**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\(^3\), Katrin Kirchhoff\(^4\)

\(^1\)AWS AI Labs, USA  
\(^2\)AWS AI Labs, Australia

huybrech@amazon.com, ronanks@amazon.com, lixilai@amazon.com, hnosr@amazon.com, sravanb@amazon.com, kattrinki@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 consuming automatic speech recognition 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 store 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
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 $U_b$ denote the down-sampled chunks that are passed to the first self-attention layer, and $c_n$ the corresponding context embeddings. The attention computation takes the embedding dimension $d$, queries $Q$, keys $K$, values $V$ and Mask of Fig. 2 as input:

$$\text{Attention}(Q, K, V) = \text{Softmax} \left( \frac{\text{Mask}(QK^T)}{\sqrt{d}} \right) V,$$  \hspace{1cm} (1)

with the Mask adapting $Q$, $K$ and $V$ as follows:

$$Q^n = [U_{b_n}, c_n^{b_n}],$$  \hspace{1cm} (2)

$$K^n_b = V^n_b = [U_{b_n-LC+b_n}, c_n^{b_n}],$$  \hspace{1cm} (3)

where $b$ denotes the chunk (or block) number, $LC$ denotes the number of left context chunks and $U_{b_n-LC+b_n}$ denotes the concatenation of chunks $U_{b_n-LC}$ to $U_b$ (included).

For layer $n > 1$:

$$Q^n_{b_n} = [Z^{n-1}_{b_n}, c_n^{b_n-1}],$$  \hspace{1cm} (4)

$$K^n_b = V^n_b = [Z^{n-1}_{b_n-LC+b_n}, c_n^{b_n-1}],$$  \hspace{1cm} (5)

where $Z^n_b$ 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 $c_{b_n-LC-1}^{b_n}$ 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:

$$K^n_{b_n} = V^n_{b_n} = [Z^{n-1}_{b_n-LC-N_{ctx}+b_n-LC-1}, Z^{n-1}_{b_n-LC+b_n}, c_n^{b_n-1}],$$  \hspace{1cm} (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 $c^n_i$ also explicitly depends on context embeddings $c_{n-1-LC-1}$ and $c_{n-1}$, 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 follow the following diverse public test sets: (1) MDTDialogue:\footnote{\url{https://github.com/Phyllida/Dialogue-Datasets}}: 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 shallowing the kernel size of our convolutional modules is 31. We optimise our model via the hybrid CTC and attention losses. 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 p4dc24large 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.

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 CCO); (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, where 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

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-
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 WERR of 35.2% and 49.0% over the model with and without chunks instead of none while still carrying over context leads to look at a chunk size of 320ms and left context size 0. Taking a closer look at A WERR 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.
6. References


