2132–2136, 2020.

[25] Y. Shi, Y. Wang, C. Wu, C.-F. Yeh, J. Chan, F. Zhang, D. Le, and M. Seltzer, "Emformer: Efficient memory transformer based acoustic model for low latency streaming speech recognition," in *Proc. ICASSP*. IEEE, 2021, pp. 6783–6787.

[26] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *Proc. ICASSP*. IEEE, 2017, pp. 4835–4839.

[27] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid CTC/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[28] 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," *Proc. Interspeech*, pp. 5036–5040, 2020.

[29] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. ICASSP*. IEEE, 2015, pp. 5206–5210.

[30] J. Garofolo, D. Graff, D. Paul, and D. Pallett, "Csr-i (wsj0) complete ldc93s6a," *Web Download. Philadelphia: Linguistic Data Consortium*, vol. 83, 1993.

[31] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The Kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. CONF. IEEE Signal Processing Society, 2011.

[32] C. Wang, M. Rivière, A. Lee, A. Wu, C. Talnikar, D. Haziza, M. Williamson, J. Pino, and E. Dupoux, "VoxPopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation," in *ACL 2021-59th Annual Meeting of the Association for Computational Linguistics*, 2021.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[34] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen *et al.*, "Espnet: End-to-end speech processing toolkit," *arXiv preprint arXiv:1804.00015*, 2018.

[35] N. Moritz, T. Hori, and J. Le Roux, "Triggered attention for end-to-end speech recognition," in *Proc. ICASSP*. IEEE, 2019, pp. 5666–5670.

[36] Y. Shi, Y. Wang, C. Wu, C. Fuegen, F. Zhang, D. Le, C.-F. Yeh, and M. L. Seltzer, "Weak-attention suppression for Transformer based speech recognition," *Proc. Interspeech*, pp. 4996–5000, 2020.

[37] K. Veselỳ, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," in *Proc. Interspeech*, 2013, pp. 2345–2349.