VAD-free Streaming Hybrid CTC/Attention ASR for Unsegmented Recording

Hirofumi Inaguma, Tatsuya Kawahara
Graduate School of Informatics, Kyoto University, Kyoto, Japan
{inaguma,kawahara}@sap.ist.i.kyoto-u.ac.jp

Abstract
In this work, we propose novel decoding algorithms to enable streaming automatic speech recognition (ASR) on unsegmented long-form recordings without voice activity detection (VAD), based on monotonic chunkwise attention (MoChA) with an auxiliary connectionist temporal classification (CTC) objective. We propose a block-synchronous beam search decoding to take advantage of efficient batched output-synchronous and low-latency input-synchronous searches. We also propose a VAD-free inference algorithm that leverages CTC probabilities to determine a suitable timing to reset the model states to tackle the vulnerability to long-form data. Experimental evaluations demonstrate that the block-synchronous decoding achieves comparable accuracy to the label-synchronous one. Moreover, the VAD-free inference can recognize long-form speech robustly for up to a few hours.

Index Terms: Streaming automatic speech recognition, monotonic chunkwise attention, CTC, voice activity detection

1. Introduction
Recent progress of end-to-end (E2E) automatic speech recognition (ASR) enables us to build competitive systems to conventional hybrid systems with much smaller development efforts. For live streaming applications, frame-synchronous models such as connectionist temporal classification (CTC) [1] and RNN transducer (RNN-T) [2] are promising approaches because of the robustness for long-form speech [3–4]. Attention-based encoder-decoder (AED) [5–6] have shown outstanding performances in the offline task [7–10] and have been intensively investigated for streaming extensions [10–13]. Among them, monotonic chunkwise attention (MoChA) [11] is attractive because of the monotonic constraint of alignments and linear-time decoding complexity at test time. The notable advantages of MoChA over frame-synchronous models are faster decoding and low-latency input-synchronous searches. We also propose a VAD-free streaming ASR model trained jointly with the CTC objective.

Experimental evaluations on English and Japanese lecture corpora demonstrate that the block-synchronous decoding achieves comparable accuracy to the label-synchronous decoding and even outperforms it in some cases. We also show that the VAD-free inference does not degrade accuracy so much without the ground-truth segmentation and achieves better performance than cascading an external VAD model.

2. Streaming Hybrid CTC/Attention ASR
MoChA extended hard monotonic attention (HMA) [10] by equipping an additional chunkwise soft attention module restricted to local w frames. To generate the i-th token with a linear-time complexity at test time, HMA introduces a discrete decision $z_{i,j} \in \{0, 1\}$ ($j$: encoder time index) and samples it from a Bernoulli random variable, Bernoulli($p_{i,j}$), where $p_{i,j} \in [0, 1]$ is a selection probability as a function of encoder and decoder outputs. To enable the backpropagation training, the expected alignment score $\alpha_{i,j}$ is calculated with $p_{i,j}$ by considering all alignment paths as

$$\alpha_{i,j} = \alpha_{i,j} \left( (1 - p_{i,j-1}) \frac{\alpha_{i-1,j}}{p_{i,j-1}} + \alpha_{i-1,j} \right).$$ (1)

The chunkwise attention score for a context vector is calculated with $\alpha_{i,j}$ at training time and with discrete indices at test time. We also apply StableEmit [32] to reduce the emission latency by multiplying $p_{i,j}$ in Eq. (1) by a constant factor $1 - \lambda_{se}$ ($\lambda_{se} > 0$) during training.

Applying an auxiliary CTC loss $L_{ctc}$ on top of the encoder of AED models is effective in encouraging the decoder to learn

Instead of pursuing better generalization to long-form speech from a training perspective, we seek a solution to find a suitable timing to reset the model states from a decoding perspective. Firstly, we propose a block-synchronous beam search decoding, in which the advantages of breadth- and best-first searches are taken to achieve efficient batched inference and low display latency. We allow continuing search within a block by relaxing the label-synchronous hypothesis pruning to consider (potentially) various lengths of candidates in the beam given a partial observation. Secondly, we propose a VAD-free streaming inference algorithm leveraging CTC probabilities to determine a timing to reset the states. Instead of performing audio segmentation before ASR, our method recognizes all speech, including silence frames, and therefore the ASR model does not have to wait for the segmentation to be completed. Moreover, the unified framework is suitable for context management and on-device applications. Although our base model is MoChA, the proposed VAD-free inference can also be applied to any streaming ASR model trained jointly with the CTC objective.
a monotonic alignment [26, 27, 33, 34]. We also introduce a quantity loss \( L_{\text{mon}} \) or a CTC-synchronous training (CTC-ST) loss \( L_{\text{sync}} \) [17] to improve the performance and reduce the emission latency [14, 32]. The total objective \( L_{\text{total}} \) is formulated as

\[
L_{\text{total}} = (1 - \lambda_{\text{txt}}) L_{\text{mocha}} + \lambda_{\text{txt}} L_{\text{ctc}} + \lambda_{\text{lm}} L_{\text{lm}} + \lambda_{\text{sync}} L_{\text{sync}},
\]

where \( \lambda \) is a corresponding task weight. Unlike [27], we do not perform joint CTC decoding during beam search because they were not helpful in our experiments.

3. Efficient block-synchronous decoding

When using MoChA, beam search decoding is conducted in a label-synchronous way (i.e., breadth-first search) at test time to find the most probable output sequence. However, active hypotheses in the current beam are pruned after the expansion when and only when (1) all the hypotheses find the next token boundaries (i.e., \( j \) s.t. \( z_{i,j} = 1 \)) or (2) their pointers to the encoder outputs reach the last encoder output observed so far. In other words, all the active hypotheses must have the same output sequence length at each output step. Therefore, the search cannot proceed forward if some active hypotheses fail to detect the next boundaries correctly, even when a new acoustic observation comes in. Moreover, when applying subword tokenization (e.g., byte pair encoding (BPE) [35]) to word sequences, hypotheses in the beam could have different lengths even when they correspond to the same word sequence. This problem becomes more serious when recognizing long-form speech, resulting in a non-negligible recognition delay in the online streaming scenario. So our goal is to continue sequence generation as long as some of the active hypotheses detect the subsequent token boundaries over the current acoustic observation.

3.1. Proposed algorithm

To perform beam search with a minimal display latency given a partial acoustic observation, we propose an efficient block-synchronous beam search decoding for MoChA, which relaxes the constraint of the label-synchronous hypothesis pruning. The block-synchronous decoding combines the advantages of breadth- and best-first searches, efficient batched computation [29] and small display latency. The proposed algorithm is shown in Algorithm 1. Given the \( m \)-th input block \( x^m \) of a fixed length \( T_{\text{block}} \) [10ms], we perform the breadth-first search over the corresponding encoder outputs \( h^m \) of length \( T_{\text{block}} \) (< \( T_{\text{block}} \)). Unlike label-synchronous decoding, however, active hypotheses in the current beam are forcibly pruned no matter whether all active hypotheses detect the next boundaries in \( h^m \). The search in the \( m \)-th block continues until (1) none of the active hypotheses find any further boundary in \( h^m \) (line 5) or (2) the number of generated tokens in \( h^m \) surpasses \( U_{\text{max}} = (T_{\text{block}} \times R_{\text{len}}) / 2 \).

Let \( \Omega_k \) be a set of hypotheses having a possibility to detect the next boundary in \( h^m \). Hypotheses in \( \Omega_k \) are added to \( \Omega_\text{next} \) without prefix expansion if any next boundary is not detected in \( h^m \) (line 7). In this case, we allow to generate \( \langle \text{eos} \rangle \) only because of MoChA’s behavior (see Algorithm 1 in [11]). Otherwise, it is expanded by the top-\( k \) tokens and is added to \( \Omega_\text{next} \) (line 8). When \( \langle \text{eos} \rangle \) is generated, the hypothesis is added to a

\[
\begin{align*}
L_{\text{total}} = (1 - \lambda_{\text{txt}}) L_{\text{mocha}} + \lambda_{\text{txt}} L_{\text{ctc}} + \lambda_{\text{lm}} L_{\text{lm}} + \lambda_{\text{sync}} L_{\text{sync}},
\end{align*}
\]

complete hypothesis set \( \Omega_\text{mocha} \) instead (line 76). Pruning is conducted over \( \Omega_\text{mocha} \) with a beam width \( B \) at every output step (line 22). To avoid biasing to shorter hypotheses because of the monotonic decrease of sequence-level log probabilities, we normalize them by the current output sequence length (length normalization [37]).

The entire search process for an utterance or a session is finalized when all pointers to \( h^m \) in \( \Omega_\cdot \) reach the last encoder output, \( h^m_{\text{len} - 1} \). The details will be described in Section 4. VAD-free streaming inference

In the AED models, it is crucial to keep decoding contexts to improve the recognition performance because of the conditional dependency of output symbols. However, it is required to exclude long samples from the training data to fit the GPU/TPU memory and perform efficient training. Therefore, when recognizing long-form speech during inference, the models must generalize to unseen samples, but it is challenging in general [4].

This is more effective than pruning over \( \Omega_\text{mocha} \cup \Omega_\cdot \).

Algorithm 1 Block-synchronous beam search decoding with MoChA at the \( m \)-th block

1: function BLOCKSYNCH\((h^m, \Omega_\text{sync}, B, R_{\text{len}})\)
2: \( U_{\text{max}} \leftarrow |h^m| \times R_{\text{len}}, \Omega_\text{next} \leftarrow \{}\)
3: for \( i = 1, \ldots, U_{\text{max}} \) do
4: if \( \Omega_\cdot = \{} \) then \>
5: break \>
6: end if
7: \( p_{\text{mocha},i} = \text{Decoder}(\Omega_\cdot, [h_{\text{len} - 1}], h^m]\)
8: \( \text{score} = \log p_{\text{mocha},i} + \lambda_{\text{lm}} \log p_{\text{lm},i} \) \>
9: Normalize by length
10: \( \Omega_\text{next} \leftarrow \{}\)
11: for \( y \in \Omega_\cdot \) do
12: if \( \sum_j z_{i,j} = 0 \) then
13: add \( y \) to \( \Omega_\cdot \) \>
14: No boundary detected \>
15: end if
16: for \( k \in \mathcal{Y} \) do
17: if \( k = \langle \text{eos} \rangle \) then
18: add \( y \) to \( \Omega_{\text{eos}} \)
19: else
20: end for
21: end for
22: \( \Omega_\cdot \leftarrow \text{top-} B \text{ in } \Omega_\text{next} \) \>
23: end for
24: \( \Omega_\cdot \leftarrow \Omega_\cdot \cup \Omega_{\text{eos}} \)
25: return \( \Omega_\cdot, \Omega_{\text{eos}} \)
26: end function

The search is equivalent to frame-synchronous and label-synchronous decoding by setting \( T_{\text{block}} \) to 1 and \( \infty \), respectively. Therefore, the proposed algorithm is a generalized form of both search methods.

Concurrently to this work, streaming block-synchronous decoding with an offline Transformer decoder was proposed in [13]. They determined to move to the next block when generating \( \langle \text{eos} \rangle \) at the current block and introduced complicated heuristics to avoid token repetition. The decoding complexity was quadratic of the input length because of incremental decoding. In contrast, our method can move to the next block without heuristics to avoid token repetition. The decoding complexity is linear and, moreover, the decoding complexity of our method is linear, and we do not perform joint CTC decoding.

4. VAD-free streaming inference

In the AED models, it is crucial to keep decoding contexts to improve the recognition performance because of the conditional dependency of output symbols. However, it is required to exclude long samples from the training data to fit the GPU/TPU memory and perform efficient training. Therefore, when recognizing long-form speech during inference, the models must generalize to unseen samples, but it is challenging in general [4].
4.1. Proposed algorithm

To balance the limitation of the generalization capability to long-form speech and the effective context management, we determine a suitable timing to reset model states based on CTC probabilities. We regard consecutive blank tokens (∅) generated from the CTC branch as a silence region and use them to find a reset point. Unlike a CTC-based pre-segmentation in [20], our method does not perform the segmentation explicitly, i.e., we do not detect the onset. Instead, we recognize all frames including long silence. Therefore, we do not have to wait for the pre-segmentation to be completed to start recognition, leading to latency reduction.

The proposed algorithm is shown in Algorithm 2. We adopt the block-synchronous decoding in Section 3. We count the number of consecutive blank tokens n∅ from the previous reset point and detect the next reset point when n∅ surpasses a threshold N∅ (condition 1, line 15). We also regard a weak non-blank spike whose probability is less than Pspike as a blank token. Note that the recognition continues until the end of the current block regardless of the result of the reset point detection. Moreover, we also allow resetting the states when (eos) is generated (condition 2, line 22). This is important for determining the reset point when enough silence is not found for a while. Once a reset point is detected, we push the most probable hypothesis to Ω+ to a session-level hypothesis set Ωsession and reset both decoder states and Ω+ (line 20-27). When using LSTM encoders, we also reset the encoder states. To deal with speech frames around the block boundaries, we re-encode acoustic features in the previous block after the state reset and use the last states as the initial states in the current block (back-off initialization). When using Conformer encoders [38], however, we do not have to reset the encoder states because they are agnostic to input offsets thanks to relative positional encoding and time-restricted self-attention. To avoid frequent state resets, we introduce a safeguard in which the reset point detection continues until enough silence is not found for a threshold N∅ [10ms] (line 29). Moreover, LSTM LM states are carried over to the next block to provide useful contexts before the reset point.

5. Experimental evaluations

5.1. Experimental setup

We used the TEDLIUM release v2 (TEDLIUM2) [39] and the Corpus of Spontaneous Japanese (CSJ) [40]. TEDLIUM2 consists of about 210-hour English lecture speech. CSJ consists of about 600-hour Japanese spontaneous academic lecture speech. We combined three official test sets in CSJ and 80-channel log-mel filterbank coefficients computed with a 25-ms window shifted every 10ms with Kaldi [41].

We investigated three kinds of encoder architectures: unidirectional LSTM (UniLSTM), latency-controlled bidirectional LSTM (LC-BLSTM) [42], and latency-controlled Conformer (LC-Conformer) encoders [43]. The UniLSTM consisted of five layers of LSTM with 1024 units. The LC-BLSTM had 512 units in both directions. We set both the current and right block sizes to 40, i.e., 400ms. The LC-Conformer encoder had the same architecture as Conformer (M) [38] with a kernel size of 15, while the number of layers was reduced from 16 to 12. We used hierarchical downsampling [44] with the max-pooling function.

5.2. Results

5.2.1. Utterance-level evaluation

We first compare the type of beam search decoding with the ground-truth segmentation in Table 1. Notice that we did not use CTC probabilities to reset the model states here. Using the UniLSTM encoder, we confirmed comparable WER with the block-synchronous decoding compared to the label-synchronous one on the TEDLIUM2 dev set while it was slightly degraded on the test set. Minimum WER training [45] could mitigate the degradation. In contrast, we observed WER reduction down to the block size of 240ms and 400ms on the CSJ dev and test sets, respectively. A possible explanation is that the block-synchronous decoding introduced some effective having a stride of 2 at the last frontend CNN layer, 4th, and 8th Conformer blocks. We also replaced batch normalization in each convolution module with layer normalization [32]. We adopted the masking strategy in [43], where lookahead frames were truncated in the same block, including the frontend CNN layers. We set the left and current block sizes to 960ms and 320ms, respectively. Therefore, the average algorithmic latency (collective) on the MoChA was 5.16s, 4.78s, and 4.30s on the TEDLIUM2, UniLSTM, and Conformer models on TEDLIUM2 with λsync = 0.1 and λsync = 2.0. During inference, we used B = 10 with a four-layer LSTM LM. We set (N∅, N∅, Ps(x = 0), R⟩) to (1000, 0.0, 1.0). The codes are publicly available [46].

We applied CTC-ST [17] to all LSTM models with λsync = 1.0 in Eq. (2). Moreover, StableEmit [32] was applied to the UniLSTM and Conformer models on TEDLIUM2 with λsync = 0.1 and λsync = 2.0. During inference, we used B = 10 with a four-layer LSTM LM. We set (N∅, N∅, Ps(x = 0), R⟩) to (1000, 0.0, 1.0). The codes are publicly available [46].

5.2.2. Utterance-level evaluation

5.2.1. Utterance-level evaluation

We first compare the type of beam search decoding with the ground-truth segmentation in Table 1. Notice that we did not use CTC probabilities to reset the model states here. Using the UniLSTM encoder, we confirmed comparable WER with the block-synchronous decoding compared to the label-synchronous one on the TEDLIUM2 dev set while it was slightly degraded on the test set. Minimum WER training [45] could mitigate the degradation. In contrast, we observed WER reduction down to the block size of 240ms and 400ms on the CSJ dev and test sets, respectively. A possible explanation is that the block-synchronous decoding introduced some effective

having a stride of 2 at the last frontend CNN layer, 4th, and 8th Conformer blocks. We also replaced batch normalization in each convolution module with layer normalization [32]. We adopted the masking strategy in [43], where lookahead frames were truncated in the same block, including the frontend CNN layers. We set the left and current block sizes to 960ms and 320ms, respectively. Therefore, the average algorithmic latency (collective) on the MoChA was 5.16s, 4.78s, and 4.30s on the TEDLIUM2, UniLSTM, and Conformer models on TEDLIUM2 with λsync = 0.1 and λsync = 2.0. During inference, we used B = 10 with a four-layer LSTM LM. We set (N∅, N∅, Ps(x = 0), R⟩) to (1000, 0.0, 1.0). The codes are publicly available [46].

We applied CTC-ST [17] to all LSTM models with λsync = 1.0 in Eq. (2). Moreover, StableEmit [32] was applied to the UniLSTM and Conformer models on TEDLIUM2 with λsync = 0.1 and λsync = 2.0. During inference, we used B = 10 with a four-layer LSTM LM. We set (N∅, N∅, Ps(x = 0), R⟩) to (1000, 0.0, 1.0). The codes are publicly available [46].

Having a stride of 2 at the last frontend CNN layer, 4th, and 8th Conformer blocks. We also replaced batch normalization in each convolution module with layer normalization [32]. We adopted the masking strategy in [43], where lookahead frames were truncated in the same block, including the frontend CNN layers. We set the left and current block sizes to 960ms and 320ms, respectively. Therefore, the average algorithmic latency (collective) on the MoChA was 5.16s, 4.78s, and 4.30s on the TEDLIUM2, UniLSTM, and Conformer models on TEDLIUM2 with λsync = 0.1 and λsync = 2.0. During inference, we used B = 10 with a four-layer LSTM LM. We set (N∅, N∅, Ps(x = 0), R⟩) to (1000, 0.0, 1.0). The codes are publicly available [46].

We applied CTC-ST [17] to all LSTM models with λsync = 1.0 in Eq. (2). Moreover, StableEmit [32] was applied to the UniLSTM and Conformer models on TEDLIUM2 with λsync = 0.1 and λsync = 2.0. During inference, we used B = 10 with a four-layer LSTM LM. We set (N∅, N∅, Ps(x = 0), R⟩) to (1000, 0.0, 1.0). The codes are publicly available [46].

We applied CTC-ST [17] to all LSTM models with λsync = 1.0 in Eq. (2). Moreover, StableEmit [32] was applied to the UniLSTM and Conformer models on TEDLIUM2 with λsync = 0.1 and λsync = 2.0. During inference, we used B = 10 with a four-layer LSTM LM. We set (N∅, N∅, Ps(x = 0), R⟩) to (1000, 0.0, 1.0). The codes are publicly available [46].
We conduct the ablation study of the VAD-free inference algorithm on TEDLIUM2 in Table 2. We concatenated all utterances in each evaluation set. We observed that the safeguard had the largest impact, indicating that MoChA is more likely to generate (eos) at the end of blocks with the block-synchronous decoding. Length normalization was also important for longer hypotheses to rank at the top. Back-off initialization was important for the universal framework eliminates the need for developing separate encoders. Unlike the simulated experiments, there exist a lot of silence frames in the real recordings. We compared the proposed method with the pre-segmentation with WebRTC VAD because both corpora do not necessarily contain all transcriptions in a session, we recognized all frames but removed tokens that did not match the ground-truth segments to calculate WER. We also evaluated the models trained on TEDLIUM2 with the IWSLT tst2013, tst2014, and tst2015 sets. Results in Table 3 showed that the proposed VAD-free inference achieved comparative WERs to those with the ground-truth segmentation on TEDLIUM2. Although the results on CSJ degraded slightly compared to when using the ground-truth segmentation, they were much better than cascading VAD and ASR models. Moreover, the degradation was much smaller than the CTC-based pre-segmentation [20,21], which had a large latency to start recognition. We observed that the VAD model was more likely to generate short segments that did not suit E2E models, especially for the LC-Conformer encoder. This was because the LC-Conformer encoder had a total history context of 11.52 seconds. Although there is room for improving VAD, the proposed unified framework eliminates the need for developing separate models independently.

5.2.4. Session-level evaluation on real long-form recordings

Finally, we conduct experiments on the real session-level lecture recordings. Unlike the simulated experiments, there exist a lot of silence frames in the real recordings. We compared the proposed method with the pre-segmentation with WebRTC VAD because both corpora do not necessarily contain all transcriptions in a session, we recognized all frames but removed tokens that did not match the ground-truth segments to calculate WER. We also evaluated the models trained on TEDLIUM2 with the IWSLT tst2013, tst2014, and tst2015 sets. Results in Table 3 showed that the proposed VAD-free inference achieved comparative WERs to those with the ground-truth segmentation on TEDLIUM2. Although the results on CSJ degraded slightly compared to when using the ground-truth segmentation, they were much better than cascading VAD and ASR models. Moreover, the degradation was much smaller than the CTC-based pre-segmentation [20,21], which had a large latency to start recognition. We observed that the VAD model was more likely to generate short segments that did not suit E2E models, especially for the LC-Conformer encoder. This was because the LC-Conformer encoder had a total history context of 11.52 seconds. Although there is room for improving VAD, the proposed unified framework eliminates the need for developing separate models independently.

6. Conclusions

In this work, we have proposed the block-synchronous beam search decoding and the VAD-free inference algorithm to recognize unsegmented long-form speech with the hybrid CTC/MoChA framework. Experimental evaluations on English and Japanese lecture corpora demonstrated that the proposed decoding method enabled stable recognition of long-form speech with a linear-time decoding complexity. It was more accurate than performing VAD with an external model.

Table 1: Comparison of beam search type for MoChA with the ground-truth segmentation

| Encoder | Output synchronization | $T_{\text{block}}$ | TELDUM2 dev | TELDUM2 test | CSJ dev | CSJ test |
|---------|------------------------|--------------------|-------------|-------------|--------|---------|
| Label   | $\infty$              | 11.7/10.9/6.6/7.5 |             |             |        |         |
| UniLSTM | Block 64              | 11.6/11.7/6.4/7.3 |             |             |        |         |
|         | Block 40              | 11.6/11.6/6.4/7.4 |             |             |        |         |
|         | Block 32              | 11.6/12.0/6.5/7.5 |             |             |        |         |
|         | Block 24              | 11.8/12.2/6.5/7.6 |             |             |        |         |
|         | Block 16              | 12.0/12.6/6.6/7.7 |             |             |        |         |
|         | Frame 4               | 13.2/13.6/8.0/9.7 |             |             |        |         |

Table 2: Ablation study with UniLSTM MoChA on simulated long-form recordings of TEDLIUM2 (all concatenated)

| Encoder | $T_{\text{block}}$ | VAD | TELDUM2 dev | TELDUM2 test | IWSLT tst2013 | IWSLT tst2014 | IWSLT tst2015 |
|---------|-------------------|-----|-------------|-------------|---------------|---------------|---------------|
| UniLSTM | 32                |     | 11.4        | 10.7        | 13.3          | 12.9          |               |
| LC-BLSTM| 40                | $\times$ | 11.9        | 11.7        | 19.7          | 19.7          | 10.9          |
| LC-Conformer | 32 | $\checkmark$ | 12.9       | 12.0        | 12.9          | 12.0          |               |

Table 3: Session-level results with real long-form recordings

| Encoder | $T_{\text{block}}$ | VAD | TELDUM2 test | IWSLT tst2013 | IWSLT tst2014 | IWSLT tst2015 |
|---------|-------------------|-----|-------------|---------------|---------------|---------------|
| UniLSTM | 32                |     | 10.9        | 8.3           | 8.3           | 13.1          |
| LC-BLSTM| 40                | $\times$ | 8.8         | 16.9/15.1/29.7 | 8.7         |               |
| LC-Conformer | 32 | $\checkmark$ | 8.9       | 16.4/15.1/30.1 |               |               |
| UniLSTM | 32                |     | 26.1        | 13.1          |               |               |
| LC-BLSTM| 40                | $\checkmark$ | 18.3       | 25.5/23.0/37.1 | 11.9          |               |
| LC-Conformer | 32 | $\checkmark$ | 34.9 | 37.3/36.2/43.5 |               |               |

Figure 1: WER on simulated long-form recordings on the TEDLIUM2 test set. All models used block-synchronous decoding. The UniLSTM model used $T_{\text{block}} = 32$.
